Title

Data-Driven Real-Time Risk Predictive Intelligence — A Use Case of Go-Arounds

Author

Dai, Lu

Publication Date

2022

Peer reviewed|Thesis/dissertation

Data-Driven Real-Time Risk Predictive Intelligence – A Use Case of Go-Arounds

By

Lu Dai

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Mark Hansen, Chair
Professor Joan Walker
Professor Maximilian Auffhammer

Spring 2022

Data-Driven Real-Time Risk Predictive Intelligence – A Use Case of Go-Arounds

Abstract

Data-Driven Real-Time Risk Predictive Intelligence – A Use Case of Go-Arounds

by

Lu Dai

Doctor of Philosophy in Engineering – Civil and Environmental Engineering

University of California, Berkeley

Professor Mark M. Hansen, Chair

Although the National Airspace System is one of the safest and most efficient transportation infrastructures, growing air traffic demand and the implementation of autonomous technologies place strain on its safety and efficiency. Recent advances in computing and artificial intelligence (AI) offer an unprecedented capability to incorporate intelligent decision-making into a wide variety of spheres of our life, most notably for big and practical engineering problems. In this dissertation, we develop data-driven real-time risk predictive intelligence to provide decision support for the air transportation system, particularly for the critical air traffic control process during flight approach and landing.

Anomalous aircraft behaviors and states are of high interest to the aviation community and hold the keys to ensuring safe, efficient, and environmentally clean flight operations. Through one type of flight anomaly – go-arounds (the aborted landing of an aircraft on final approach) – this dissertation demonstrates the complex interplay between transportation engineering and AI: from theoretical study and algorithmic development, to the computer and software systems, and to the eventual deployment. We investigate the concrete technical and operational challenges of building risk predictive intelligence by integrating a blend of advances from data science, machine learning, software systems, domain-specific sciences and engineering knowledge. While this dissertation focuses on the aviation domain, the established methodological framework has the potential in many other contexts to assess the risk of non-nominal events.

We first design a trajectory-based anomaly detection algorithm for identifying go-around events from raw and noisy surveillance data. The current practice of go-around detection mainly relies on voluntary self-reports from controllers or pilots, unrepresentative survey/interview data, or a limited sample of simulation/training data. We therefore propose a rigorous way of detecting go-around occurrence by analyzing historical four-dimensional flight trajectories. This algorithm not only labels the flight in binary responses but also annotates when and where the go-around occurred. We further validate the detection results with another independent data source and find that our detection

algorithm identifies more true positive events since it can capture go-arounds initiated farther away, and with more robust criteria.

In order to capture the heterogeneous interacting components that may affect the go-around occurrence, feature engineering is carried out to derive a wide variety of operational and environmental variables according to literature search, theoretical studies, interviews with domain experts, and data mining. Among the seven categories of features derived – aircraft characteristics, approach stability, in-trail separation, weather, airport conditions, go-around clustering effects, and runway incursion risk, we propose a new metric termed runway occupancy buffer (ROB) to better reflect air and surface operations interplay during flight approach. We train machine learning models to predict this metric conditioned on other categories of features. The predicted value not only serves as a feature input for modeling go-arounds, but may also directly assist air traffic control in maintaining safe, efficient buffers between successive arrivals.

With the labeled events and derived features, we then investigate the traffic and environmental conditions that affect go-around occurrence by quantifying their underlying contributions through principal component logistic regression and counterfactual analysis. While previous studies have investigated various causes of go-around occurrence, none has developed a comprehensive, quantitative assessment of the relative importance of a wide range of factors. Our method overcomes the high dimensionality and multi-collinearity of the original data set while preserving the ability to assess the contribution of the original features to go-around occurrence. We find that factors in the top tier of importance include the approach stability of the subject aircraft, its separation and speed difference from the aircraft in front, and factors related to visibility and cloud ceiling.

While the post-event observation-driven insights help decision-making at a strategic level, being able to predict go-around probabilities could provide tactical guidance to foresee and perhaps prevent go-arounds. Existing models on go-around predictions are based on a single snapshot of features in the time series process. We fill the gap by developing machine-learning-based engines for multivariate sequential predictions of go-around probabilities over the entire approach. The sequential models exhibit a consistent and monotonically increasing performance as more information is preserved in the internal state when the flight gets closer to the airport. The LSTM, in general, performs better in predicting go-around occurrence thanks to their continuous hidden state space and ability to learn dependencies.

To address the class imbalance issue inherent with the go-around prediction problem, or for any rare event prediction, data augmentation is explored to generate high-fidelity synthetic go-around sequences for improved model training. In particular, we synthesize domain-specific insights with concurrent advances in the Generative Adversarial Networks (GANs) literature to design a GAN architecture for the go-around use case, capable of generating multivariate sequences with variable length and mixed data types. Empirically, we find that this architecture improves the fidelity of the generated go-around sequences, in terms of sequence length, feature distribution, and serial correlation. The performance of the go-around prediction model is compared with different amounts of synthetic go-

arounds added to the training set. Experimental results show that models trained with 30% go-around samples perform better. Further efforts on model development and generalization are required for researchers to confidently use such workflows.

We additionally present the Go-Around Prediction (GAP) software service, which encapsulates all these pieces of work into a practical application system to provide real-time guidance to air traffic control and ease the future design of risk predictive intelligence. To enable the GAP capabilities, we build the real-time data injection pipeline atop Apache Software Service, ensure pre-trained models can be promptly executed in response to real-time messages, identify suitable test scenarios for the real-time emulation demonstration, and develop a web-based user interface to display the real-time representation of the go-around prediction results. We demonstrate the feasibility and practicality of the GAP service by applying it to a real-world test scenario, with the end-to-end real-time data input and go-around detection output. The GAP software system provides a foundation for designing, developing, and deploying a progression of capabilities that expedites the discovery, prognosis, and mitigation of safety-related threats in transportation systems.

Together, various components of this dissertation work are closely interconnected to enable data-driven real-time risk predictive intelligence, while at the same time, each component offers its own contribution. The methodology framework includes the anomaly detection algorithm to identify risky events from unlabeled data, statistical models to uncover and quantify the factor contributions to the event occurrence, generative adversarial networks to augment the minority class, sequential learners to continuously monitor developing risks, and a data streaming pipeline for real-time deployment. It advances the state-of-the-art and is the first effort in realizing a multi-domain situational awareness, predictive, and alerting tool for go-around occurrences, therefore an end-to-end actionable solution to practitioners. In the spirit of near-term practicality, we offer low-cost building blocks that can be used for other real-world applications with data of similar structure, such as risk mitigation in future transportation systems where complexity is expected to be greater with the introduction of autonomous vehicles and urban air mobility into the legacy infrastructure. In view of long-term applicability, the dissertation work holds initial promise to inspire more and further research by theoreticians and practitioners to develop data-driven real-time solutions to predictive intelligence in a broader domain.

To my family.

Table of Contents

List of Abbreviations

AAR – Airport Arrival Rate

ADASYN – Adaptive Synthetic sampling

ADR – Airport Departure Rate

AI – Artificial Intelligence

ANSP – Air Navigation Service Providers

APTC – Airport Configuration

ASDE-X – Airport Surface Detection Equipment Model X

A-SMGCS – Advanced Surface Movement Guidance & Control System

ASOS – Automated Surface Observing System

ASPM – Aviation System Performance Metrics

ASRS – Aviation Safety Reporting System

ATC – Air Traffic Control

AV – Autonomous Vehicles

AWC – Aviation Weather Center

BPTT – Back-Propagation Through Time

CDT – Complete Disjunctive Table

CGAN – Conditional Generative Adversarial Network

DIP – Digital Information Platform

EM – Expectation Maximization

ERC – Extended Runway Centerline

ETA – Estimated Time of Arrival

FCN – Fully Convolutional Network

FN – False Negative

FP – False Positive

FSF – Flight Safety Foundation

GAN – Generative Adversarial Network

GAP – Go-Around Prediction

HMM – Hidden Markov Model

IATA – International Air Transport Association

IASMS – In-Time Aviation Safety Management System

IFF – Integrated Flight Format

IFR – Instrument Flight Rules

IMC – Instrument Meteorological Conditions

IOHMM – Input Output Hidden Markov Model

ISSA – In-Time System-Wide Safety Assurance

JMS – Java Message Service

LASSO – Least Absolute Shrinkage and Selection Operator

LOS – Loss of Separation

LSTM – Long Short-Term Memory

MAPt – Missed Approach Points

METAR – Meteorological Aerodrome Report

ML – Machine Learning

MLE – Maximum Likelihood Estimation

MLP – Multi-Layer Perceptron

MSE – Mean Squared Error

NAS – National Airspace System

NASA – National Aeronautics and Space Administration

NM – Nautical Mile

NTSB – National Transportation Safety Board

OAG – Official Airline Guide

OLS – Ordinary Least Squared

PAPI – Precision Approach Path Indicator

PC – Principal Component

PCLR – Principal Component Logistic Regression

RBF – Radial Basis Function

RF – Random Forest

RMSE – Root Mean Squared Error

RNN – Recurrent Neural Network

ROB – Runway Occupancy Buffer

ROT – Runway Occupancy Time

RSA – Runway Safety Area

RSS – Residual Sum-of-Squares

RTA – Remaining Time to Arrival

SDS – Skyview Data Service

SME – Subject Matter Expert

SMOTE – Synthetic Minority Oversampling Technique

SRO – Simultaneous Runway Occupancy

SVD – Singular Value Decomposition

SVM – Support Vector Machine

SWIM – System Wide Information Management

SWS – System-Wide Safety

TAAM – Total Airspace and Airport Modeler

TN – True Negative

TP – True Positive

TRACON – Terminal Radar Approach Control

UAM – Urban Air Mobility

VAE – Variational Autoencoders

VFR – Visual Flight Rules

VMC – Visual Meteorological Conditions

XAI – Explainable Artificial Intelligence

XGBoost – Extreme Gradient Boosting

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest gratitude to my intelligent, energetic, caring, youthful advisor, Mark Hansen, for his constant support of my research and professional growth. His insightful advice, enthusiastic encouragement, broad vision, open-mindedness, and continuous support are what made this dissertation research possible and enjoyable. As a research and life advisor, and an incredibly supportive friend, Mark has always been there for me. He sparked my interest in air transportation, introduced me to the glamorous scientific world, offered excellent guidance on my study, directed me to proceed with my research, kept me on the right track, took me through the darkness, helped open doors for me, and shaped my academic trajectory. It is through his guidance that I developed essential research skills and critical thinking abilities. It is his patience, trust, companionship, support, and inspiration that helped me be a better person. As the song "For Good" says, *because I knew you, I have been changed for good.* Mark is my rock to be brave in facing all the hurdles in life, my harbor allowing me to show my timidity and insecurity, and my beacon navigating me to grope for my own proud path.

I am extremely grateful to my qualifying exam and dissertation committees, Mark Hansen, Joan Walker, Maximilian Auffhammer, and Michael Cassidy, for their inspiring insights and invaluable feedback on this thesis work. I would like to thank Jonathan R. Shewchuk and Sergey Levine from the computer science department for the insightful conversations as I sought to bring machine learning into my research. I am grateful to all Berkeley faculty members who dedicated their time and expertise with me through lectures, talks, projects, or simply casual conversations.

I sincerely appreciate the considerable research support from NASA, FAA, Lawrence Berkeley National Lab, and Robert P. Wadell. This dissertation work was part of the project "SMART NAS NRA: Big Data Analytics for Aeronautics" (NNA16BE49C) supported by the NASA Ames Research Center, and was subsequently supported by the NASA Small Business Innovation Research program to develop the "Go-Around Prediction Service" (80NSSC21C0097). I am grateful for the valuable inputs and feedback from the advisory team led by Nikunj C. Oza and Robert W. Mah. Thanks to the FAA for supporting me in investigating flight delays and traffic flow management. Thanks to the LBNL for inviting me to examine the traffic influences on the pandemic spread. These projects have added so much fun to my life. I also want to express my sincere appreciation to Robert P. Wadell for his Endowed Fellowship for Civil Engineering Innovation (2019 – 2021), which funded my PhD study and gave me the freedom to explore many important research problems.

I am also indebted to John Schade from ATAC, who has pushed tirelessly for this dissertation research. John has been incredibly supportive of my work and has led the whole research team towards the right direction. I would also like to thank John Schade for his help with running the real-time software system, and Joseph Robinson for the validation analysis of the go-around detection.

I am also thankful for Michael Hanowsky and William J. Dunlay for mentoring me at Jacobs Engineering Group (LeighFisher Inc.) and ever since. Thanks to Ralf

Ruckelshausen and Jack Bell from the SFO Safety & Security Services office for their motivation in analyzing go-arounds, which has made a strong start for this dissertation.

I am eternally grateful for a group of close colleagues who warmly welcome me to the aviation community. Thanks to Michael Ball and David Lovell at UMD, Megan S. Ryerson at UPenn, Yu Zhang at USF, Bo Zou at UIC, and Yanjun Wang at NUAA for sharing thoughts and perspectives with me. I am also thankful for Frank Ketcham, a Captain with Delta Airlines, who has consistently contributed operational insights into my research and helped me ground this dissertation work in practicality.

I have been extremely fortunate to have worked and learned with my amazing collaborators: Nikunj C. Oza, Robert W. Mah, and Paul Krois from NASA, John Schade and Kennis Chan from ATAC, Joseph Post, Dave Knorr, Brian Bagstad, Kerry Capes, Garry Cohen, Carlos Gonzalez, Leo Prusak, Guillermo Sotelo, and Jim Wetherly from FAA, Eoin Brodie, Chaincy Kuo, James B. Brown, Haruko Wainwright from the Lawrence Berkeley National Lab.

Thanks to all the wonderful staff within CEE and ITS, especially Shelley Okimoto, who have made my PhD journey a breeze. I am grateful to the MIT community for recognizing me as the CEE "Rising Stars" and connecting me with outstanding scholars for scientific interactions. I am also thankful for Meta (formerly Facebook) for offering me opportunities to meet people from diverse professions and broaden my view on AI.

I feel very fortunate to have so many friends from diverse ethnicities and backgrounds journeyed with me, both in and outside research. I would like to thank Andrew Foertsch, who has been one of my first friends here and the most passionate person I know, for helping me overcome culture shock. I would like to thank Megan Ryerson, the most optimistic and compassionate person I know, for setting a role model for me as a female researcher in Engineering. I have tremendously enjoyed the many invigorating discussions, refreshing perspectives, enjoyable collaborations, and unceasing explorations with Yulin Liu and Ivan Tereshchenko on transportation, statistics, machine learning, and optimization. I have received considerable help on predictive analytics from Mogeng Yin, Jingchao Zhou, and Ashank Verma. I deeply appreciate the wise counsel and valuable conversations with Chao Mao, Mengqiao Yu, Tianhao Zhang, Ruoying Xu, Yang Ju, Xize Wang, Dounan Tang, Timothy Brathwaite, and Feras Zarwi.

I am incredibly thankful to my sweet, considerate, gentle, empathetic partner, Ke Xu, for holding my hands from the beginning of this journey and for his love, support, and respect that have enriched my life greatly.

Lastly, I dedicate this dissertation to my dearest grandma, Kangxiu Xian, my parents – Changbin Dai and Defen Huang, my brother – Yijie Dai, my grandparents – Shiliang Dai, Buzhen Zhang, Shaoxi Huang, Yongzhen Xu, and Beipin Zou, my uncles – Dehai Huang, Yan Xu, Changping Dai, Changge Dai, and Changxu Dai, and my aunts – Fajiao Dai, Qiuyan Dai, and Deyan Huang, who never dreamed that their names would be on a doctoral dissertation, and whose endless love, enduring encouragement, and persistent support made everything possible.

1. Introduction

1.1. Safety First

Transportation systems today have served as the backbone of civilization. They touch the lives of 7.9 billion people and support 1.4 billion motor vehicles globally [1]. At any given time, there are an average of 9,728 planes carrying 1 million passengers in the sky [2]. The system is still expanding in scale, preparing to welcome new entrants such as unmanned (aerial) vehicles, and to safely incorporate new paradigms such as autonomous driving and urban air mobility. The transportation system is being revolutionized through advances in automation, electrification, and new business models, while also facing challenges posed by decarbonization, inequity, pandemic, and so on. There is a critical need to continue ensuring the safety and efficiency of the system. Such importance is also evident in the high societal costs associated with accidents, as well as the potential threats to the legacy transportation infrastructures. According to the American Society of Civil Engineers, 43% of public roads are poor or mediocre [3]. Construction delays, signage issues, and asphalt deterioration are just a few of the things contributing to an increase in crashes and accidents.

Although the National Airspace System (NAS) is one of the safest and most efficient transportation infrastructures, growing air traffic demand and the implementation of autonomous NextGen technologies place strain on NAS safety and efficiency. One of the primary goals of all air navigation service providers (ANSPs) is to assure safety since each aircraft operation involves human life. Safety always comes first for any airline in all aspects of air transportation.

Indeed, nothing can ever guarantee absolute safety in a moving airplane. From birds, to weather (e.g., wind shear, snowstorms, lightning), malfunction (e.g., engine failure, aircraft stalling), human factors (e.g., runway incursion, miscommunication), and even terrorism, the hurdles to safe flying can seem insurmountable. The aviation community has spent decades of effort to reduce the risks of flying and make accidents become exceedingly rare events. Boeing analyzed worldwide commercial flights from 1959 to 2017 and found that the annual number of fatalities has remained fairly low and stable [4]. The total fatalities due to aviation accidents since 1970 is 83,772. The total number of incidents is 11,164. According to the Aircraft Crashes Record Office (ACRO), aviation has been much safer in recent years, with less than 170 incidents per year between 2009 and 2021, compared to as many as 226 as recently as 1998 [5]. It is noteworthy that most of these fatalities and accidents happened during the final approach and landing phases, which account for just 16% of the airborne time (total time spent in the air). As seen in Figure 1, from 2008 to 2017, 56% of the fatalities and 62% of the accidents happened during the approach and landing, making this the riskiest phase of flying.

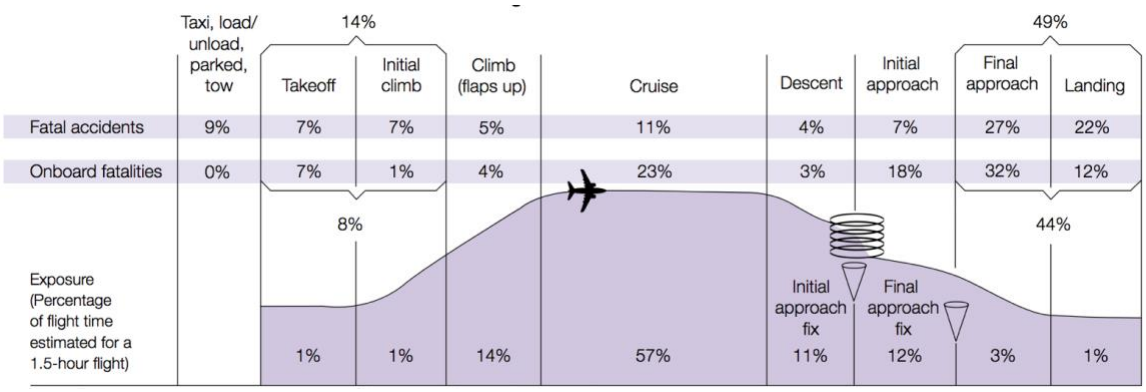

Figure 1. Percentage of fatal accidents and fatalities. Source: Statistical summary of commercial jet airplane accidents conducted by Boeing Commercial [4].

The literature has devoted substantial attention to flight approach safety. Until recently, though, it has not received much attention from data scientists. Although extensive data from radar and surveillance systems, navigation systems, engines and sensors, text documents, and vocal crew inputs are continually collected and stored, current analytics for flight approach safety are reactive [6, 7, 8, 9, 10], rule-based [11, 12, 13], and narrow in scope (considering only flightpath profile[14, 15, 16], or weather conditions [17, 18], or human factors [19, 20, 21], etc.). Research on real-time risk predictive analytics is required to enable proactive monitoring and prediction of anomalous flight approach procedures that may degrade safety and efficiency, while considering a broad range of situational measurements and dynamic environmental conditions. With the recent publishing of the In-Time System-Wide Safety Assurance (ISSA) strategic roadmap [22] and the launch of the NASA System-Wide Safety (SWS) project [23, 24], the aviation community has reemphasized the commitment to safety research. A key outcome in the first decade of ISSA-related research is *improved safety through initial real-time detection and alerting of hazards at the domain level and decision support for limited operations*.

1.2.Go-Arounds

Typically, the approach-and-landing accident is triggered by an unstabilized approach and is the consequence of a subsequent failure to initiate a go-around. Go-arounds occur when an arrival aircraft terminates its normal approach to landing, reverses its descent by climbing abruptly, and then circles around to attempt another landing. In other words, a go-around is an aborted landing of an aircraft that is on the final approach when proceeding with the landing is considered to be unsafe due to adverse conditions such as wind shear, runway incursion, or unstabilized approach.

A running example throughout this dissertation concerns this unique maneuver – go-arounds. Go-arounds play a special role in the final approach and landing phase as their occurrence not only signals abnormal flying status on the final approach (pre-go-around

risks), but themselves are a kind of operational anomaly (go-around risks). Overall, go-arounds or missed approaches occur at a rate of around three or four per thousand arrival operations at major airports – see Figure 2 for a count of go-arounds at major US airports in 2018 through 2020. Even with the drastic reduction in operations due to COVID-19 in 2020, the number of go-arounds at many major airports remained consistent with or exceeded prior years. From 2012 to 2017, the average percentage of final approaches leading to go-arounds was 0.4% across the core 30 U.S. airports [25].

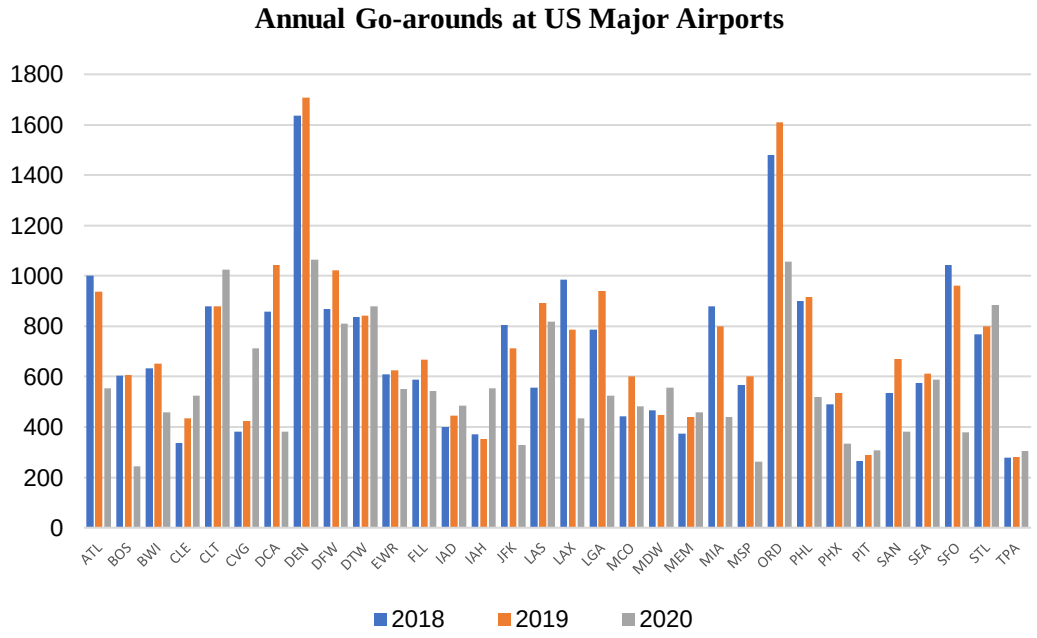

Figure 2. Total annual go-arounds at U.S. airports.

In addition, there have been several high-profile incidents in the last few years in which go-arounds were used as the mitigation of last resort:

At 11:56 PM on July 7, 2017, at the conclusion of the regularly scheduled international passenger flight from Toronto to San Francisco, Air Canada Flight 759 (AC759), an Airbus 320, nearly landed on a taxiway adjacent to the designated landing runway 28R at San Francisco International Airport (SFO). Four fully loaded and fueled aircraft waiting for takeoff have occupied the taxiway. Five airplanes carrying over 1,000 passengers were at imminent risk when the incident occurred. It was later estimated by National Transportation Safety Board (NTSB) investigators that the Air Canada flight descended below 100 feet above ground level before pulling up to execute the go-around.

At 9:30 PM on October 24, 2017, Air Canada Flight 781, an Airbus 320, was cleared for landing on runway 28R at SFO. Since the preceding aircraft that had landed on 28R vacated the runway more slowly than expected, the Tower air traffic controller instructed Air Canada 781 to go around. However, the aircraft crew of Flight 781 never responded. Flight 781 was ordered multiple times to go around without responses. Controllers even flashed red runway lights as a visual indication to the crew that they should abort the landing, but Flight 781 continued the approach to 28R. Fortunately, the preceding aircraft had managed to exit the runway just in time, averting a potentially deadly collision.

At 11:45 AM on January 9, 2018, Aero Mexico Flight 668, a Boeing 737 flying from Mexico City to San Francisco, was cleared to land on runway 28R at SFO. Instead, the Flight 668 aircraft lined up on the parallel runway 28L, where a Virgin America Airbus A320 jet was waiting to take off for Kona, Hawaii. Tower air traffic controllers noticed the inconsistency at the last minute and ordered Flight 688 to go around. Again, if not for the vigilance of the tower controllers, a massive aviation calamity may have happened.

These near-miss incidents, while alarming in nature, are even more significant by the fact that they all occurred at the same airport within around six months. In each of these instances, the Tower air traffic controllers were able to issue mitigations to the developing risk; nevertheless, these were of the "last call" variety. Catastrophe would have ensued without prompt responses from the operators (air traffic controllers or pilots). A robust go-around prediction and alerting capability would expand the time window for operator action and thus provide a greater safety margin for these sorts of operational incidents. Below, we summarize a few essential motivating aspects for studying go-arounds.

Go-arounds are associated with significant safety concerns during the final approach, since go-arounds are emergency maneuvers intended to mitigate risk compared to preceding the landing. For instance, some go-arounds occur due to attempted wrong surface landings. These situations occur when aircraft align with a taxiway rather than a runway and initiate a go-around, thus flying over the taxiway. In other instances, aircraft attempt to land on the incorrect runway and end up executing a go-around to avoid colliding with departing aircraft. In 2016, the FAA ATO identified wrong surface landings as a Top 5 safety issue. Wrong surface landings are still on the FAA high priority list. In order to improve communication and training on these incidents, and to mitigate risk in the NAS, the FAA conducted a study and found a clear linkage between wrong surface landing attempts and go-arounds. A technology capable of detecting go-around risk early enough, allowing mitigation actions to prevent safety hazards such as wrong surface landings, has yet to be developed.

In addition to pre-go-around risks, go-arounds themselves significantly disrupt airport operations and degrade efficiency, particularly at large airports during peak hours. To begin, even though pilots have extensive training, executing go-arounds is a challenging maneuver. The pilot must promptly accelerate the engines to full power, adopt a suitable climb altitude and airspeed, raise the landing gear, and retract the flaps in a short time. Second, the outcome of a go-around can be hazardous. Around 10% of go-arounds

result in exceeding aircraft performance limits, or fuel emergencies [26]. Finally, a go-around is an emergency maneuver that also leads to increasing air traffic controller workload [27] and noise [28], while degrading airport throughput [29] and flight on-time performance [26]. Air traffic controllers are confronted with the challenging tasks of de-conflicting the go-around from other traffic, including traffic landing and taking off from adjacent airports, as well as fitting the go-around aircraft back into the congested arrival stream. This time-sensitive task increases workload and potentially results in the development of additional dangers. Similarly, flight crews must contend with a surge in piloting workload [30] when they may be at the end of a busy flying schedule. During peak hours, the disruption induced by a go-around may ripple upstream into heavy inbound flows, causing substantial holding and vectoring that lead to rapidly accumulating delays. Any preemptive warning of an emerging go-around situation could buy the controller and flight crew valuable time to better anticipate the actions required to handle the impending situation more safely and efficiently.

Furthermore, making the decision to execute a go-around in a timely manner is crucial. According to an extensive study published by Flight Safety Foundation (FSF), delay in deciding to initiate a go-around increases the risk of this procedure [26]. The effectiveness of cooperation between pilots and air traffic controllers in making go-around decisions based on their anticipation of landing circumstances is of great importance and practical significance for boosting both the safety and efficiency of the aviation system. However, interviews indicate that the collaborative decisions on go-arounds are heavily impacted by individual experiences and mental states, as opposed to the collective knowledge about the complete picture of underlying situations for the go-around procedure [26]. Predicting the probability of go-arounds may help prevent situations where an incident occurs because a go-around was not executed. According to the research conducted by the FSF on 16 years of runway excursions, *83% could have been avoided with a decision to go around*.

1.3.Learning and Decision Making

Individuals, businesses, and governments now have unprecedented access to intelligent decision-making across a wide variety of domains because of the recent advancements in computing and artificial intelligence (AI). A wide range of applications benefit from decision support systems, from movie recommendations, to mapping services, traffic management, and so forth. Such systems are made possible by machine learning (ML), a field of artificial intelligence that has grown in popularity in the scientific community due to the increased availability of data, improvements in hardware and software, and various algorithmic breakthroughs.

In recent years, machine learning has emerged as a technique of paramount significance in the field of transportation engineering, enabling the solution of complex and practical engineering problems. Both ML and transportation have substantial effects on society. The

interaction between the two is a challenging, complex, and growingly important problem domain for society moving forward. Much of the challenge, and art, involved in machine learning is defining and formulating the problem. Successful machine learning applications have been seen in a wide variety of fields, including flow prediction, scheduling, pricing, routing, ridesharing, logistics, behavior modeling, networks, economics, and planning.

For the air transportation system to benefit from machine learning-based decision support, this dissertation is only a small step forward in enabling data-driven real-time risk predictive intelligence. We introduce one type of flight anomaly – go-arounds, develop anomaly detection algorithms to annotate this event from unlabeled data. Building on these labeled data, we devise generative adversarial networks to augment the minority class, sequential learners to continuously monitor developing risks, and a data streaming pipeline for real-time deployment. These problems have primarily been studied in isolation by various research communities using vastly different methodologies. Through the lens of a specific application – go-arounds – in the aviation field, this dissertation examines the intricate dynamics between all the moving elements involved in transportation engineering and artificial intelligence: from the theoretical and algorithmic development to the computer and software systems to the eventual application.

1.4. Objective and Challenges

Considering the pre-go-around risks, the safety of go-arounds themselves, and the effectiveness of decision-making for go-arounds, we would like to more fully understand the precursors and contributing factors to go-arounds, and provide aviation stakeholders (e.g., air traffic controllers and pilots) with a predictive tool to help them recover and avoid the need for a go-around or, if a go-around is needed, to provide them with more time to proactively manage the event, thereby increasing safety and the orderly management of traffic. The objective of the dissertation is to develop data-driven real-time risk predictive intelligence, through a use case of go-arounds, that brings intelligent decision support to the air transportation system, thereby making it safer and more efficient.

To implement such capability, we must continuously monitor the near-airport (within ten nautical miles) domain, merge data from disparate sources, and identify developing hazards (i.e., factors leading to a go-around) to stakeholders in advance, allowing them to take effective risk mitigation actions. Below, we discuss several challenges that must be overcome to achieve this objective:

- **Annotation quality.** The raw data does not explicitly designate go-around flights, which serve as ground truth for the study. Successful modeling is contingent upon the labeling process and the quality of annotated data. Finding a way to efficiently and reliably label go-around flights from a large set of trajectories is critical.

- **System heterogeneity.** The approach process consists of numerous heterogeneous interacting components, including the landing aircraft itself, other planes sharing the airspace, the wind, visibility, ceiling, runway conditions, etc. Each, alone or in combination with others, may produce different impacts on the flight approach. Careful feature engineering is needed to extract the information and capture the heterogeneity of the system.

- **Complex interactivity.** Go-arounds result from the interactions of numerous disparate factors. For example, the visibility of the landing environment can directly affect the spacing between airplanes, in addition to many other factors. What methodologies and tools would be appropriate for studying the high dimensionality and multi-collinearity of the data systematically, while retaining the ability to evaluate the contribution of the original features to go-around occurrence?

- **Dynamical process.** Air traffic control as a whole is a highly stochastic, cascading, nonlinear, hybrid process, in which a small change may induce complicated and delayed effects on the system. There are many ways in which uncertainties may be introduced into the prediction process due to the temporal and geographical resolution of the data. While building the predictors, it is crucial to ensure that the models are appropriately constructed, validated, and capable of capturing the complex dynamics of the approach sequences.

- **Class imbalance.** The primary difficulty with the go-around prediction problem, or with any rare event prediction task, is the imbalanced class distribution of the data. The class imbalance would considerably degrade the performance of any learning model, particularly with regard to the minority class we are interested in. Additional efforts are required to address the class imbalance issue and further enhance the work.

- **Real-time deployment.** In order to convert the developed model into a real-time tool, it must be combined with a real-time data ingestion mechanism. A distributed system is needed to offer unified, high-throughput, low-latency streaming pipelines for handling real-time data feeds.

The dissertation presents research that is intended to address these challenges, with a use case of go-arounds in the aviation domain. Several other challenges and research areas of interest to data-driven real-time risk predictive intelligence are discussed in Chapter 10. This work will heavily rely on a blend of advancements from data science, machine

learning, software systems, and domain-specific sciences and engineering knowledge. A science and a corresponding engineering practice are needed to lead the development of a multi-domain situational awareness and prognostic safety awareness, prediction, and alerting tool for go-around occurrences, and to assure long-term desired performance. The capability to deal with unlabeled, imbalanced, and real-time data holds significant promise for the applicability of AI, as this is the crux of many real-world problems, especially in safety-related sectors. While this dissertation focuses on the aviation domain, the methodologies developed have potential in many other transport (and non-transport) contexts in which time series data can be employed to assess the risk of non-nominal events.

In the spirit of near-term practicality, we aim to help Air Traffic Control (ATC) facilities and airport-focused personnel identify hazards in the approach domain in time for effective mitigation. This dissertation work has the significant potential to reduce the number of go-arounds at major airports and improve the operator's situational awareness and overall safety margin for these types of close-call incidents, hence reducing the risk of an accident. The predictive analytics will alert and aid with go-around decision-making, therefore lowering the rate of go-around failure. Additionally, early warning of an oncoming go-around occurrence provides more time to salvage the approach or better cope with an oncoming go-around. This additional reaction time should result in a significant reduction in workload for both controllers and flight crews.

1.5. Structure of the Dissertation

This dissertation focuses on one type of flight anomaly – go-arounds, and is a first step toward resolving the abovementioned challenges of risk predictive intelligence, by developing a methodological framework for studying go-arounds. As illustrated in Figure 3, the framework includes anomaly detection algorithms to identify go-around events from unlabeled data, generative adversarial networks to augment the minority class, sequential learners to continuously monitor developing risks, and a data streaming pipeline for real-time deployment. The remainder of this dissertation is structured in the following manner:

Chapter 2 introduces key concepts in the dissertation, and reviews previous research on go-arounds and, more broadly, operational anomalies during the flight approach. We focus on the research that examines how go-around decisions are made and the factors that trigger the occurrence of go-arounds, as well as the current practice of machine learning models.

Chapter 3 presents our method for detecting go-arounds from historical surveillance trajectories. The detection algorithm not only labels the flight in binary responses but also annotates when and where the go-around occurred. We present the process of ground truth labeling and validate it using another independent data source: the FAA go-around report. The work in this chapter was published in Dai et al. [31, 32].

Chapter 4 details the derivation of a diverse set of operational and environmental features, including aircraft characteristics (weight class, operating airline, landing runway), flight approach features (localizer deviation, speed, flight energy), the occurrence of other go-arounds at about the same time, in-trail separation (loss of separation, speed difference, altitude difference), as well as features pertaining to surface operations, airport configuration, and local weather. The work in this chapter was published in Dai et al. [31, 32, 33].

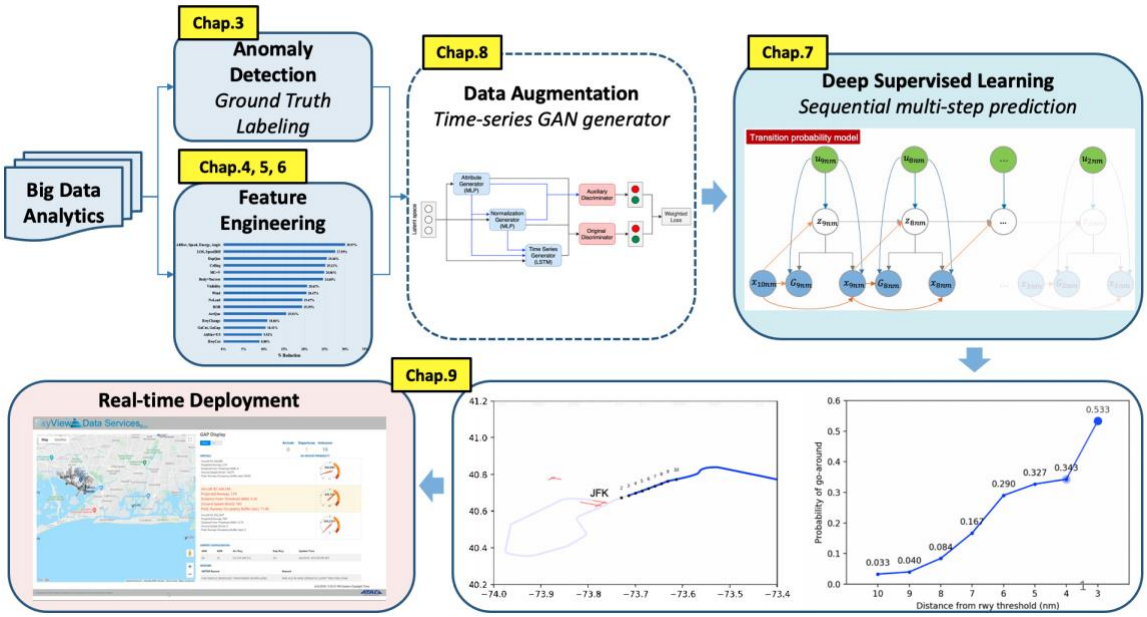

Figure 3. Data-driven real-time risk predictive intelligence, akin to the structure of the dissertation.

Chapter 5 proposes a new metric – runway occupancy buffer (ROB) – to better reflect the interplay between air and surface operations during flight approach procedures. We employ different machine learning models to model the ROB. The work in this chapter was published in Dai et al. [31, 32, 33].

Chapter 6 examines how a mix of traffic and environmental conditions (i.e., derived features in Chapter 4) affect the go-around decisions, and quantifies their underlying factor contributions to go-around occurrence based on real flight observations. We employ statistical models and counterfactual analysis to uncover the relationships between go-around occurrences and the derived features. With this analysis, we could answer the question: based on observations of historical flight operations, what are the most salient factors affecting go-around occurrence? This could be critical to decision-making at a strategic level. The work in this chapter was published in Dai et al. [31, 32].

Chapter 7 provides tactical guidance to foresee and perhaps prevent go-arounds by building the risk predictive analytics using the go-around detection results in Chapter 3, features derived from Chapter 4 and Chapter 5, and post-event observation-driven insights in Chapter 6. We formulate the go-around prediction problem as sequential learning and subsample the multivariate sequences with one nautical mile spacing (from the landing runway threshold). Different learning algorithms are employed to make sequential predictions of go-around probabilities of each landing aircraft over the entire approach corridor. The work in this chapter was published in Dai et al. [34].

Chapter 8 demonstrates our exploration towards tackling the class imbalance issue for the go-around prediction task: downsampling, sampling-based augmentation, and GAN-based augmentation. We propose a proof-of-concept Generative Adversarial Network (GAN) architecture capable of generating high-fidelity synthetic go-around sequences to augment the minority class. We develop metrics to assess the fidelity of the generated synthetic go-around sequences and further benchmark the model with a sampling-based method. The previously developed go-around prediction model performs better while augmented with GAN-generated samples.

Chapter 9 introduces the Go-Around Prediction (GAP) service, which enables the practical application of the above work in a real-time setting. Many advances are needed to enable real-time risk predictive intelligence, including algorithmic developments, distributed data streaming, and a web-based interface. We demonstrate the feasibility and practicality of the GAP service by applying it to the airport domain use case. A proof-of-concept demonstration is provided in a video format, with the end-to-end real-time data input and go-around detection output.

Chapter 10 presents a “flight plan” of the directions for future research. We discuss the remaining research questions and further considerations for deploying the data-driven real-time risk predictive intelligence for practical use.

1.6. Contributions

This dissertation is a first step in enabling data-driven real-time risk predictive intelligence, from theoretical study and algorithmic development to the computer and software systems, to the eventual deployment. By studying a use case in the aviation field – go-arounds, the dissertation introduces the concrete technical and operational challenges and fills critical gaps in the state-of-the-art. This research work is directly relevant to future aviation system safety and addresses the Technology Taxonomy area of Safe All Vehicle Access by developing multi-domain situational awareness and prognostic safety awareness, predictive, and alerting tools. Furthermore, it offers low-cost building blocks that can be used for other real-world applications. The main contributions of the dissertation are as follows:

First, our data collection, storage, and preprocessing pipeline, as well as the model architecture design, allow for the continuous monitoring of the system in real time and the fusion of diverse data sources to identify emergent anomalous behaviors.

Second, we have fused multiple datasets in order to capture a wide variety of factors that may contribute to the go-around occurrence. We have collected features from the dataset directly, but also derived features pertaining to approach stability, in-trail separation, and surface operations according to literature search, theoretical studies, and interviews with subject matter experts (SMEs) such as airport and ATC Tower operational personnel. We also propose a new metric – runway occupancy buffer (ROB) – to better reflect air and surface operations interplay during flight approach procedures.

Third, we design and implement a trajectory-based go-around detection algorithm and demonstrate its use in real-world flight operations. The go-around detection algorithm utilizes four-dimensional flight trajectories and employs multiple criteria based on theoretical and empirical investigation, rather than relying on a single criterion. In addition, the algorithm not only labels the flight in binary responses but also annotates when and where the go-around occurred. Our detection method is applicable to any airport for which the surveillance track data are available.

Fourth, we have uncovered statistical relationships between go-around occurrence and the derived features using a principal component logistic regression (PCLR) model. This technique enables us to overcome the high dimensionality and multi-collinearity of the original data set while preserving the ability to assess the contribution of the original features to go-around occurrence. The contribution analysis is critical to decision-making at a strategic level.

Fifth, we develop novel machine learning (ML) based prediction engines using large stores of historical data, in order to provide reliable, sequential predictions of the probability of occurrence of potentially unsafe conditions and alert those risks to stakeholders. From an application standpoint, the sequential prediction of go-around occurrence could not only help in mitigating risks occurring at the pre-go-around state, but also lessen inherent risks and uncertainty from the disruption of an airport when a go-around is executed. The many prediction models we have experimented with establish a baseline for predictive performance that may be improved upon in future research.

Sixth, we explore different approaches to tackle the class imbalance issue for better training of the predictive models. Our work makes the first effort to leverage GANs to generate synthetic flight anomalous scenarios to augment the minor class for better model training. We empirically show that our GAN-based generated samples improve the model performance. Lessons learned are summarized for future researchers.

Seventh, we provide the path forward to provide real-time predictive analytics. This dissertation work is deployed for practical use in the real-time setting, with application scenarios demonstrated in Chapter 9. This research work has the potential, for the first time, to make detecting, interpreting, and predicting go-arounds a practical way to increase

terminal-area safety and throughput, while taking a comprehensive set of environmental and operational factors into account. We demonstrate a proof-of-concept tool for a near-term capability integrated with the NASA In-Time Aviation Safety Management System (IASMS), allowing pilots and controllers to identify risky situations ahead of time. This accelerates the predictive analytics for safety threats in the real-time arena. The real-time application provides a foundation for designing, developing, and implementing a progression of capabilities that expedites the discovery, prognosis, and mitigation of safety-related threats in transportation systems.

Eighth, we implement an end-to-end pipeline from raw data to support the inference of contributing factors and event occurrence prediction. The building blocks of the pipeline can be directly applied to aviation (or non-aviation) contexts with data of similar structure. It might help reduce system risks in future airspace systems where complexity is expected to be greater – such as the introduction of multiple unmanned and Urban Air Mobility (UAM) aircraft into airport operations.

2. Literature Review

In this chapter, we first clarify some basic concepts that are frequently used in this dissertation, then provides an overview of prior research work on aviation safety, particularly associated with go-arounds, and finally we present several common and powerful machine learning frameworks used in this dissertation.

2.1. Preliminaries

Approach [35]: Begins when the crew initiates changes in aircraft configuration and/or speeds enabling the aircraft to maneuver for the purpose of landing on a particular runway. It ends when the aircraft is in the landing configuration and the crew is dedicated to land on a specific runway. It may also end by the crew initiating a go-around.

Go-around [35]: Begins when either the pilot or the controller aborts the descent to the planned landing runway during the approach phase. It ends after speed and configuration are established at a defined maneuvering altitude or to continue the climb for the purpose of cruise.

Note that another strict definition given by FAA is that go-arounds result in the aircraft returning to the landing queue to attempt the landing once again [25]. We here use the definition given by International Air Transport Association (IATA), and also extend it to touch-and-go, stop-and-go, circling approach, missed approach and break-off approach.

Landing [35]: Begins when the aircraft is in the landing configuration and the crew is dedicated to touch down on a specific runway. It ends when the speed permits the aircraft to be maneuvered by means of taxiing for the purpose of arriving at a parking area. It may also end by the crew initiating a Go-around.

Stabilized approach [36]: A stabilized approach is one in which the pilot establishes and maintains a constant angle glidepath towards a predetermined point on the landing runway. It is based on the pilot's judgement of certain visual clues, and depends on the maintenance of a constant descent airspeed and configuration.

Runway incursion [37]: Any occurrence at an aerodrome involving the incorrect presence of an aircraft, vehicle or person on the protected area of surface designated for the landing and take-off of aircraft.

2.2. Operational Anomalies During the Approach

The mainstream literature related to go-arounds has focused on the behavior and performance of pilots and controllers when a go-around occurs. From the pilot's point of view, Causse et al. (2013) found that the negative emotional consequences attached to the go-around – the uncertainty of a decision outcome and the reward/punishment – can

temporarily jeopardize pilot decision making and cognitive functioning, while Dehais et al. [19] examined the errors in pilot's flying performance (e.g., flightpath deviations) and visual scanning behaviors during go-around execution. From the air traffic controller's point of view, Jou et al. [27] point out that controllers' failure to maintain situational awareness was the leading cause of Taiwan's go-around incidents in 2010. Kennedy et al. [38] found that controller age and expertise have significant impacts on aircraft landing decision making during a flight simulation task.

Another area of study is criteria that should be used by controllers and or pilots for deciding whether to initiate a go-around. The Flight Safety Foundation [26] developed surveys and interviews to identify four groups of factors that were most influential to the decision of a go-around: flight path profile, aircraft configuration, flight energy, and environmental conditions. Campbell et al. [39] developed go-around criteria in terms of airspeed, glideslope deviation, localizer deviation, and rate of descent, at different starting altitudes from which pilots cannot successfully recover from an unstabilized approach on full-flight simulators. They found that the airspeed and localizer deviations impact go-around occurrence the most. To further validate their proposed go-around criteria, Campbell et al. [40] collected objective simulation data and subjective post-simulation written questionnaires from an experiment in which pilots were instructed to go around if the aircraft was outside of the proposed criteria range at 300 feet, or if either pilot was uncomfortable with the approach. The results revealed that the proposed criteria performed well, but some minor adjustments are still needed to make pilots more comfortable with the go-around criteria. Later Zaal et al. [41] evaluated the effects of environmental conditions on the proposed go-around criteria using statistical tests and decision tree analysis. Wind speed, visibility, and localizer deviation substantially affect the go-around decision making and perception of risk. The study suggests that certain environmental conditions might warrant altered decision thresholds of the go-around criteria. While the criteria above suggest some of the factors that influence go-around occurrence, other research has addressed this question through statistical analysis of historical flight data. Shepherd et al. [42] presented an in-trail runway occupancy scenario event tree model that accounts for the risks associated with the go-arounds due to multiple runway occupancy and runway incursions. The go-around execution probability and the go-around failure probability are calculated. Surveillance track data is utilized in more recent research. Sherry et al. [43] detected go-arounds (termed as aborted approaches in the paper) using the cumulated change of aircraft heading angles. The go-around rate of the Chicago O'Hare International airport (ORD) airport was reported as 0.74% with some false positives resulting from procedure turns or other normal maneuvers that meet the quantitative criteria but are not aborted approaches. The go-around detection algorithm based on cumulative turn angle is a valuable contribution, but may not be fully representative. Sherry et al. [43] further reviewed 467 voluntary Aviation Safety Reporting System (ASRS) reports to identify underlying factors leading to aborted approaches. They classified the factors as airplane issues (48%), traffic separation issues (27%), weather (16%), runway issues (5%), and crew-ATC interaction issues (4%). These summary statistics afford valuable insights,

but the ASRS reports are voluntary and are made only when there is a perceived safety issue.

Donavalli [18] identified go-arounds by simply checking whether the approaching aircraft crosses the end of the arrival runway. The go-around detection results were used in two-proportion Z tests to compare the go-around rates for different weather condition scenarios. High wind gust speeds and thunderstorms have a significant impact on go-around occurrence. However, possibly due to the limited variability of the 18-day data set, the Z-test did not find a significant visibility impact. The authors further developed a linear regression model that defines the proportion of daily go-arounds as the dependent variable, different weather factors as the independent variables. However, none of the variables were significant, suggesting that many factors other than weather conditions also contribute to the go-around occurrence. Without incorporating a wide range of situational conditions and environmental measures in the feature space, models would not be appropriate for making any concrete analysis and policy decisions.

Deshmukh et al. [44] identified go-arounds by looking at the violation of two linear regression lines, which are specified for the bounds of normal operations in terms of latitude, longitude, altitude, groundspeed, and aircraft energy. Each flight track is sampled to have 60 timesteps and was truncated into two parts – the first 55-timestep track is known information for deriving aircraft energy and separation features, and the last 5-timestep track is used for go-around detection and provides ground truth labels for the classification task. The paper neither considers the impacts of runway operation and weather impact, nor explains how the classification model could serve their purpose of identifying go-around precursors. In addition, given the truncation timestamp choice, a go-around could easily be initiated during the 55-timestep portion of the track, resulting in artificially high-performance evaluations. It is likely that the prediction performance would decline rapidly if the portion of the track used for performance was reduced.

While previous studies have yielded valuable insights about various causes of and contributors to go-around occurrence, none has developed a comprehensive, quantitative assessment of the relative importance of a wide range of factors that affect the likelihood of a go-around.

There is little work on predicting go-arounds in the literature. Bro [45] investigate the utility of neural networks in predicting go-arounds using 2000 hours of general aviation training flight data. In their work, the features are extracted to create a *snapshot* of an aircraft's parameters at 200 feet above ground level on approach. Figuet et al. [46] predicted the probability of go-arounds for each landing aircraft at a cutoff point located 10 km in front of the runway threshold with supervised learning classifiers. However, a flight approach procedure is a time series sequence. Prediction problems involving sequentially structured data cannot be effectively dealt with by models which only take one snapshot of features in the process.

A predictive model for go-arounds should cover the entire approach and leverage a time series of relevant data to capture the inherent temporal structure of the flight approach. Existing models fall short of this, as they are based on only one snapshot of features in the time series process.

2.3. Machine Learning Frameworks

In this section, we review several common and powerful machine learning frameworks specific to this dissertation – classifiers, Markovian models, recurrent networks, and generative adversarial networks. Each of these frameworks has seen extensive empirical and theoretical applications. It is important to identify when a framework is appropriate for a problem at hand. This dissertation utilizes these disparate machine learning frameworks to address important problems of risk predictive intelligence in the context of go-arounds.

2.3.1. Classical Machine Learning

This class of models are widespread and usually branch out into supervised learning and unsupervised learning. They are simple to use, easy to understand, and usually do not require a large amount of computational power. Supervised learning is the more commonly used and more powerful tool in the real world, if data is labeled. It focuses on two types of tasks: classification that determines the class of an object (e.g., go-around or not), and regression that predicts a continuous value on a numerical axis (e.g., runway occupancy buffer). For instance, K-nearest neighbor, decision tree, SVM and ensemble methods like random forest and boosting have been commonly used to make predictions.

Logistic regression is a technique borrowed by machine learning from the field of statistics. It is named for the function used at the core of the method – logistic function, which also called the sigmoid function in "S" shape to map any real-valued number into a value between 0 and 1. The logistic regression is popular for binary classification problems to calculate or predict the probability of a certain class or binary (yes/no) event.

Unlike the objective of linear regression, SVM minimizes the sum of squared coefficients and tries to find the optimal separating hyperplane in the multidimensional feature space within a threshold error value (margin). A kernel function is usually used to map the data into higher dimensional space and find the hyperplane without increasing the computational cost. In this study, Gaussian Radial Basis Function (RBF) kernel and polynomial kernel [47] are applied to capture potential nonlinearities.

Ensemble learning utilizes the prediction of several base models to improve robustness over a single model. Random forest and extreme gradient boosting (XGBoost) are based on training an ensemble of decision trees, which map the daily feature vectors to the observed daily arrival delay in the leaf node. RF model builds shallow decision trees independently, using a random subset of features, on various subsamples of the dataset. Boosting models sequentially grow decision trees and tries to reduce the bias by learning from previous iterations.

2.3.2. Markovian models

Hidden Markov Models are generative models that can not only be used to learn patterns in sequences, but also to generate new sequences. They were first introduced by Rabiner and Juang [48] as generative models to classify speech signals and have since become a standard method for sequential predictions.

The Hidden Markov Model is a probabilistic graphical model of the representation of the joint probability distribution. A graph of a standard HMM is shown in Figure 4, where nodes represent variables, and edges represent transitions and dependence relationships. Specifically, the white nodes in the figure suggest hidden states that are typically not observed, and the blue nodes represent the visible output emitted by those hidden states. The underlying probabilistic transitions between hidden states are termed transition probabilities and denoted as edges between the white nodes. The transitions between hidden states and output states are called emission probabilities and are represented by the edges between white and blue nodes.

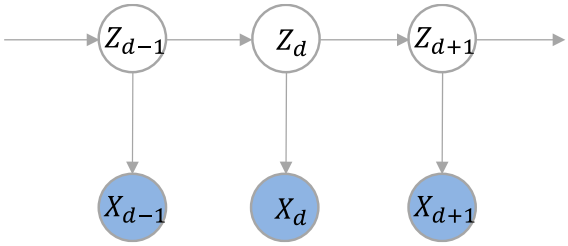

Figure 4. HMM architecture.

For our use case, we employ a variant of the input-output hidden Markov model (IO-HMM), an extension to the HMM that can better capture the sequential structure inherent in our problem to model and predict the go-around occurrence for an approaching flight. The IO-HMM is a special graphical model and was firstly proposed by Bengio and Frasconi [49] for learning problems involving sequentially structured data. While it originated from HMM, it has a much more flexible architecture, which can be interpreted as a statistical model for target propagation [50] based on the Expectation-Maximization (EM) algorithms. HMMs adjust their parameters using unsupervised learning, whereas the IO-HMM uses EM in a supervised fashion. Experiments on artificial tasks [51] have shown that IO-HMM, which uses EM recurrent learning, can deal with time dependencies more effectively than backpropagation through time and other alternative algorithms. The IOHMM is a conditional model that predicts the labels given the features, rather than looking at the joint probability. It can be applied to achieve our goal of fully exploiting both input and output portions of the flight sequence data, as required by the go-around prediction task.

2.3.3. Recurrent networks

Deep learning is a kind of machine learning frameworks that incorporates artificial neural networks with representation learning. With powerful function approximators like neural networks, deep learning has been more popular in recent years. Various applications such as natural language processing and computer vision have offered cutting-edge performance.

Recurrent neural networks (RNNs) are a kind of deep learning models in which the connections between nodes form a directed or undirected graph along a temporal sequence. Using RNNs for sequence modeling and generation has become the norm [52]. The vanilla recurrent neural network is a feed-forward neural network with sequence memories. The parameters are shared across time steps and thus provide two significant advantages: First, there are a lot less parameters to deal with in a RNN compared with a fully connected neural network when it comes to sequence learning. Second, RNNs can handle a flexible length of sequences. We denote the forward propagation of a vanilla RNN in Equation (1) and (2) in functional form, with the input vector x_t, output y_t, and hidden state h_t at the current time step t. In order to produce the next hidden state, h_{t+1}, we need some weight matrices W, bias coefficient b, and a non-linearity activation function.

$$y_t = W_x x_t + W_h h_{t-1} + b \tag{1}$$

$$h_{t+1} = \tanh(y_t) \tag{2}$$

In a RNN, we essentially back-propagation through time (BPTT), going forward through the whole sequence to compute losses L, and going backward through entire sequence to compute gradients, as in Equation (3) and (4):

$$\frac{\delta L}{\delta W_x} = \sum_{t=0}^{T} \frac{\partial L_t}{\partial h_t} x_t \tag{3}$$

$$\frac{\delta L}{\delta W_h} = \sum_{t=0}^{T} \frac{\partial L_t}{\partial h_t} h_{t-1} \tag{4}$$

It is possible for the gradient to “explode” (tend to infinity) or “vanish” (ten to zero) exponentially with respect to the sequence lengths. This is known as the exploding/vanishing gradient problem, which results in substantial inefficiency when training long sequences. With a proven ability to tackle the gradient issue, Long Short-Term Memory (LSTM) has become one of the most used variations of RNNs, and is often used to classify, process and make predictions based on time series data. A variety of transportation-related studies have been undertaken, including work on predicting traffic flow [53], traffic speed [54], individual locations [55], transportation mode [56], pedestrian motion [57], as well as aircraft trajectory [57, 58, 59], and flight delay[60].

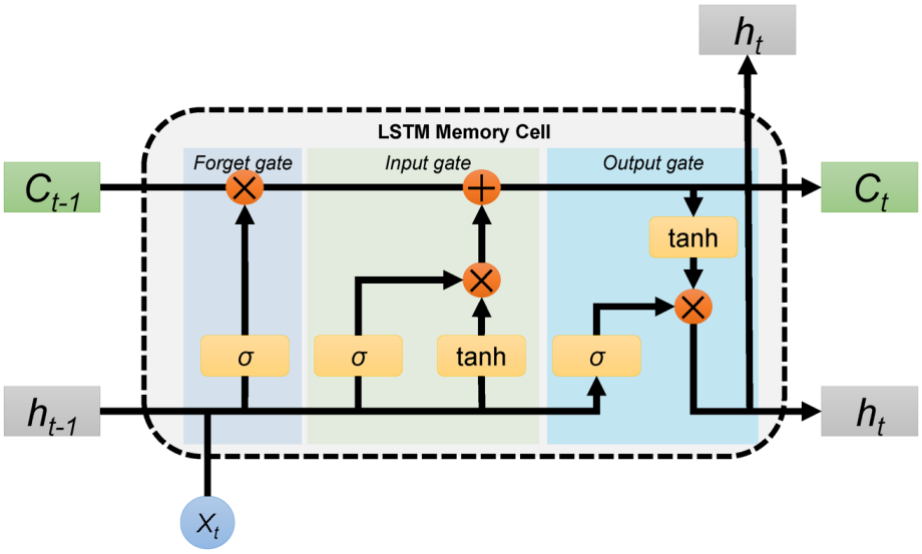

Figure 5. The architecture of an LSTM memory cell [61].

Figure 5 depicts the architecture of a common LSTM memory cell. The specialty of an LSTM memory cell is that it adds an input gate i, an output gate o, and a forget gate f for regulating the information flow into and out of the cell c that remembers values over arbitrary time intervals. The forward pass of an LSTM cell is expressed as Equation (5) – (9) in compact forms, with the operator ∘ denotes the Hadamard product (element-wise product). The three gates are activated by the sigmoid function (σ), while the cell memory and hidden state are activated by the hyperbolic tangent function ($tanh$).

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \tag{5}$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{6}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{7}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{8}$$

$$h_t = o_t \circ \tanh(c_t) \tag{9}$$

2.3.4. Generative adversarial networks

A generative adversarial network (GAN) is a class of machine learning frameworks designed by Ian Goodfellow and his colleagues in 2014. Given a training set, the technique learns to generate new data with the same statistics as the training set. Figure 6 demonstrates the concept of the GAN architecture.

GANs use an adversarial training workflow consisting of a generator and a discriminator, which are deep models like neural networks. The generator learns to map from a latent space (e.g., a multivariate normal distribution) to a data distribution of interest

and produces new data samples. The real data samples and the candidate data samples produced by the generator from the true data distribution, are together fed into the discriminator. The discriminator is trained to classify (or distinguish) each sample as real or fake and the classification errors are used to train the parameters of both the generator and discriminator through backpropagation.

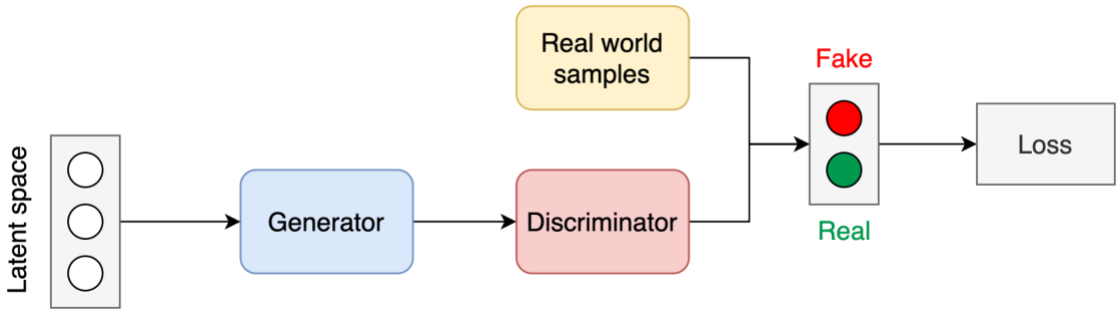

Figure 6. The architecture of a canonical GAN.

The core idea of a GAN is based on the indirect, or adversarial, training of the generator through the discriminator, which itself is also being updated dynamically. In other words, the generator is trained to succeed in "fooling" the discriminator by producing synthetic samples that the discriminator thinks are not synthesized, while at the same time, the discriminator is also trained to increase its ability to distinguish the slight contrast between the genuine (real samples) and the generated (fake samples).

Mode collapse [62] is a well-known problem in GANs where they generate only a few modes of the underlying distribution. It is particularly exacerbated in time series use cases, such as when we generate go-around sequences, because of the high variability in the range of feature values. Some researchers perceive the root problem to be a weak discriminative network that fails to notice the pattern of omission, while others assign blame to a bad choice of objective function.

3. Anomaly Detection

3.1. Go-Around Detection

The raw flight trajectory data does not explicitly label go-around flights that can serve as ground truth for the subsequent analysis. Current practice of go-around detection is mostly based on voluntary self-reports of controllers or pilots, which are typically unreliable and incomplete. Ref. [63] used ASRS data, which is self-reported, voluntary, post-event data, to forecast the trends of go-around causes and counts. It requires manual labeling, and the results fail to provide predictive information before go-around happens. Survey or interview data from flight crew, which could be unrepresentative, was used by Flight Safety Foundation [26] to examine go-around decision-making and the outcome of go-arounds. 3D full-flight simulation data [64] and the training flight data help with the analysis of go-arounds, but the size of data is insufficient.

We did not find an algorithm that rigorously detects go-around occurrence from flight track data in the open literature. Therefore, we propose a scientific way of detecting go-around occurrence by analyzing historical flight trajectories. The trajectory-based detection algorithm can not only label the flight in binary responses (go-around or not), but also identify when (timestamp) and where (3D position) the go-around occurred. Each flight trajectory will be processed and must meet the criteria to be considered as a go-around. For each labeled go-around flight, we treated the timestamp/trackpoint at the start of ascent as the initial time/location for the go-around procedure, and truncated the trajectory after that point.

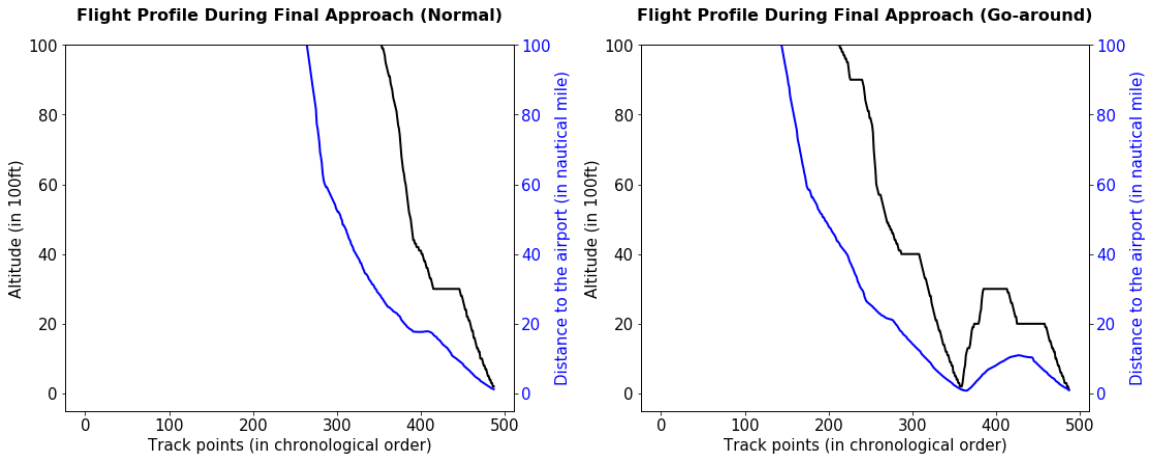

Figure 7. Altitude profile (black) and distance profile (blue) of a normal landing flight (left) and a go-around flight (right).

According to [35], "A go-around begins when the crew aborts the descent to the planned landing runway during the approach phase; it ends after speed and configuration are established at a defined maneuvering altitude or to continue the climb for the purpose of the cruise." Our proposed go-around detection algorithm identifies go-arounds by analyzing historical full flight trajectories. Figure 7 compares the horizontal and vertical profiles between a normal flight approach (left) and a typical go-around flight (right). In these plots, the black curves show the altitude and blue curves show the horizontal distance away from the airport. A typical go-around flight would first decrease its altitude and distance to the landing runway threshold, then climb and fly away from the airport for another approach and landing.

The main block of this algorithm is the change point detection using piecewise linear regression. As discussed above and as shown in Figure 7, the trend shifts in aircraft altitude and the distance to the airport signify go-around events in most cases. Piecewise linear regression models are suited here to capture those changes since the models look for first-order changes, that is, points at which the rate of change differs from one region to another. We first define a buffer size of 5, indicating the number of data points the algorithm will consider to find a change point. Then we compute two residual sum-of-squares (RSS) for a piecewise linear fit, with RSS_1 determines the sum of the RSS for the first five data points and the RSS for the subsequent five data points, while RSS_2 determines the RSS for the combined set of these ten data points. The two values (RSS_1 and RSS_2) are used to calculate the F-statistic associated with the two fits. The F-statistic is used to evaluate if one line or two lines would better represent the analyzed region of data, i.e., whether a change point exists for this region. The detailed trajectory-based go-around detection algorithm is presented in Table 1.

The trajectory-based go-around detection algorithm is designed to detect any aborted landing behaviors – including missed approaches and go-arounds – that are initiated within 10 nm of the airport, regardless of whether the flight overflies the runway, or whether it proceeds to the Missed Approach Points (MAPt). The definition of go-around in this study applies to both VFR aircraft and IFR aircraft, as long as the aircraft tries again for the landing.

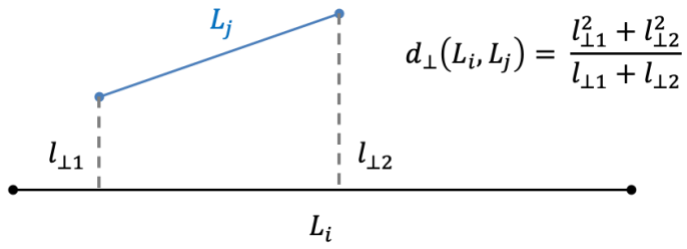

Figure 8. Perpendicular distance [65] between the extended runway centerline L_i and the trajectory segment L_j.

Table 1. Go-around detection algorithm.

Algorithm: Go-around Detection Algorithm

INPUT: Four-dimensional flight track data (latitude, longitude, altitude, and time)

INITIALIZE: Coordinates of arrival runway thresholds

OUTPUT: Go-around labels and their related properties (execution time, execution altitude, etc.)

Procedure

Step 1: Data preprocessing. Apply median filtering with a sliding window size of 10 to remove noise from the trajectory records. Remove incomplete trajectories (the altitude of the last track point is higher than 500 feet) and define landing endpoint (the rate of descent equals 0 feet) for complete trajectories.

Step 2: Altitude check. Piecewise linear regression is applied to identify points at which the slope of the altitude evolution curve, the black curve shown in Figure 7, is changed. Each flight trajectory is processed and must meet the following criteria to be considered as a go-around:

- The altitude at the start of ascent is no more than h_{start} (default value of 5500 feet);
- The total altitude gain during the ascent must not be less than Δh (default value of 400 feet).

Step 3: Define the analyzed segment. For flights that pass the altitude checks in Step 2, the landing endpoint (defined in Step 1) is updated to the point at which the altitude starts increasing. Each aircraft's analyzed segment is a two-minute (T_{final}) trajectory segment ending at the landing endpoint.

Step 4: Calculate the landing runway. We did not directly use the landing runway information recorded in the given dataset due to a significant number of missing and incorrect records. Instead, we calculate the landing runway using the two-minute analyzed segment extracted from Step 3. For every trajectory segment formed by two track points, we calculate the perpendicular distance [65] from the extended centerlines of all the available arrival runways (by configuration) using formulas in Figure 8. Each track segment thus votes for the closest Extended Runway Centerline (ERC) segment. The landing runway is the one that receives the most votes from track points in the vector. This runway is also used as one of the features in the statistical model.

Step 5: Distance check. For each track point of the two-minute analyzed flight trajectory segment, calculate the distance to the runway threshold markings of the corresponding landing runway obtained in Step 4. Piecewise linear regression is applied to identify points at which the slope of the curve representing distance to landing runway threshold, the blue curve is shown in Figure 7, is changed. Each flight trajectory is processed and must meet the following criteria to be considered as a go-around:

- When a go-around flight is within 1-nautical-mile range of the airport, its altitude does not exceed h_{1nm} (default value of 1500 feet);
- Go-around must occur within the 10-nautical-mile range of the airport, in order to distinguish go-arounds from aircraft being vectored or in holding patterns;
- The ascending segment of a go-around trajectory must be within 10 nm of the center of the airport.

Step 6: Multi-go-arounds. The two consecutive go-around procedures should be separated by at least 5 minutes (T_{multi}). The trajectory starting point for the second and subsequent flight trajectory segments is when the previous go-around trajectory segment starts ascending.

end procedure

We developed our algorithm based on consultations from subject matter experts and sensitivity analysis to determine parameters such as the altitude at the start of ascent h_{start} and the total altitude gain during the ascent Δh. We further applied it to the JFK arrival flights except for military flights, general aviation, and helicopters. Figure 9 illustrates the JFK airport chart and the runway threshold of runway 22L as an example. We collected the coordinate information of the midpoint of all the runway threshold bars at JFK airport (marked as yellow for runway 22L in Figure 9), and applied it as the input to the go-around detection algorithm described in Table 1. In this dissertation, the "runway threshold" specifically refers to the midpoint of the runway threshold bar.

Figure 9. JFK airport chart (left) and the runway threshold marking (right).

Two of the detected go-around flight trajectories are visualized in Figure 10. The flight in the left plot aborted the descent to the planned runway 22R, overflew the runway, returned to the landing queue, and finally landed on runway 22L. The flight in the right plot was planned to land on runway 4L, but it ended up landing on runway 22L (Note that this flight is counted as a go-around and its subsequent successful landing is not considered). In total, 433 go-arounds have been detected from July 1st to December 24th in 2018, which accounts for 0.43% of all JFK arrivals within the period. This statistic agrees with the FAA report, which indicates that the average percentage of go-around occurrence across the core 30 airports in the U.S. from 2012 to 2017 is 0.4% [25].

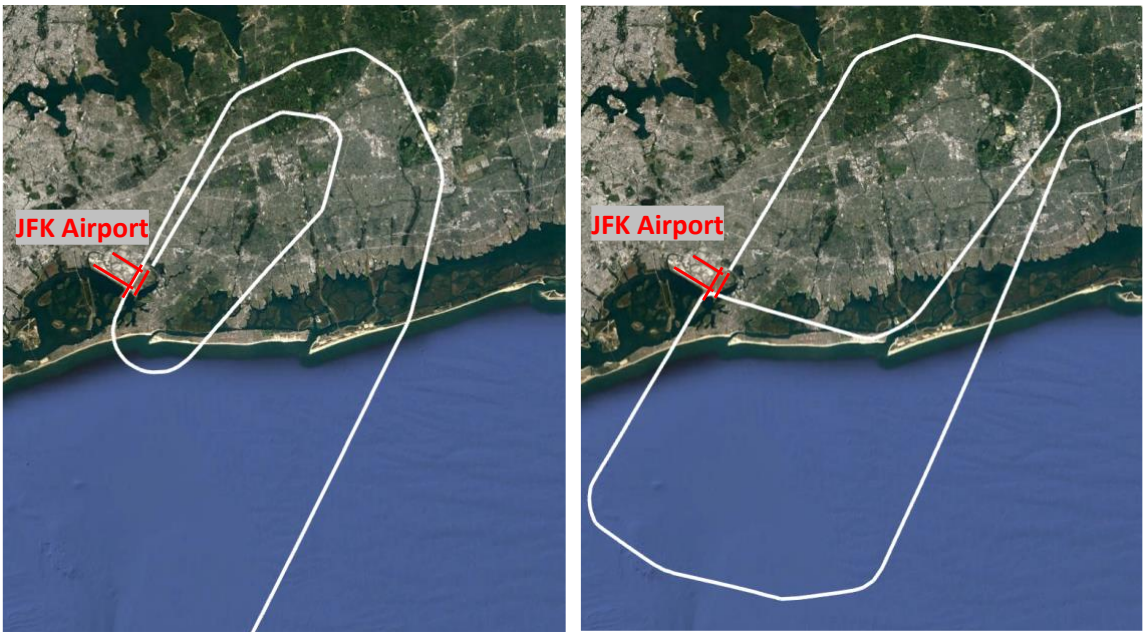

Figure 10. Exampled go-around flight trajectories.

3.2. Validation

The other effort in this chapter involved validating actual go-around occurrences that would serve as ground truth or label data for the models. For validation purposes, we apply our detection algorithm on the historical trajectory data to identify a rapid reversal in descent within 10 nm of the airport. We then compare the go-around detection results (i.e., labels generated for historical flights) with the Sherlock go-around reports for the New York Terminal Radar Approach Control facility (New York TRACON, also known as N90). The N90 report is based on business rules developed by the FAA in the Performance Data Analysis and Reporting System (an entirely independent method).

Both methods initially exhibited a significant number of false positives due to general aviation aircraft operating in the vicinity of JFK. A filter was created to identify and remove go-around events involving GA aircraft. We classified jet aircraft and multi-engine turboprop aircraft as commercial aircraft. Single-engine turboprops, all piston-engine aircraft, and any helicopters, gliders, and tiltrotors were classified as GA aircraft. The filter was applied to the labels produced by both methods of go-around detection.

Next, the go-around events from each methodology were cross-referenced by comparing the aircraft identifier and the go-around event timestamp, allowing for up to five minutes of difference in time between the methods. Then, through applying filters to the Sherlock go-around N90 reports to exclude general aviation VFR flights and to enhance our detection algorithm to handle operational situations like diversions, we were able to get the methodologies to converge on a labelled data set for 2018 operations at JFK.

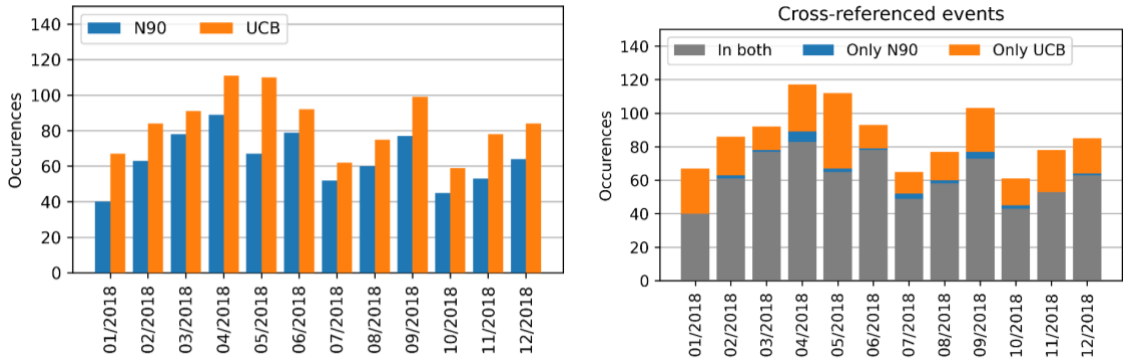

Figure 11. Month-by-month cross-referenced results of the Sherlock go-around N90 reports (blue) and our go-around detection algorithm (orange).

Figure 11 shows the cross-referenced results of the Sherlock go-around N90 reports (in blue) and our go-around detection algorithm results (in orange) by month. In total, our go-around detection algorithm identifies 1,012 go-around occurrences (including multi-go-around events), while the N90 report detects 767 go-around events. Nearly all of the go-around events identified in the N90 report, 743 of them, were also detected by our algorithm. We thus visually inspect the few occurrences in which the two methodologies disagree. Among the 24 N90-only events, 17 of them are false positives, and we add the 7 true-positive events to update our labeled set. Among the 269 events detected by our algorithm only, 255 of them are true positives, and we remove the 8 false-positive events from our labeled set. There were still 6 occurrences detected by our algorithm but did not appear in the N90 reports. We thought these to be questionable events and hence omitted them from the labeled set (i.e., still in the flight data set for analysis, but labeled as non-go-around flights). Figure 12 shows two of the questionable events, with the left aircraft NCA159 descends to 400 feet and then climbs, but does not seem to be aligned with any runway. On the right, the aircraft DAL405 descends and aligns with runway 31L though there is most likely a data error.

In general, our algorithm is able to identify more true positive events because it can capture go-arounds initiated farther away from the runway threshold, and with more robust criteria. Additional work is required to refine this detection algorithm and to consult with SMEs to increase the annotation quality.

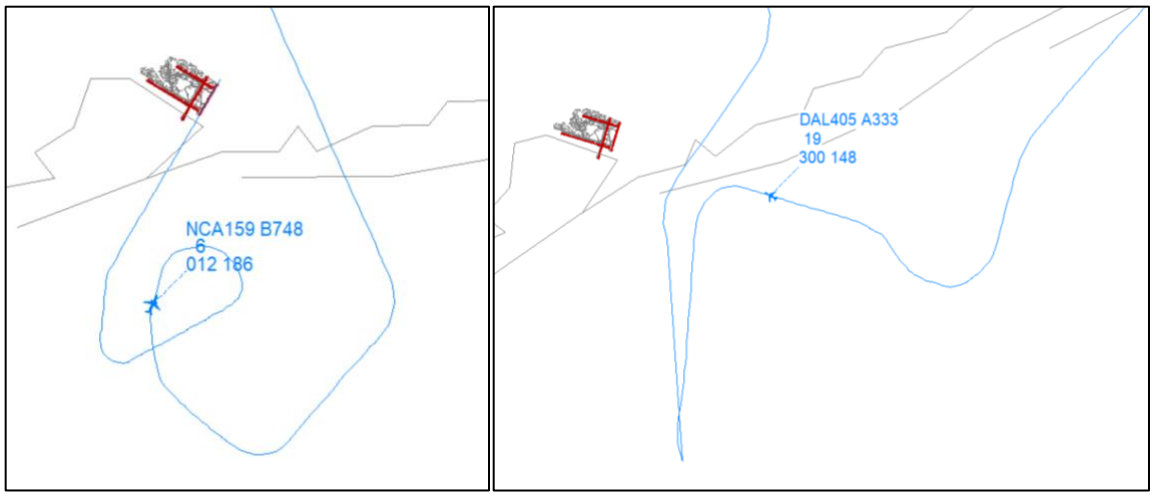

Figure 12. Questionable go-around labeled events.

4. Feature Engineering

4.1. Overview

The goal of this chapter is to determine the features that may affect go-around occurrence and how we transform them from raw data to provide useful inputs for the following statistical analysis and machine learning applications. A feature is an attribute or property of the system being studied (in this case the air traffic on approach to an airport) on which analysis or prediction is to be done. Through literature search, theoretical studies, interviews with subject matter experts (SMEs) such as airport and Air Traffic Control Tower operational personnel, as well as the data availability, we developed seven categories of features including aircraft characteristics, approach stability features, in-trail separation features, weather, airport conditions, go-around clustering effect, and surface operation features.

In this chapter, we will first introduce a concept, called information cutoff gate, to demonstrate why and how we subsample the flight sequences for the subsequent studies. We then summarize the process of feature extraction and derivation in categories. Among all the features derived, we propose a new metric, called runway occupancy buffer, to measure the surface operation conditions. We will go into more details for this feature in Chapter 5.

4.2. Information Cutoff Gate

In the raw data, the flight trajectory data are time series of aircraft position recorded every few seconds (roughly every 5 seconds), which constitutes a dense representation of the trajectories in the terminal airspace. Presumably, each track point in the dataset could be counted as one observation since the features vary. Processing such dense time series observations in machine learning algorithms could result in overfit and increase the training time.

In order to reduce the computational complexity while maintaining an informative representation of trajectories, we apply linear extrapolation to subsample the flight trajectories at every nautical mile away from the landing runways, and only kept the trajectories within ten nautical miles of any the landing runway thresholds at the analyzed airport (Flights beginning apparent go-arounds outside the 10 nm range will not be considered). We hereafter define those points at every nautical mile away from the landing runway thresholds as ***information cutoff gates***, as shown in Figure 13. We will only derive features and annotate go-around labels at those gates based on the information available when the subject flight passes a certain information cutoff gate. In other words, we will model go-around probabilities based on information available when the aircraft reaches these gates, approximately every 15 – 30 seconds during the approach phase.

Another reason of identifying the information cutoff gate is to make sure that we only consider those features that can be evaluated at a certain time prior to when the go-around occurs – there is no point in predicting go-arounds after it already happens. The extrapolation technique ensures that the sequential prediction will be only based on the information available when the subject flight passes a certain information cutoff gate. Note that this distance-driven sampling strategy also guarantees that time-varying features are comparable for all the flights in the data set. Taking "5nm gate" as an example, (1) only go-arounds detected within [0, 5) nm from the landing runway threshold will be considered; (2) flights initiating go-arounds when they are more than 5nm from the threshold are not considered; (3) we do not include any features in modeling that cannot be known when the flight passes the 5 nm arc.

As a result, each flight approach procedure can be represented by a sequence with a fixed number of information cutoff gates – located at d = 10, 9, …, 1 nm from the runway threshold. Some of the flight sequences may have a length of fewer gates if the aircraft initiated a go-around or otherwise aborted the approach prior to concluding the approach procedure. For these flights, we consider only those features that can be evaluated before the initiation of a go-around. The available features thus depend on the distance of the go-around initiation point from the runway. In this study, we assume a go-around is executed when its altitude starts increasing during the approach procedure based on our detection algorithm in Chapter 3.

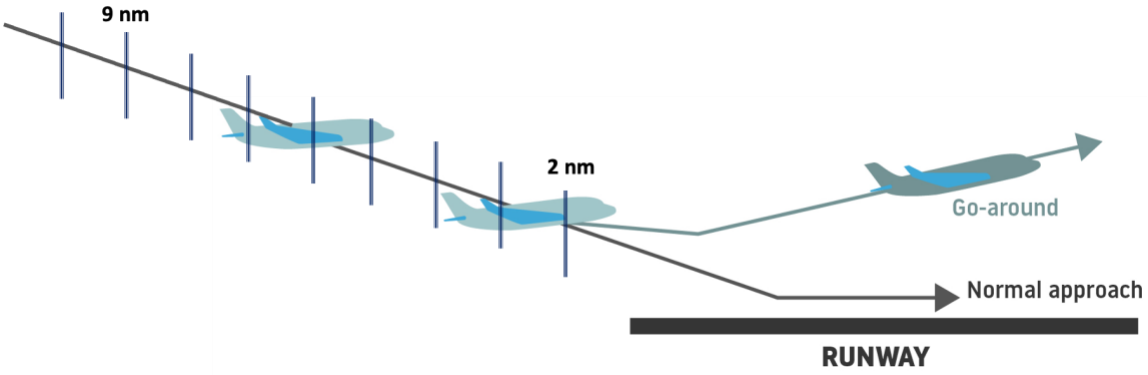

Figure 13. Information cutoff gates as shown as the vertical lines on the virtual approach path on the left.

4.3. Features

We conducted several interviews with subject matter experts and NASA to understand more fully the potential factors that should be included in the go-around prediction models from an operational standpoint. Considering the data availability, we derive the following seven categories of features that we would use as inputs to the statistical model and predictive models. These include: aircraft characteristics, approach stability features, in-

trail separation features, weather features, airport conditions, and surface operation features. The following subsections summarize the process of feature extraction and derivation from the raw data. The variable code and variable description can be found in Table 3 at the end of this chapter. Though the space is limited to report the summary statistics of all the features at all information cutoff gates, we include the summary statistics for the 5 nm information cutoff gate in Table 3 as an illustration.

Using data described in Appendix, observations are obtained for every flight operating during the study period with seven categories of features. After preprocessing and matching with the ASPM dataset, the final output has up to ten distance-varying feature vectors for each flight observation, one at each information cutoff gate. Each vector represents one distance-stamped record. We will feed the time series feature matrix into the predictive model presented in Chapter 7, 8, and 9, while using only one static feature matrix (at the 5 nm information cutoff gate) for statistical modeling in Chapter 6.

4.3.1. Aircraft and Runway Characteristics

We categories airlines as international and domestic carriers, because we expect that pilots working for domestic (U.S.) airlines are more experienced with landing in JFK airport. Aircraft type is considered by categorizing as narrow body and wide body, in order to capture differences in separation requirements and handling characteristics. We also identify the landing runway to capture different approach patterns. Specifically, one-hot encoding, which creates binary columns to indicate the presence of each possible value from the original categorical feature, is applied to create dummy variables for wide-body aircraft (*Body*), international airliner (*Airline*), and the calculated landing runway (*Runway*).

4.3.2. Approach Stability

As the [36] emphasized, "If not stabilized, go around". Continuation of an unstabilized approach to land may result in an aircraft arriving at the runway threshold too high, too fast, out of alignment with the runway centerline, incorrectly configured, or otherwise unable to land safely. Accordingly, we derive altitude deviation (*AltDev*), groundspeed (*Speed*), angle with the extended runway centerline (*Angle*), and Kinetic energy height (*Energy*) as flight approach stability features to capture potential instability indicators that may prompt a go-around.

Normally the optimum vertical profile to use during a landing approach is a 3-degree glideslope path [66], which requires that the aircraft descend at about 300 feet per nautical mile. A large deviation from the target descent rate indicates an unstable approach. Thus, we calculate the altitude deviation from the standard 3-degree glideslope path (*AltDev*) to capture the potential unstabilized approach risk:

$$AltDev_i^d = \left| h_i^d - 6076.12 \cdot d \cdot \tan(3°) \right| \tag{10}$$

where h_i^d is aircraft i's altitude in feet at the d nm information cutoff gate, 1 nm = 6076.12 feet.

When the aircraft i is at the d nm information cutoff gate, its groundspeed (*Speed*) after median filtering is extrapolated to capture the situation in which an aircraft approaches too fast or too slow. We applied the median filtering as a preprocessing step to remove out-of-range isolated noise in the trajectory data. The median filter is a smoothing technique, which runs through the data entry by entry, replacing each entry with the median of neighboring entries. The angle of horizontal deviation from the extended runway centerline (*Angle*) is also calculated using the flight latitude and longitude information to measure the misalignment of the standard approach path. As shown in Figure 14, the solid blue line is the ERC of runway 31R. When a flight, represented as red dot, intercepts the distance arc (e.g., 5nm), we record and calculate the following features at this moment: *Altitude (alt)* (in 100 feet), *Groundspeed (speed)* (in knots), and *Perpendicular distance to ERC (horiz)* (in nautical miles).

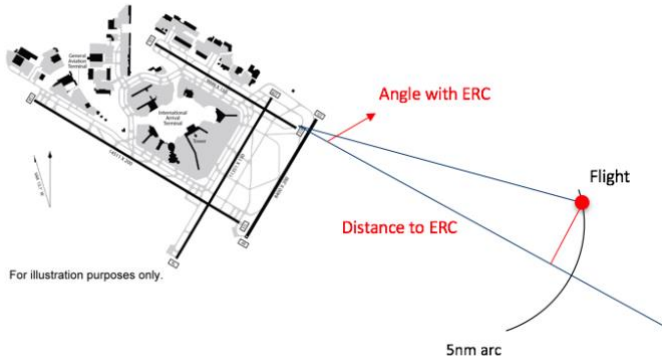

Figure 14. Diagram of trajectory performance features.

Aircraft energy management is of great importance during the approach procedure to maintain safety. An aircraft's energy state is the sum of potential energy and kinetic energy per unit weight [67]. However, the calculation requires information on aircraft mass, which depends on payload and fuel load data that are not available to the researchers. Thus, we use the energy height metric that can be defined as the hypothetical height [68], H_i, at which the aircraft i's potential energy ($m_i g H_i$) is equal to the total energy at its current state ($m_i g h_i + \frac{1}{2} m_i v_i^2$), which is calculated as:

$$H_i^d = h_i^d + \frac{(v_i^d)^2}{2g} \tag{11}$$

where H_i^d is the kinetic energy height to be calculated at the d nm information cutoff gate (*Energy*, in feet), h_i^d and v_i^d are respectively the aircraft altitude and aircraft groundspeed when the flight i is at the d nm information cutoff gate, and g is the constant of gravitational acceleration. This metric can be calculated for each flight during the approach process to represent the aircraft energy-related risks using only the surveillance track data.

4.3.3. In-trail Separation

Separation is defined as the distance, either horizontal or vertical, between two aircraft. Here we are interested in the distance between a given aircraft on approach the lead aircraft (if any) landing on the same runway. The minimum required horizontal separation in this situation depends on the relative weight class of two aircraft and meteorological conditions (visual or instrument). A defined Loss of Separation (LOS) between airborne aircraft occurs whenever the specified separation minima in controlled airspace are breached. In this study, the loss of separation is calculated as the difference between the minimum required separation from the FAA standard and the actual separation between the lead-trail aircraft pair. We expect that a greater loss of separation (in nautical miles) increases the probability of go-around. Therefore, to capture the separation effect, we derive four variables that are employed in the statistical models: *Loss of separation (LOS)*, a dummy variable *NoLead* indicating the case where there was no leading aircraft for a given flight, the speed difference *(SpeedDiff)* and the altitude difference *(AltDiff)* between leading and trailing (subject) flight.

The algorithm for obtaining these variables requires three steps – finding leading and trailing aircraft pair, obtaining actual separations, speed difference, and altitude difference for the lead-trail aircraft pair, and finally calculating the loss of separation. We elaborate on these steps below:

i. Group flights with the same (calculated) landing runway obtained from the go-around detection algorithm in Table 1, and sort them in chronological order based on the time that flights cross the runway threshold. For each group, we create a list of tuples where each tuple contains two consecutive aircraft that have been sorted. Within each tuple, if the runway threshold crossing time difference of the two aircraft is smaller than 10 minutes, then we define them as a lead-trail aircraft pair. Otherwise, we set a dummy variable *NoLead* to 1 for the trailing (subject) aircraft to indicate the case in which, for all practical purposes, there was no leading aircraft for a flight.

ii. For each trailing flight, we find the linearly extrapolated timestamp t at which the trailing (subject) flight is *d* nm to its landing runway. At the extrapolated timestamp t, we again extrapolate the locations (latitude, longitude, altitude) and groundspeed of both leading and trailing aircraft. The separation between these two extrapolated locations (in terms of latitude and longitude) is noted as S_t. We also calculate the speed difference *(SpeedDiff)* and altitude difference *(AltDiff)* between leading and trailing (subject) flight at this extrapolated timestamp t. To be specific, we subtract the extrapolated groundspeed/altitude of the leading aircraft from the extrapolated

groundspeed/altitude of the trailing (subject) aircraft when the trailing flight is at the *d* nm information cutoff gate.

iii. Obtain the separation minima from FAA Wake Separation Standards [69] based on the weight class of leading and trailing aircraft under VMC (S_m^{VMC}) and IMC (S_m^{IMC}). When the trailing flight is at 5 nm to its landing runway, if the meteorological condition is recorded as "VMC" in the ASPM quarter-hour dataset, the standard separation minima is $S_m = S_m^{VMC}$ (e.g., 1.9 nm for the Large-Large lead-trail pairs), otherwise $S_m = S_m^{IMC}$ (e.g., 3.0 nm for the Large-Large lead-trail pairs). Thus, the loss of separation *(LOS)* is $S_l = \max(0, S_m - S_t)$, and is directly employed as a continuous variable in the model.

4.3.4. Weather

We expected that runway configuration change, arrival traffic, airport capacity, visibility, ceiling, and wind condition could also trigger a go-around. To capture the expected non-linearity of impacts of various visual conditions on go-around occurrence, the visibility variable (in statute miles) is discretized into three continuous subsections: [0, 1], (1, 3], (3, 5], (5, 10]. Similarly, the ceiling variable (in 100 feet) is discretized into four continuous subsections: [0, 5], (5, 10], (10, 30], (30, 100]. The intervals are based on the criteria set for defining low IFR, IFR, marginal VFR, and VFR [70]. For example, if the recorded ceiling equals 600 feet, the discretized ceiling variables Ceiling_k are 5, 1, 0, 0.

The ASPM airport quarter-hour dataset provides surface wind speed (in knots), wind angle (in degrees), and arrival runway configuration. For each landing aircraft, we apply trigonometric calculations to compute the headwind/tailwind speed and crosswind speed with the information of landing runway configuration at the airport. For the variable wind in which the wind angle was not available, we set the headwind/tailwind speed and the crosswind speed as $\sqrt{2}/4 \times$ wind speed. When the wind is a headwind, the tailwind is set to zero, and vice versa.

4.3.5. Airport Conditions

We subtract the arrival/departure rate (counts) from the arrival/departure demand (counts) to capture the airport traffic conditions. A negative sign in these demand-minus-capacity variables indicates the absence of an arrival queue. The Airport Arrival/Departure Rate (AAR/ADR) and the number of intended landing/departing aircraft (demand) are obtained directly from the ASPM dataset on a quarter-hourly basis. For a given flight, the change of runway configuration variable (*RwyChange*) is set to 1 if the used runway configuration during the observed period is different from the preceding 15-minute period, and 0 otherwise. As an additional indicator of operational traffic, we include daytime dummy variables if the observed aircraft reaches d nm information cutoff gate between 6

am and 6 pm in local time (*Daytime*). A dummy variable is also created to indicate the Instrument Meteorological Conditions (*IMC*), as opposed to Visual Meteorological Conditions (*VMC*).

4.3.6. Go-Around Clustering Effect

From the go-around detection results, we observed that go-arounds sometimes occur in clusters—that is several occur in a short time interval. To capture this effect, we calculate the time difference between when a given flight is at the d nm information cutoff gate and the initiation time of the latest go-around that occurred in the past 24 hours. This minimum time difference (*GaGap*) among all other go-arounds in record is used as a temporal clustering feature. If no go-arounds occurred in the past 24 hours for a given flight, we set the *GaGap* to 1440 (minutes). The *GaGap* variable only focuses on the effect from the previous go-around flight and such an effect weakens with time. As a second clustering feature, we include the number of go-arounds (except the given flight if it was a go-around) that occurred in the past 30 minutes in JFK airport when a given flight *i* was at d nm from the runway threshold (*GaCnt*). This variable measures the clustering effect in terms of quantity.

4.3.7. Runway Incursion Risk

In the case where pilots or controllers anticipate a runway incursion, a common practice would be to initiate a go-around [37]. Therefore, we have derived two variables – predicted Runway Occupancy Buffer ($\widehat{ROB}$) and counts of objects (both aircraft and vehicles) on the runway *(RwyCnt)* - to serve as indicators of incursion risk and used them as features in our go-around model.

One of the incursion variables is the number of aircraft or ground vehicles on the runway *(RwyCnt)* when the subject flight is d nm from its landing runway threshold. We first define the Runway Safety Area (RSA) polygon bounded by holding position markings painted on the taxiway or runway surface [71, 72]. When the subject aircraft reaches the d nm information cutoff gate, we count the total number of arrivals, departures, and crossing aircraft/vehicles that are contained in the corresponding landing RSA polygon at that moment, using ASDE-X surface track data.

The *ROB* is defined as the time difference between the runway threshold crossing time of the trailing aircraft and the runway exit time of the leading aircraft. When a trailing aircraft reaches a certain information cutoff gate, the $\widehat{ROB}$ is predicted using algorithms given in the next chapter. It captures the variations in the runway threshold interarrival time [73], landing runway occupancy time [74], and the spacing buffers routinely applied by air traffic controllers [75]. Note that we incorporate the predicted $\widehat{ROB}$ at each information cutoff gate as one of the feature inputs for the following statistical modeling (Chapter 6) and predictive analytics (Chapter 7), instead of the observed ROB. This is to reflect the fact that the information available to controllers and pilots are limited at the time the aircraft crosses that gate. The decision of go-arounds is largely depending on human anticipation

of the surface status and runway incursion risk. Such nested configuration also allows us to explore how this predicted feature can improve model results. In Chapter 5, we will get into details on how to obtain the feature, predicted runway occupancy buffer $\widehat{ROB}$. For flights that do not have leading aircraft, this value is set 0 seconds, but with a dummy variable, *NoLead* added as described above.

4.4. Feature Types

In this section, we further discuss the features and divide them into two types – attributes and time series features. This serves as preliminary for Chapter 7 and Chapter 8 of this thesis, as different data/feature types require different handlings when we develop the learning models (e.g., IO-HMM, and GAN).

The attributes are static features that will not change during the final approach, such as aircraft type, weight class, operated airline, and landing runways. The time series features are dynamic, as they will vary along with the approach. Examples include flight altitude and ground speed. Table 2 shows an example of a go-around flight sequence in the dataset, where rows corresponding to nautical mile distance timestamp, and columns describing features at each timestamp.

Table 2. An example flight sequence data.

Timestamp	**Distance to threshold (nm)**	**Weight class**	**Operated airline**	**Altitude (100 feet)**	**Groundspeed (knot)**	…
1	10	Large	UA	26.52	128.07	…
2	9	Large	UA	24.00	118.06	…
3	8	Large	UA	21.99	130.41	…
4	7	Large	UA	19.83	119.17	…
5	6	Large	UA	17.00	76.29	…
6	5	Large	UA	17.00	103.04	…
7	4	Large	UA	14.00	95.85	…
8	3	Large	UA	11.00	118.18	…
9	2	Large	UA	7.64	123.00	…
10	1	Large	UA	4.00	125.29	…

Table 3. Model variables and summary statistics.

Variable Category	Variable Code	Variable Description	When flight *i* is at 5 nm from the threshold		
			Mean	**Min**	**Max**
(I) Aircraft and Runway Characteristics	Airline^{+}	1 if flight *i* is operated by an international airline, 0 otherwise	0.21	0	1
	Body^{+}	1 if flight *i* is wide-body aircraft, 0 otherwise	0.24	0	1
	Runway^{+}	Dummy variable for calculated landing runway of flight *i*	-	0	1
	Daytime^{+}	1 if the observed time is between 6 am and 6 pm in local time, 0 otherwise	0.61	0	1
(II) Approach Stability	Angle	Angle with the Extended Runway Centerline (in degree)	7.99	0.00	68.45
	AltDev	Absolute altitude deviation from 3-degree glideslope (in feet)	151.92	0.00	719.12
	Speed	Flight groundspeed (in knots)	163.19	77.28	277.85
	Energy	Kinetic energy height (in feet)	2841.45	1318.72	5370.49
(III) In-trail Separation	LOS	The loss of separation between leading and the trailing flight *i* (in nautical miles)	0.09	0.00	2.61
	SpeedDiff	Groundspeed difference between leading and the trailing flight *i* (in knots)	20.92	-86.95	144.73
	AltDiff	The altitude difference between leading and trailing flight *i* (in 100 feet)	13.30	0.50	22.28
	NoLead^{+}	1 if there is no leading aircraft in front of flight *i* within 10-minute landing sequence, 0 otherwise	0.13	0	1
(IV) Weather	Wind	Wind speed where the headwind component is subtracted (in knots)	5.72	0	25.98

Variable Category	Variable Code	Variable Description	When flight i is at 5 nm from the threshold		
			Mean	Min	Max
	$Visibility_k$	Discretized visibility (k = 1,2,3; intervals are [0, 1], (1, 3], (3, 5] and (5, 10] in miles)	-	0	10
	$Ceiling_k$	Discretized ceiling (k =1, 2, 3, 4; intervals are [0, 5], (5, 10], (10, 30], (30, 100] (in 100 feet)	-	2	100
(V) Airport Conditions	ArrQue	Difference between airport supplied arrival rate (AAR) and the number of intended landing aircraft (counts)	-2.04	-15	31
	DepQue	Difference between airport supplied departure rate (ADR) and the number of intended depart aircraft (counts)	1.24	-15	52
	RwyChange+	1 if the used runway configuration is changed from the previous quarter hour, otherwise 0	0.08	0	1
	IMC+	1 for IMC, 0 for VMC	0.15	0	1
(VI) Go-around Clustering Effect Features	GaGap	The minimal time interval between the approaching time of flight i and the initiation time of the latest go-around occurred in the past 24 hours (in minutes)	712.62	5.37	1440
	GaCnt	The number of go-arounds occurred in the past 30 minutes	0.06	0	5
(VII)Surface Operations	$\widehat{ROB}$	Predicted runway occupancy buffer (in seconds)	30.60	-28.89	146.79
	RwyCnt	The number of aircraft and vehicles appearing on the landing runway (counts)	2.04	0	9

+ Variables are one-hot encoded.

5. Runway Occupancy Buffer

5.1. Overview

To enable the safe and efficient integration of NextGen, the FAA Administrator's Strategic Priorities is moving to risk-based decision-making and places great emphasis on finding an optimal compromise between runway safety and efficiency. As the air traffic demand grows, how to maximize the runway throughput without compromising runway safety levels becomes more important to Air Traffic Management and Control (ATM/ATC) strategies. Toward this end, it would be useful for controllers and pilots to have real-time predictions of when arriving flights will cross the runway threshold relative to the exit/departure of the previous aircraft from the runway. Such predictions could support a decision-support tool to assist in adjusting the arrival time to increase or reduce this time difference as appropriate.

In this study, we introduce a new metric - Runway Occupancy Buffer (ROB) - to unmask the interaction between air and surface operation during flight approach procedures. For an in-trail arrival aircraft pair, the leading aircraft must clear the runway before the trailing aircraft crosses the runway threshold to prevent Simultaneous Runway Occupancy (SRO). The difference between these two timestamps is what we call the ROB. During busy periods, ROBs ideally should be small in order to maximize runway efficiency. In this case, when the leading aircraft exits the runway, the trailing aircraft will be about to cross the runway threshold. Otherwise, large ROB indicates inefficient runway use and may cause unnecessary delays. Conversely, ROBs that are too small (or negative) create a risk of incursions, accidents, or incidents.

ATC would benefit from the ability to predict ROB so that actions might be taken to achieve a safe and efficient value for this metric. Also, as explained in Chapter 4, the predicted ROB is a useful feature for predicting go-arounds. Therefore, we focus on applying machine learning techniques to better understand and predict the ROB metric, which can lead to an improvement in runway safety and runway throughput. Our motivation is three-fold. First, from a safety point of view, modeling and predicting ROBs would feed as a predictive tool at the airport to alert air traffic controllers and flight crew about runway operational risks and impending aircraft behaviors. Controllers can have longer reaction times to handle unsafe situations. If the predicted ROB is much longer or shorter than expected, controllers and pilots may coordinate to adjust the aircraft speed, altitude, heading or execute a go-around. Second, from an efficiency perspective, the prediction of ROB can enable the pilot to adjust speed for achieving the desired spacing ROB, thus improving runway capacity. With longer prediction horizons, the predictability is not only beneficial to airport performance in the terminal area but also improve performance in the downstream and upstream. Third, as to uncertainty, the accurate prediction of ROBs may permit a narrowing the safety margins applied by air traffic controllers. Due to the uncertainty of actual operations, human decision-making errors still

exist. Reliable ROB predictions could boost controllers' and pilots' confidence during the decision-making process. In sum, the capability to predict ROBs with small uncertainty could result in a smaller buffer time that ultimately increases capacity without compromising safety levels.

5.2. Related Work

Predictability is recognized as an important operational performance goal for ATM. Liu et al. [76] investigates the potential benefit of the predictability in airport surface operation system from the controllers' perspective, flight operator's perspective, and traffic management perspective. They conclude that the predictability on the airfield and surface reduces the controller's workload surges, has the potential to better deal with off-nominal situations, improves performance in the downstream and queue area.

Most of the research on runway safety and efficiency has focused on analyzing operations on the surface and in the air separately. For the analysis of runway operations, a number of advanced analytical methods have been proposed to predict Runway Occupancy Time (ROT), which is the amount of time that a runway is occupied, or not usable by another aircraft. Meijers et al. [77] uses the Random Forest (RF) model to compute the feature importance of ROT for 36 major US airports based on ASDE-X radar tracks. It was found that the runway exit, the aircraft type, the airline, the final approach speed of the landing aircraft and the presence of the following aircraft in approach explain over 80% of the variance of ROT. Although some predictors identified in this paper, such as the runway exit and the exit angle, cannot be used for real-time prediction, it still provides useful insights into the factors that contribute to ROT variability. Herrema et al. [78] investigates the identification and prediction of abnormal runway occupancy times only. Lasso, Multi-layer perceptron and neural networks are used to predict taxi-out time, time to fly and true airspeed profile on the final approach. Martinez et al. [79] presents a boosting tree framework to predict the actual ROTs and the expected exit at different distances from the runway threshold.

For the analysis of approach procedures in the air, Tosic and Horonjeff [73] estimated the landing runway capacity by computing the runway threshold interarrival time. It is assumed that the system is free of errors, and the runway occupancy time is always less than that threshold interarrival time. Substantial literature can also be found in analyzing so-called *remaining time to arrival* (RTA), which is defined as the time difference between the present time and the Estimated Time of Arrival (ETA). A neural network approach [80] was applied to study sources of variability in flight arrival times and achieved $\pm$4.5min accuracy at a 95% level for the 10nm range. Levy and Bedada [81] present the results for the real-time estimation of ETA at the runway threshold. The paper concludes that the predictive accuracy of $\pm$3.5 min accuracy at the 90% level should be adequate to improve gate management.

Airport safety and efficiency can be optimized by improving the prediction capabilities of ROT, flight arrival times and their uncertainty. Nikoleris and Hansen [82] point out that the benefits of precise runway arrival times are greatly reduced when ROTs are highly variable. We need to understand and model the relationship between precision in meeting RTAs at the runway threshold and variability in ROTs systematically. Studies only concern predictability in either of the two might be less beneficial due to the combination of errors in both models. They thus consider queueing models to understand and model the relationship between system throughput, precision in meeting RTAs at the runway threshold, and ROTs systematically [82]. Stochastic variations in the time for leading aircraft to clear the runway may delay trailing aircraft's threshold crossing time if their schedule separation does not include any excess time to absorb such variations. Conversely, the prediction of the time for the trailing aircraft to reach the arrival runway threshold may influence the runway exiting behavior of the leading aircraft.

With limited work to date on integrated modeling air and surface operation, our work in this section aims to fill that gap by using historical trajectory data to model runway safety, landing throughput, trajectory prediction and variability in ROT systematically. We first rigorously define the ROB and provide the empirical analysis. The ROB not only considers the runway threshold interarrival time proposed by Tosic and Horonjeff [73], but also considers the landing occupancy time proposed by Simpson, Odoni and Salas-Roche [83]. We then directly model the ROBs using Linear Regression (LR) models and Random Forest (RF) regression model. Lastly, the performance of all candidate models is investigated and compared. Model interpretation and feature importance are also discussed.

5.3. Empirical Analysis

According to the ATC safety requirement that no more than one aircraft can occupy the runway at any time—i.e., no simultaneous runway occupancy (SRO), we define the Runway Occupancy Buffer (ROB) to capture the interaction between ROT variability of the leading aircraft and the predictive RTA of the trailing aircraft. In this section, we precisely define ROB and provide more practical insights into this metric. We then illustrate the algorithm for recognizing leading-trailing aircraft pairs and calculating ROB from the ASDE-X surface track data. Lastly, summary statistics of ROB are presented and discussed.

5.3.1. Definition

The Runway Occupancy Buffer (ROB) ΔT, in seconds, is the time difference between the runway threshold (abbr. *thd*) crossing time of the trailing aircraft $t_{trail,thd}$ and the runway exit time of the leading aircrafts $t_{lead,exit}$, as shown in formula (1):

$$\Delta T = t_{trail,thd} - t_{lead,exit} \quad (12)$$

Note that the runway exit standard we use in this study is more rigorous than the standard used in identifying SRO. To make sure that no part of the fuselage or the tail of an aircraft may infringe on the runway, Runway Safety Area (RSA) is defined and bounded by holding position markings painted on the taxiway or runway surface. Generally, an aircraft exiting a runway is not clear of the runway until all parts of the aircraft have crossed the applicable holding position markings [84]. In line with this purpose, we create the RSA polygon by collecting the coordinates of all the holding position markings for all the runways in the JFK airport – 04L/22R, 04R/22L, 13L/31R, 13R/31L. Only the track points that are contained in the analyzed RSA polygon will be analyzed. Figure 15 shows the one-day traffic in blue on the abstracted 04L/22R RSA polygon.

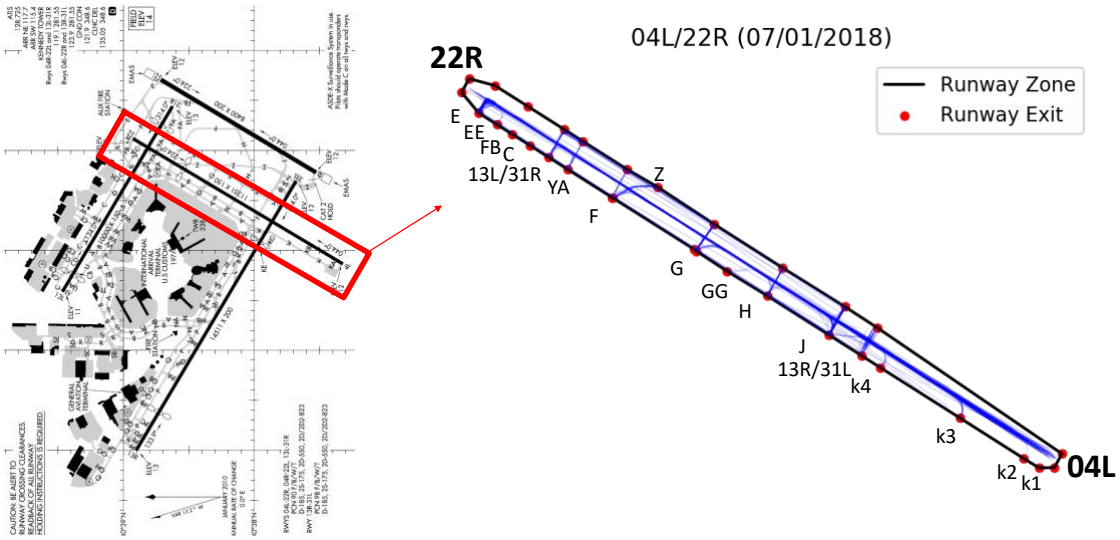

Figure 15. One-day traffic on the abstracted 04L/22R RSA polygon (07/02/2018).

We visualize two pairs of in-trail arrivals (three aircraft land on the same runway) in Figure 16 to illustrate related variables. In this figure, three colored blocks represent three consecutive flights that arrive in-trail. They are arranged in chronological order from the left to the right as the arrow is pointing. The left edge of the colored block i is the runway threshold crossing time $t_{thd,i}$, while the right edge represents the runway exit time $t_{exit,i}$. Thus, a single block represents the time duration that the flight occupies the landing runway, in essence, Runway Occupancy Time. As the formula (13) defines,

$$\Delta T_1 = t_{thd,2} - t_{exit,1};\ \Delta T_2 = t_{thd,3} - t_{exit,2} \tag{13}$$

The spacing between every two colored blocks is the ROB for the trailing aircraft. If two blocks overlap, ΔT is negative. It indicates a lack of separation in time that could mean a runway incursion. When the buffer time is large, as ΔT_2, it indicates a loss of runway throughput which, when the airport is busy, could reduce airport throughput and increase delay.

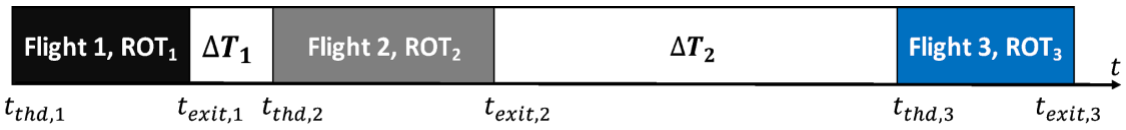

Figure 16. Concept display of Runway Occupancy Buffer.

If $\Delta T = 0$, then when the leading aircraft exits the holding position markings, the trailing aircraft is just about to cross the runway threshold. This theoretical ideal requires precise spacing of the arrivals which results in maximum runway throughput without violating the SRO prohibition.

5.3.2. Calculation

We use the algorithm described as follows to calculate the RTA, ROT, and ROB from ASDE-X dataset for each in-trail arrival. Go-around flights are detected first using the algorithm in Chapter 3 and removed from the dataset. Military, general aviation, and helicopter flights are also excluded from this study since they have very different approach patterns from commercial flights.

The calculation algorithm has two steps – finding leading and trailing aircraft pairs and calculating the RTA, ROT, and ROB. The extrapolation strategy guarantees that the RTA, ROT, and ROB are calculated using the same standard, thus all these measurements are comparable for all the flight observations. We emphasize that the ROB is neither the runway threshold interarrival time [73], not the landing occupancy time [83]. The calculation of ROB only depends on the flight track data – the timestamps at which the subject (trailing) flight crosses the runway thresholds, and the timestamp at which its leading flight exits the runway holding position marking.

5.3.3. Observed statistics

We only analyze in-trail arrivals identified by the algorithm described in Table 4. Flights not having leading aircraft, and flights whose leading aircraft have already exited the runway will not be considered for the ROB prediction. After data cleaning and matching, there are, on average, 338 in-trail arrival flights each day in the analyzed airport within the analysis period. Thus, we compile a dataset including 56,731 observations of ROTs and ROBs for aircraft operating at JFK airport.

The ROT ranges from 45 seconds to 117 seconds. The distribution of ROT is consistent with those reported in the literature using Automated Surface Observing System (ASOS) data archives [77] and Advanced Surface Movement Guidance & Control System (A-SMGCS) data [86]. The average value of the observed ROB is 85 seconds, which indicates that, on average, the trailing aircraft crosses the runway threshold around 85 seconds after the leading aircraft exits the runway. Nonetheless, individual observations vary widely, with a standard deviation of 79 seconds. The maximum observed ROB is 569 seconds, which relates to the criteria we set in the calculation algorithm (Step 4 in Table 4) to filter

out aircraft pairs with runway threshold crossing time differences over 600 seconds. In addition, the magnitude of ROB is indicative of the traffic conditions in the analyzed airport. As seen in Figure 17, ROB is small in the afternoon and evening when there is heavy arrival traffic. ROB is generally higher when the majority of morning traffic is composed of departures, or there is little traffic after midnight.

Table 4. ROB, RTA, ROT calculation algorithm.

Algorithm: ROB, RTA, ROT Calculation

INPUT: IFF flight track data, ASDE-X flight track data, RD summary

INITIALIZE: Coordinates of arrival runway thresholds and holding position markings in the analyzed airport.

OUTPUT: In-trail aircraft pairs and flight's RTA, ROT, ROB

Procedure

Step 1: Data querying. Query the track point data for each flight in the RD summary.

Step 2: Extrapolated runway threshold crossing time. For each flight, we extrapolate [latitude, longitude, time] to acquire the timestamp at which the subject (trailing) flight crosses the runway threshold $\boldsymbol{t_{thd}}$.

Step 3: Extrapolated runway exit time. Construct the spatial K-D tree [85] from the holding position markings coordinates (red dots in Figure 15) in line with aircraft heading. For spatial continuity, we also incorporate the last flight trackpoint in the KD tree search space. The K-D tree will be used to query the runway exit for each arrival. The timestamp at which the flight crosses the runway exit is extrapolated $\boldsymbol{t_{exit}}$.

Step 4: In-trail relationship. Group flights with the same (calculated) landing runway obtained from Chapter 3 and sort them in chronological order by the time $\boldsymbol{t_{thd}}$ that flight intercepts the runway threshold. For each group, we create a list of tuples where each tuple contains two consecutive aircraft that have been sorted. We designate each tuple as a leading-trailing aircraft pair if:

- the runway threshold crossing time difference of the two aircraft is smaller than 10 minutes $t_{thd,trail} - t_{thd,lead} < 600\ (sec)$
- the leading aircraft has not exited the runway $t < t_{exit}$

Otherwise, we remove the trailing flight from the tuple.

Step 5: Arithmetic. For each trailing flight filtered from the last step, RTA, ROT, and ROB can be calculated at time t when the analyzed flight passes a certain information cutoff gate during the approach procedure:

$$T_{RTA} = t_{thd} - t \tag{14}$$

$$T_{ROT} = t_{exit} - t_{thd} \tag{15}$$

$$\Delta T = t_{trail,thd} - t_{lead,exit} \tag{16}$$

end procedure

Table 5. Summary statistics of the observed ROB.

Distance to runway threshold (nm)	Observations	ROB (seconds)			
		mean	*std.*	*min*	*max*
10	56,731	85.225	79.055		569.194
9	52,283	69.176	51.474		555.832
8	49,790	61.784	34.410		535.856
7	47,924	58.821	30.838		456.775
6	44,902	54.988	27.303	-35.490	176.637
5	39,745	49.781	23.513		146.791
4	31,368	42.596	19.089		137.712
3	17,044	32.644	15.047		93.605
2	2,045	18.633	11.975		57.760

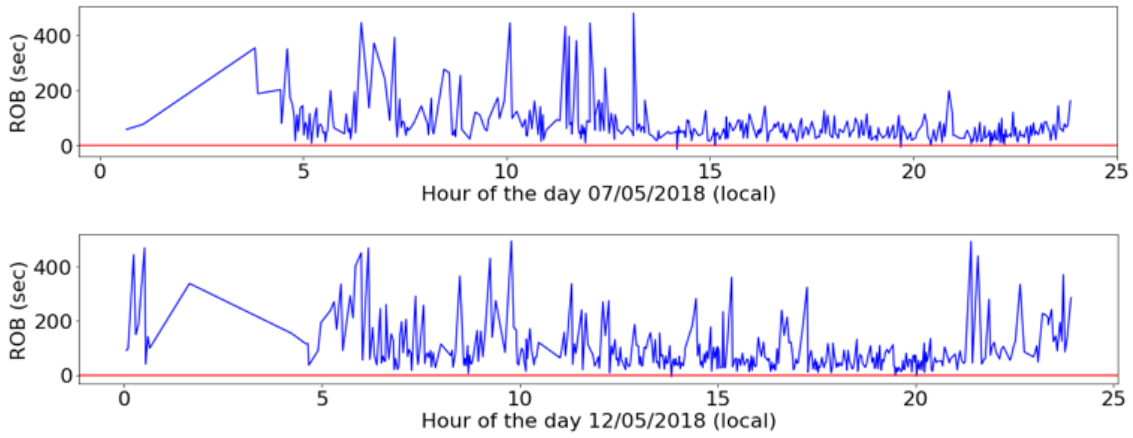

Figure 17. One-day ROB temporal pattern (07/05/2018 at top and 12/05/2018 at the bottom).

In Figure 18, we investigate in detail the in-trail arrival scenario with the minimum ROB of -35 seconds. Both aircraft land on the 04R runway threshold shown at the right end, and then take the high-speed runway exit FB, depicted by the red dots on the left end. The yellow pins extrapolated from the dataset represent the aircraft positions. In the top figure, when the trailing flight crosses the runway threshold, the leading aircraft has already been taxing on FB but has not yet satisfied the runway exit standard of crossing the holding position markings. After about 35 seconds, the leading aircraft (green arrow) crosses the

holding position markings, and the trailing flight (yellow arrow) is nearby, as shown in the bottom figure. Note that such situations result in negative ROBs, but do not necessarily imply a runway incursion.

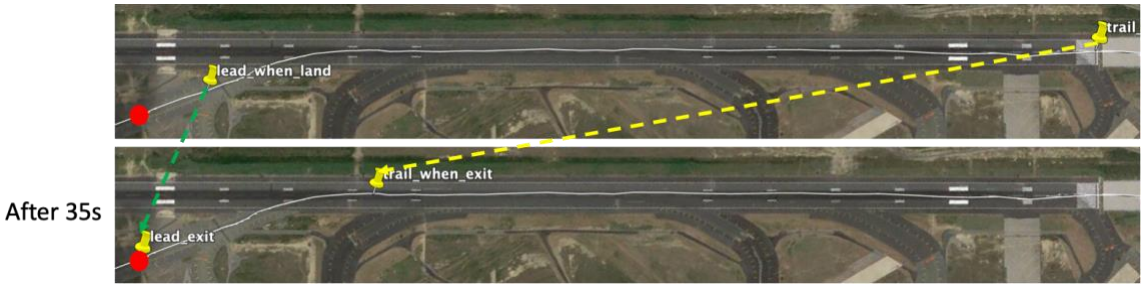

Figure 18. The minimal ROB scenario visualization.

5.4. Runway Occupancy Buffer Prediction

In this section, we would like to develop predictive models based upon the above empirical observations of runway occupancy buffer (ROB) extracted from the real-world dataset. Regression-based machine learning algorithms are employed to learn the relationship between the observed ROB and the relevant potential factors. As the subject flight (trailing aircraft) approaches the airport, we seek to predict its runway occupancy buffer time relative to the leading aircraft using the same set of features described in Chapter 4. It is as if pilots and air traffic controllers anticipate runway conditions and assess whether the approach environments are safe for the trailing aircraft to land.

As pointed out by Nikoleris and Hansen [82], the interdependency between RTAs (air operations) and ROT (ground operations) likely exists, and studies only concern predictability in either of the two might be less beneficial due to the combination of errors in both models. A longer ROT of the leading aircraft will result in a delayed arrival time for the trailing aircraft. Nonetheless, the converse relationship may also hold. It is conceivable for the RTA of the trailing aircraft to affect the ROT of the leading aircraft. For example, controllers will attempt to shorten the ROT of the leading aircraft if the trailing aircraft landing is imminent. This provides the impetus to model and predict the ROB explicitly with predictive algorithms to capture the interaction between RTA (air operations) and ROT (ground operations).

We primarily investigate two types of machine learning algorithms for the ROB model development: linear regression (ordinary least squares, Ridge, Lasso, Elastic Net) and Random Forest (RF). Linear regression is carried out to estimate the ROB with the following Equation (17):

$$y = X\boldsymbol{\beta} \tag{17}$$

where y is the observed ROB, X is the design matrix including variables in Table 3. $\boldsymbol{\beta}$ is the variable coefficient vector obtained by minimizing the sum of the squares of the residuals:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \beta_0 - X\boldsymbol{\beta})^2 \tag{18}$$

To avoid the risk of overfitting and improve the prediction performance, we use regularized regression to reduce the model complexity. A set of regularized linear regressions are investigated: LASSO regression, Ridge regression, and Elastic Net regression. Regularization addresses concerns about variance-bias tradeoff, multicollinearity, sparse data handling, feature selection, and the interpretability of the output.

In ridge regression, the cost function is penalized by the square of the coefficient magnitude $\|\beta\|_2$:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \beta_0 - X\boldsymbol{\beta})^2 + \lambda_r\|\boldsymbol{\beta}\|_2^2 \tag{19}$$

where the penalty term λ_r regularizes the coefficients vector $\boldsymbol{\beta}$. If the coefficients take large values, the optimization function is penalized and the coefficients are shrunk in ridge regression.

In LASSO regression, $l_1$norm, $\|\boldsymbol{\beta}\|_1$, is used as the penalty, which accounts for the absolute value of the coefficients magnitude and may result in coefficients with zero value. Thus, Lasso regression helps in reducing overfitting and feature selection. The cost function is denoted by the following Equation (20):

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \beta_0 - X\boldsymbol{\beta})^2 + \lambda_l\|\boldsymbol{\beta}\|_1 \tag{20}$$

The Elastic Net regression is a hybrid of LASSO and Ridge regression, where the cost function linearly combines the l_1 and l_2 penalties, as indicated in Equation (21). The OLS regression is a special case of ridge regression and LASSO regression as the λ gets close to zero.

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \beta_0 - X\boldsymbol{\beta})^2 + \lambda_r\|\boldsymbol{\beta}\|_2^2 + \lambda_l\|\boldsymbol{\beta}\|_1 \quad (21)$$

Besides linear models, we train random forest regression model to learn a set of decision trees that map all the features to the realized ROT in the leaves of trees by splitting the dataset recursively [87]. The random forest model builds shallow decision trees independently, using a random subset of features, on various subsamples of the dataset. The ensemble of decision trees is built to capture possible nonlinearities between the ROB and explanatory variables described in Table 3, and is not expanding the feature space and yet prevent overfitting.

The performance of six models will be investigated and compared: naïve average model, Ordinary Least Squared (OLS) model, Least Absolute Shrinkage and Selection Operator (LASSO) regression, ridge regression, elastic net regression, and random forest regression. We use the same feature set (except for the ROB) at each information cutoff gate as described in Chapter 4. The same flight is used to generate 10 observations corresponding to 10 distance-stamps. However, observations in each distance dataset, reported in Table 5, are analyzed independently. Since we remove the in-trail aircraft pairs if the leading aircraft has already exited the runway, the number of observations at different distances from the runway threshold will vary as the trailing aircraft approaches the airport, as shown in Table 5. Data is split into 80% of the training set for hyperparameter(s) tuning and 20% of the testing set for model evaluation. A five-fold cross-validation grid search is performed to minimize the Mean Squared Error (MSE) for selecting optimal combinations of hyperparameters defining the model. We then fit models with selected parameters on the entire training set (without folds) and obtain predictions of the test set.

5.5. Results

5.5.1. Model Performance

When a trailing aircraft reaches the information cutoff gate, we employ the trained model on the acquired feature matrix to provide the prediction of the ROB value directly. Six models – *naïve average method, OLS regression, LASSO regression, Ridge regression, Elastic Net regression, random forest regression* – are tested and evaluated on the same dataset for any particular distance segments. We also include a baseline of two-stage model which predicts the ROB by modeling the ROT of the leading aircraft and the RTA of the trailing aircraft separately. The performance metrics for model comparison, RMSE, are summarized in Table 6 and visualized in Figure 19.

Predictability progressively increases as the aircraft approaches its landing runway threshold. It is likely because the information provided is insufficient, or not informative enough, to make a robust prediction of ROB when the aircraft is further away. The two-stage model is superior to the Naïve average method, but inferior to other models under the

integrated modeling scheme. Models developed under the integrated framework perform much better than the two-stage model by capturing the interdependence between RTA and ROT more precisely.

Table 6. RMSE of ROB predictive models.

Dist. (nm)	Two-stage modeling (seconds)	Integrated modeling (seconds)					
		Naïve	*Ridge*	*LASSO*	*Elastic Net*	*OLS*	*RF*
2	11.0	12.0	11.2	11.2	11.2	11.2	8.8
3	14.4	15.1	12.6	12.6	12.6	12.6	9.2
4	18.0	19.1	13.2	13.2	13.3	13.2	9.6
5	19.1	23.5	13.6	13.6	13.8	13.6	9.9
6	20.1	27.3	14.0	14.0	14.2	14.0	10.1
7	22.8	30.8	15.1	15.1	15.4	15.1	11.0
8	23.2	34.4	16.6	16.6	17.0	16.6	12.0
9	33.4	51.5	29.2	29.2	29.8	29.2	15.7
10	51.0	79.1	43.6	43.6	45.3	43.6	26.7

The random forest model (magenta line) has the smallest RMSE on the test set for all models of different distance segments, outperforming linear models, the two-stage model, and the naïve average method. The random forest regression model predicts ROBs with an R-squared fit of more than 90%. The performance of all linear models does not vary appreciably. In general, the OLS model (brown line) seems to predict ROB slightly better, especially when the aircraft is further away from the runway threshold. As the number of observations increases, OLS is more likely to converge to the optimal solution and learn the distribution of ROB better. When the aircraft is close to the airport, the RMSE scores for regularized linear models are slightly higher than those for OLS.

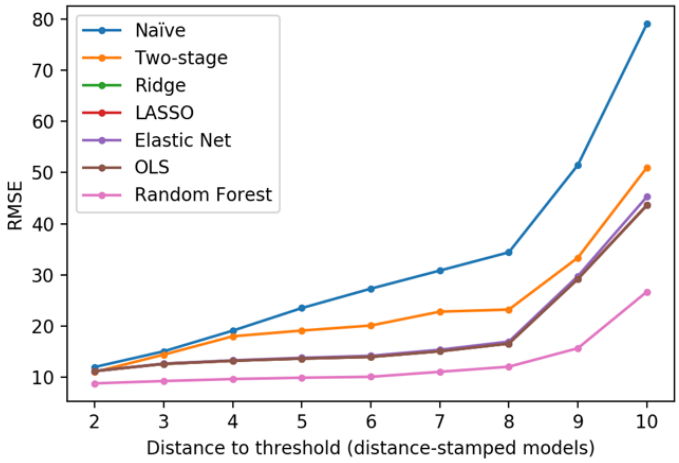

Figure 19. ROB model performance.

5.5.2. Estimation result

To gain interpretability of the model estimates, we summarize the coefficient estimates of the OLS regression model (integrated modeling) at the 5 nm information cutoff gate, in Table 7, and illustrate the feature importance of the random forest model (integrated modeling) at the 5 nm information cutoff gate in Figure 20.

Table 7. OLS estimation results (5 nm information cutoff gate).

Variables	Coef. (Std. error)	Variables	Coef. (Std. error)
Constant	18.853*** (1.977)	**Rwy_13R**	4.866 (3.866)
Speed	-0.325*** (0.009)	**Rwy_22L**	-3.894*** (0.664)
Energy	0.003*** (0.000)	**Rwy_22R**	-1.259 (1.146)
Horiz	-2.041*** (0.483)	**Rwy_31L**	-1.100 (0.790)
Alt_dev	0.561*** (0.065)	**Rwy_31R**	-1.778** (0.672)
Separation	21.820*** (0.299)	**WC_H**	-9.705*** (0.278)
Lead_alt	-2.228*** (0.087)	**WC_L**	-6.145*** (0.255)
Speed_diff	0.011* (0.006)	**WC_S**	2.818* (1.139)
Head	0.369*** (0.015)	**WC_lead_ H**	2.014*** (0.293)
Tail	-0.303*** (0.072)	**WC_lead_ L**	5.312*** (0.243)
Cross	-0.012 (0.016)	**WC_lead_ S**	2.659* (1.053)
Var	0.374* (0.182)	**Terminal 1**	-1.457** (0.496)
Arr_que	-0.064*** (0.014)	**Terminal 2**	5.302*** (0.397)
Dep_que	-0.068*** (0.009)	**Terminal 4**	4.105*** (0.400)
Visible	-0.147*** (0.043)	**Terminal 5**	4.539*** (0.403)
Ceiling	0.000 (0.001)	**Terminal 7**	0.470 (0.472)
Rwy_04R	-2.288*** (0.664)	**Terminal 8**	2.181*** (0.413)
Rwy_13L	10.336*** (1.732)	**Night**	3.543*** (0.137)
MC_V	0.014 (0.258)	**Rwy_change**	-0.288 (0.228)
Adjusted R-squared: 0.681			

Variables are significant at the 0.1% level***, 1% level**, 5% level*

From Table 7, most of the estimates are significant and fit nicely with our understanding of ROB. We first notice that the ROB will increase when the separation between the two aircraft is large. Separation indicates the relative positions of the leading the trailing aircraft. If two aircraft are further away from each other in the air, their arrivals will probably have a large time gap. The altitude of leading aircraft is also an indicator of aircraft relative positions from the vertical view. The higher the leading aircraft is, the closer it is to the trailing, and the shorter the ROB. The trailing aircraft's speed and its distance to the extended centerline have a negative effect on ROB. Results imply a higher ROB if trail aircraft is above the glideslope and a smaller ROB if it is below the glideslope. Pilot actions required to reduce altitude in order to get back on the glide slope slow down the aircraft approach. Tailwind speed has a similar negative effect on ROB as it accelerates the landing.

For the traffic conditions, longer arrival or departure queues will decrease the ROB. ROB is introduced based on flight in-trail relationships, thus depends on continuous traffic demand. If there is less traffic in the airspace, such as midnight, ROB will be large. When the arriving traffic volume exceeds the runway capacity, air traffic controllers will try to squeeze traffic and reduce ROB to improve throughput and some aircraft have to be delayed in the air. The ROBs on 13L and 13R are larger than other runways. Assuming aircraft's runway threshold crossing time are the same, aircraft exit 13L and 13R earlier and thus have less ROT than other runways.

5.5.3. Feature importance

Other than model prediction and interpretation, we are also interested in knowing which features are most predictive of ROB. We calculate feature importance after the model is fit on the whole training set with fine-tuned hyper-parameters. Literature commonly uses the Gini importance to calculate the average value of the number of splits that include the feature (across all trees), proportionally to the number of samples it splits. The Gini importance measures how effective the feature is at reducing variance when creating decision trees within RF. However, our dataset is a mixture of numerical variables and categorical variables. The Gini importance is biased, in the sense that it tends to inflate the importance of continuous variables and high-cardinality categorical variables [88]. To obtain an accurate picture of feature importance, permutation importance [89] is performed to directly measure feature importance by observing the effect on R-squared of randomly shuffling each variable. To be more specific, we first record a baseline R-squared by fitting a validation set through the RF. We then permute the column values of a variable and fit RF on this new permuted dataset. The difference between the baseline R-squared and the permuted R-squared is the importance of the feature of interest. Although the permutation importance is much more computationally expensive than the Gini importance, the feature importance measurements are more reliable.

Feature importance for the more accurate model, RF regression, is plotted in Figure 20. These importance values will not sum up to one since the values represent the difference in R-squared scores between baseline model and permuted model. The x-axis values can be interpreted as relative predictive strengths of features. The separation and the altitude of

leading aircraft are the two strongest predictors of ROB, followed by the speed of the subject (trailing) aircraft, headwind speed, the speed difference between leading and trailing aircraft, and kinetic energy. The permutation importance places the airline operated terminal dummy variables and landing runway dummy variables as less important features, implying that there is not much difference in ROB if the subject flight lands on another runway or uses a different terminal.

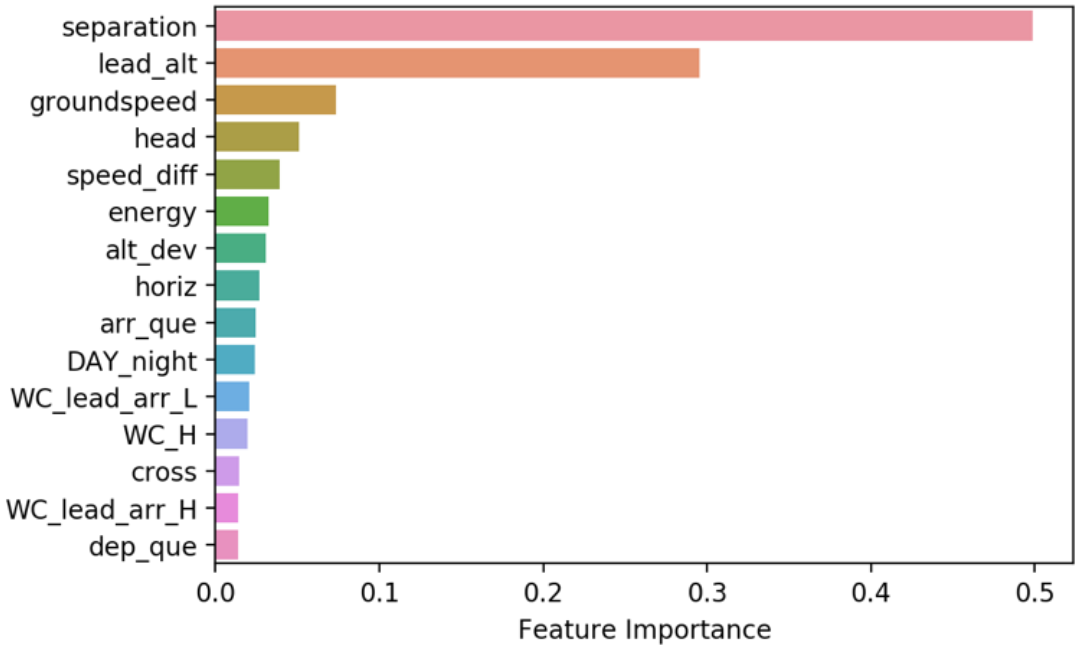

Figure 20. Feature importance for RF regression model.

Both the OLS estimation results and RF permutation importance suggest that separation has the greatest importance in predicting ROBs, followed by the leading aircraft altitude and trailing aircraft speed. Such prediction work could be used to help improve runway safety and efficiency in real time. If the predicted ROB is much longer or shorter than desired, controllers and pilots may coordinate to adjust the aircraft speed, altitude, heading or execute a go-around to reach the targeted ROB level.

In summary, the proposed metric, ROB, functioned as expected. Its interactions with other operational and environmental variables make sense. ATC would benefit from the ability to predict ROB so that actions might be taken to achieve a safe and efficient value for this metric – an immediate use is to predict go-arounds. When we implement the real-time predictive capabilities for go-around prediction (Chapter 9), we also include the real-time representation of the predicted ROB metric. Besides looking at the probability of go-arounds, pilots and controllers may also refer to this metric for guidance on aircraft speed control and traffic management during the final approach and landing procedures. The current work can be enhanced in several ways. For example, sequential learning algorithms

could be applied to better model and predict the ROB metric; The analysis could be extended to a broader scope of operational scenarios, concurrently used for both departures and arrivals; and model generalization could be investigated by applying the trained model to other airports.

6. Modeling Go-Arounds Using Principal Component Logistic Regression

6.1. Overview

In this chapter, we investigate how the derived features in Chapter 4 impact go-around occurrence and quantify their contributions. The motivation of this chapter is threefold. First of all, this study can provide flight crews, air traffic controllers, and other decision makers with better knowledge of the conditions in which a go-around is more likely to be executed. Second, quantifying the contributing factors of go-around occurrence can help identify countermeasures to reduce go-arounds, and more generally the conditions that give rise to them, which may be considered anomalous states that are inherently undesirable. Mitigation strategies can be developed to reduce the go-around occurrences through procedure modification, pilot training, and equipment design. Finally, this work may also inform efforts to develop a real-time tool that can identify, and perhaps remediate, situations in which there is a substantial risk of a go-around.

We found that almost 90% of detected go-arounds in our data set occurred within five nautical miles of the landing runway threshold. To obtain features that are proximate in time to all go-around initiations, but without losing too many go-around observations, we choose the 5 nm information cutoff gate for the contribution analysis. In other words, we develop the principal component logistic regression (PCLR) model to model go-around occurrence based only on features known when the subject flight reaches the 5 nm information cutoff gate. We first illustrate the detailed algorithm and estimation procedures of the PCLR model. The estimation results are then interpreted through factor loading analysis and reconstruction of coefficients back for the original variables. Lastly, we construct counterfactual scenarios to quantify factor contributions to go-arounds, based on the models estimated at the cutoff gate of 5 nm.

6.2. Data Preprocessing

The dependent variable Y is set to 1 if a flight is detected as a go-around occurring within [0, 5) nautical miles to its landing runway threshold, 0 otherwise. Flights that initiate go-arounds more than 5 nm from the threshold are not considered. For example, a flight that initiated go-around at 4.5 nautical miles from the runway is included in the data set, while a flight that initiated a go-around at 5.1 nm from the runway is removed from our analysis.

For explanatory variables, we use the linear extrapolation technique to derive the seven categories of features (in Chapter 4) at the 5 nm information cutoff gate. Therefore, we evaluate whether a flight initiates the go-around anywhere at [0, 5) nautical miles only based on the information available when this flight passes the 5 nm arc. The extrapolation

guarantees that we do not include any information that cannot be known in the feature space when the aircraft is at the 5 nm information cutoff gate.

Originally, 0.43% of JFK arrivals are detected as go-arounds within the period. After data preprocessing and matching flight trajectories with the 5 nm features, our final dataset has a total of 343 go-arounds initiated within 5 nm (5 nm is exclusive) from the landing runway threshold, which accounts for 0.34% of JFK arrivals between July and December of 2018.

By applying the retrospective causal inference method [90] to observational data, we can capture the statistical relations among the go-around occurrence and features described in Chapter 4. Toward this end, we first apply the go-around detection algorithm presented in Chapter 3 to the JFK arrival flight track dataset in 2018. Second, we collect a large set of features that may affect go-around occurrence, as described in Chapter 4, and build a principal component logistic regression model (PCLR) to establish statistical relations between the derived features and go-around occurrence. Lastly, we used the estimated PCLR model to construct counterfactual scenarios to estimate the contributions of different factors to go-around occurrence.

6.3. Standard Logit Model

We firstly estimated a standard binary logistic regression model to relate go-around occurrence to contributing factors. The model specification is formulated as in Equation (25) and Equation (26), where $\boldsymbol{\mathcal{V}}$ is the log-odds function, $\boldsymbol{X}$ is a design matrix that contains all contributing factors introduced in Chapter 4, and $\boldsymbol{\beta}$ is the associated coefficient vector estimated by employing maximum likelihood estimation (MLE).

$$\boldsymbol{\mathcal{V}} = \boldsymbol{X} \cdot \boldsymbol{\beta} \tag{25}$$

The probability of an aircraft initiating go-around $P_r\ (y_i = 1|\ \boldsymbol{X})$ can be written as:

$$\Pr(y_i = 1|\mathbf{X}) = \frac{1}{1 + \exp(-\boldsymbol{\mathcal{V}})} \tag{26}$$

The estimation results are presented in Table 8. The majority of coefficients are not significant at a 5% confidence level, and many have unexpected signs. For example, the estimates for the visibility and ceiling variables suggest that flights landing at an airport with good visibility and ceiling conditions would have a higher probability of go-around, which is not plausible in practice. This is probably because many independent variables used in the model are highly correlated. As a result of this multi-collinearity, the standard logistic regression model fails to give us a proper understanding of the contributing effects. To remedy this problem, we employ decorrelation techniques.

Table 8. Standard logit model estimation results.

Variable	Estimate (std.)	Variable	Estimate (std.)	Variable	Estimate (std.)	Variable	Estimate (std.)
Constant	-5.252*** (0.847)	**Visibility$_1$**	-0.175 (0.169)	**ArrQue**	0.026** (0.010)	**RwyChange**	-0.016 (0.206)
AltDev	0.258*** (0.035)	**Visibility$_2$**	0.017 (0.079)	**DepQue**	0.022** (0.007)	**Rwy04R**	-1.330*** (0.255)
Speed	-0.022** (0.007)	**Visibility$_3$**	-0.171* (0.070)	**RwyCnt**	-0.054 (0.037)	**Rwy13L**	-4.156*** (0.742)
Energy	0.003*** (0.000)	**Ceiling$_1$**	-0.282 (0.158)	$\widehat{ROB}$	-0.005 (0.003)	**Rwy13R**	1.636** (0.603)
Angle	0.032* (0.013)	**Ceiling$_2$**	0.010 (0.031)	**IMC**	0.493 (0.385)	**Rwy22L**	-0.996*** (0.253)
LOS	1.682*** (0.234)	**Ceiling$_3$**	0.011 (0.016)	**Daytime**	0.080 (0.124)	**Rwy22R**	-1.803*** (0.348)
SpeedDiff	0.001 (0.004)	**Ceiling$_4$**	-0.006** (0.002)	**AirlineIntl**	0.425** (0.162)	**Rwy31L**	-1.333*** (0.333)
AltDiff	-0.105*** (0.027)	**GaCnt**	0.546*** (0.088)	**BodyWide**	0.328* (0.158)	**Rwy31R**	-1.390*** (0.310)
Wind	0.045** (0.014)	**GaGap**	-0.000 (0.000)	**NoLead**	0.305 (0.278)		
Log-likelihood	-1721.7			**Pseudo R-squared**	0.222		

Variables are significant at the 0.1% level***, 1% level**, 5% level*.

6.4. Principal Component Logistic Regression (PCLR) and Interpretation

To handle the multi-collinearity problem, we apply Principal Component Analysis (PCA) to decorrelate and reduce the dimensionality of the original feature space. Instead of regressing the dependent variable on the explanatory variables directly, the principal components (PCs) formed by all the explanatory variables are used as covariates in the logistic regression model.

6.4.1. PCLR of mixed data

While PCA is a mature technique to decorrelate feature vectors, it must be adapted in our setting because our dataset contains a mixture of continuous and categorical variables. Specifically, the design matrix (feature space) for the 5-nautical-mile model contains 28 features vectors, 21 of which are continuous and seven are categorical, including *Runway* (8 levels), *RwyChange* (2 levels), *Daytime* (2 levels), *Airline* (2 levels), *Body* (2 levels), *MC* (2 levels) and *NoLead* (2 levels). Therefore, appropriate treatment of such mixed data

types, especially the categorical variables, is required for PCA application. Accordingly, we adapted and applied the PCA-mixed algorithm introduced by [91] to deal with our mixed set of variables. The detailed notations and algorithm are described as follow.

A. Notations

Let $\boldsymbol{X} = \left[\boldsymbol{X}_1^{n\times p_1} \middle| \boldsymbol{X}_2^{n\times p_2}\right]$ denote the full design matrix, which is constructed by two submatrices $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$. $\boldsymbol{X_1}$ contains solely continuous variables with dimension n by $p_1 (p_1 = 21)$, while $\boldsymbol{X_2}$ contains categorical variables with dimension n by p_2 $(p_2 = 7)$. We further denote $q_1, q_2, \dots, q_{p_2}$ as the number of levels for each categorical variable (e.g., $q_1 = 8$ for the first categorical variable, which is *Runway*), and $m = \sum_{i=1}^{p_2} q_i$ as the total levels for all categorical variables. Notice that the elements in $\boldsymbol{X_2}$ are integers that range from 1 to the number of levels for each variable.

B. Design Matrix Preparation

Using the above notation, we first convert $\boldsymbol{X_2}$ to a complete disjunctive table (CDT) $\boldsymbol{Z_2} = [\boldsymbol{z_1}, \boldsymbol{z_2}, \dots, \boldsymbol{z_m}] \in \mathbb{B}^{n\times m}$ by employing one-hot encoding. Then we center $\boldsymbol{Z_2}$ by respectively subtracting the mean of each column, denoted as $\boldsymbol{Z_2^c}$, and standardize $\boldsymbol{X_1}$ to zero mean and unit standard deviation, denoted as $\boldsymbol{X_1^s}$. Lastly, we combine $\boldsymbol{X_1^s}$ and $\boldsymbol{Z_2^c}$ to build a new design matrix $\boldsymbol{Z} = [\boldsymbol{X_1^s}|\boldsymbol{Z_2^c}]$. Notice that the rank of $\boldsymbol{Z}$ equals to $r = p_1 + m - p_2$.

C. Generalized Singular Value Decomposition (GSVD)

We first define a weighting matrix $M = \begin{bmatrix} \mathbb{I}_{p_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{bmatrix}$, where $\mathbb{I}_{p_1}$ is an identity matrix with dimension p_1. $\boldsymbol{W} = [w_{ii}] \in \mathbb{R}^{m\times m}$ is a diagonal matrix where $w_{ii} = \frac{n}{\mathbf{1}^{\mathrm{T}}\cdot z_i}$, and $\mathbf{1}$ is a vector of ones. Then we perform generalized singular value decomposition on the product of matrices $\boldsymbol{Z}$ and $\boldsymbol{M}$, Equation (27), where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, and $\boldsymbol{\Lambda}$ is a diagonal matrix that contains singular values sorted by their values.

$$\boldsymbol{Z}\cdot\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^{\boldsymbol{T}} \tag{27}$$

Notice that in Equation (27), $\boldsymbol{V} = \left[\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_{p_1+m}}\right]$ represents the principal component directions of the matrix $\boldsymbol{Z}\cdot\boldsymbol{M}$, and $\boldsymbol{\Lambda} = \mathrm{diag}\left\{\sqrt{\sigma_1}, \sqrt{\sigma_2}, \cdots, \sqrt{\sigma_{p_1+m}}\right\}$ where σ_i's are eigenvalues of $\boldsymbol{M}^{\mathbf{T}}\boldsymbol{Z}^{\mathbf{T}}\boldsymbol{Z}\boldsymbol{M}$. Thus, we can find the principal components of $\boldsymbol{Z}\cdot\boldsymbol{M}$, that is $\boldsymbol{F} \in \mathbb{R}^{n\times r}$, by using Equation (27). $\boldsymbol{F}$ has the same rank as $\boldsymbol{Z}$.

$$\boldsymbol{F} = \boldsymbol{Z}\cdot\boldsymbol{M}\cdot\boldsymbol{V} \tag{28}$$

D. Derived Covariates

Common techniques to derive covariates from $\boldsymbol{F}$ include: (a) pick the top k columns with the sum of explained variances that exceeds some thresholds; (b) pick the top k columns with the smallest squared singular value (i.e., eigenvalue) exceeding some

threshold; and (c) pick the top k columns with the sum of squared singular values exceeding some threshold. However, these variance-based or singular-value-based criterion might not always be optimal in predictive analytics. PCs with large variances are not necessarily the best predictors [92] as principal components with low explained variability could be highly correlated with the response variable. Therefore, the dependence between response and predictor variables must be taken into account.

In order to choose the number of principal components k big enough to account for the variance in the data as much as possible, and also reduce the dimensionality of the data, we have applied the Kaiser rule [93, 94, 95] by iteratively selecting k PCs using the aforementioned criteria (b) with a threshold δ. When varying the threshold δ from 0.5 to 1.0 with a step of 0.1, we regress selected PCs with the response value Y using logistic regression, and record the model's adjusted pseudo R-squared. We determine the final δ and k PCs with the best adjusted pseudo R-squared.

We denote the selected PCs as $\boldsymbol{F_k}$, and hereafter we use $\boldsymbol{F_k}$ as the final design matrix to conduct logistic regression analysis using the same technique described in Section 6.3. except that in Equation (25), we use the feature vectors in $\boldsymbol{F_k}$ instead of $\boldsymbol{X}$.

E. Transformation of Estimated Coefficients

While the PCLR regime gives us estimates for principal components, we eventually desire estimated coefficients of the actual features derived in Chapter 4 (e.g., ceilings and runway fixed effects). Let $\boldsymbol{\alpha}$ denote the coefficient vector for PCs, then the utility function (3) can be rewritten as:

$$\boldsymbol{\mathcal{V}} = \boldsymbol{F_k} \cdot \boldsymbol{\alpha} = (\boldsymbol{Z} \cdot \boldsymbol{M} \cdot \boldsymbol{V_k}) \cdot \boldsymbol{\alpha} = \boldsymbol{Z} \cdot (\boldsymbol{M} \cdot \boldsymbol{V_k} \cdot \boldsymbol{\alpha}) = \boldsymbol{Z} \cdot \boldsymbol{\beta} \tag{29}$$

where $\boldsymbol{V_k}$ is the first k columns of the matrix $\boldsymbol{V}$.

Given Equation (29), the logistic regression model with respect to $\boldsymbol{F_k}$ can be equivalently expressed in matrix form with respect to the original feature space $\boldsymbol{Z}$. Thus, the associated coefficient vector is given by:

$$\boldsymbol{\beta} = \boldsymbol{M} \cdot \boldsymbol{V_k} \cdot \boldsymbol{\alpha} \tag{30}$$

6.4.2. Factor Loading Analysis

In addition to obtaining principal components (PCs) and their associated estimates, we are also interested in linking PCs to the original feature space in order to identify the variables that are primarily associated with any given PC. To do so, we use factor analysis to map PCs to groups of features quantitatively.

We first denote $\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L_1} \\ \boldsymbol{L_2} \end{bmatrix}, \boldsymbol{L_1} \in \mathbb{R}^{p_1 \times k}, \boldsymbol{L_2} \in \mathbb{R}^{m \times k}$ as the loading matrix representing the variance in features explained by PCs. The formal definition is formulated as:

$$\boldsymbol{L} = \boldsymbol{M} \cdot \boldsymbol{V_k} \cdot \boldsymbol{\Lambda_k} \tag{31}$$

where $\boldsymbol{V_k}$ is the first k columns of matrix $\boldsymbol{V}$, and $\boldsymbol{\Lambda_k}$ is the k[th] order leading principal minors of $\boldsymbol{\Lambda}$. However, due to the fact that continuous variables and categorical variables are not on the same scale, and thus we cannot compare their explained variance by PCs directly from $\boldsymbol{L}$. We have derived a ***contribution matrix*** $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C_1} \\ \boldsymbol{C_2} \end{bmatrix} \in \mathbb{R}_+$ in which each element c_{ij} describes the contribution of the i[th] feature to the j[th] PC. Specifically, a larger value of c_{ij} indicates a higher contribution of the i[th] feature to the j[th] PC. Furthermore, the $\boldsymbol{C}$ matrix can be decoupled into two submatrices where $\boldsymbol{C_1} \in \mathbb{R}_+^{p_1 \times k}$ and $\boldsymbol{C_2} \in \mathbb{R}_+^{p_2 \times k}$ respectively correspond to the continuous and categorical contribution matrices. Equation (32)(33)(34) illustrate the derivation of $\boldsymbol{C}$.

$$\boldsymbol{C_1} = \mathbf{L_1} \circ \mathbf{L_1} \tag{32}$$

$$\boldsymbol{C_2} = \mathbf{H} \cdot (\mathbf{L_2} \circ \mathbf{L_2}) \tag{33}$$

where $\circ$ denotes element-wise multiplication. $\boldsymbol{H} \in \mathbb{R}^{p_2 \times m}$ is a block diagonal matrix in which the diagonal elements are vectors of the level frequency of the i[th] categorical variable, and the off-diagonal elements are 0. $q_1, q_2, \dots, q_{p_2}$ are the number of levels for each categorical variable (e.g., $q_1 = 8$ for variable *Runway*), and $m = \sum_{i=1}^{p_2} q_i$ is the total levels for all categorical variables.

$$\boldsymbol{H} = \begin{bmatrix} \left(\frac{\mathbf{1}^{\mathrm{T}} \cdot z_{11}}{n} \;\; \frac{\mathbf{1}^{\mathrm{T}} \cdot z_{12}}{n} \;\; \dots \;\; \frac{\mathbf{1}^{\mathrm{T}} \cdot z_{1(q_1)}}{n}\right) & 0 & \cdots & 0 \\ 0 & \left(\frac{\mathbf{1}^{\mathrm{T}} \cdot z_{21}}{n} \;\; \frac{\mathbf{1}^{\mathrm{T}} \cdot z_{22}}{n} \dots \;\; \frac{\mathbf{1}^{\mathrm{T}} \cdot z_{2(q_2)}}{n}\right) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \left(\frac{\mathbf{1}^{\mathrm{T}} \cdot z_{p_2 1}}{n} \;\; \frac{\mathbf{1}^{\mathrm{T}} \cdot z_{p_2 2}}{n} \dots \;\; \frac{\mathbf{1}^{\mathrm{T}} \cdot z_{p_2 (q_{p_2})}}{n}\right) \end{bmatrix} \tag{34}$$

With such operations, the contribution of a categorical variable is the weighted sum of the squared loadings of its classes in $\boldsymbol{L_2}$ and therefore is equivalent to their correlation ratios. For the contribution of a continuous variable, on the other hand, the squared loading equals the squared correlation.

6.5. Estimation Results

Using the above Kaiser rule, the first 17 principal components with $\delta = 0.8$ that explain 82% of the total variance have the best adjusted pseudo R-squared. In the end, we are left with 9 PCs after removing insignificant principal components from the estimated logistic regression model.

6.5.1. Factor Analysis and Model Result

For the convenience of understanding the model estimated coefficients for each factor (PC), we first need to deploy the pattern matrix $\boldsymbol{L}$ (Equation 31) and the contribution matrix

$\boldsymbol{C}$ (Equation 32-34) from factor analysis to interpret the relationships between PCs and the original features. The pattern matrix $\boldsymbol{L}$ indicates the magnitude and sign of the correlation between the PCs and the original variables. The contribution matrix $\boldsymbol{C}$ transforms the continuous variable contributions and the categorical variable contributions on the same scale. It then describes the magnitude of the overall contribution of an individual variable to a PC.

Table 9. Loadings of variables with above-average contributions for each PC.

PC	*Related Variable (x)*	*Loading (l_{ij})*	*Semantic Labels*
1	$Visibility_1$	-0.715	Meteorological conditions
	$Visibility_2$	-0.817	
	$Visibility_3$	-0.821	
	$Ceiling_1$	-0.643	
	$Ceiling_2$	-0.823	
	$Ceiling_3$	-0.838	
	$Ceiling_4$	-0.616	
	IMC	2.000	
2	AltDiff	0.724	Lead-trail spacing
	Energy	0.719	
	$\widehat{ROB}$	-0.411	
	SpeedDiff	0.220	
	NoLead	-1.232	
4	Angle	0.618	Approach procedure features
	AltDev	0.457	
	Speed	0.430	
	Rwy04R	-0.781	
	Rwy22L	-0.682	
5	ArrQue	0.592	Traffic conditions
	DepQue	0.586	
	GaGap	-0.399	
	GaCnt	0.210	
	RwyCnt	0.307	
	Rwy22R	1.726	
6	AirlineIntl	1.396	Aircraft characteristics
	BodyWide	1.293	
7	Rwy13R	0.975	Approach pattern
	Rwy31L	-1.136	
8	Wind	0.623	Wind and daytime effect
	Daytime	0.689	
13	RwyChange	1.172	Approach pattern
	Rwy13L	-1.415	
	Rwy31R	0.606	
15	LOS	0.812	Loss of separation

The first and second columns of Table 9 are determined by finding which variable(s) – either continuous or categorical – make above-average contribution(s) to each PC based on contribution matrix $\boldsymbol{C}$. The average contribution is the value when all variables have the

same contributions (i.e., 100% divided by the total number of variables). In the case of categorical variables, which may make an above-average contribution to more than one PC, we base the assignment on the maximum loading. We present the loading value – either positive or negative – of the i^{th} variable to the assigned j^{th} PC (from pattern matrix $\boldsymbol{L}$) in the third column. To save space, we only show the loading (l_{ij}) of the i^{th} variable to the assigned j^{th} PC in Table 9 rather than the whole pattern matrix $\boldsymbol{L}$ and the whole contribution matrix $\boldsymbol{C}$. Note that variables may have high loading values on just one or two PCs, or have a balanced spread with small loading values across more PCs. According to the allocation result, we assign semantic labels for each PC in the fourth column and use them to interpret the estimation results of the PCLR model in Table 9.

Turning to Table 10, we note that PC1 has a highly significant, positive coefficient estimate. From Table 9, we see that the visibility and ceiling variables are loaded in the opposite direction with PC1, while the IMC indicator variable is loaded in the same direction. Thus, the PC1 result indicates that adverse meteorological conditions (low visibility and ceiling, or IMC condition) increase the probability of the go-around occurrence.

The coefficient estimate of PC2 indicates that the threat of the lead and trail aircraft simultaneously occupying the runway increases the likelihood of a go-around. A small runway occupancy buffer ($\widehat{ROB}$), or the trailing aircraft approaching with high energy (*Energy*), or the trailing aircraft chasing too close to its leading flight (*SpeedDiff*) increases this threat, while the absence of a lead aircraft (*NoLead*) clearly reduces it. Similarly, the positive coefficient on PC15 implies that a higher loss of separation compared to FAA standards increases the probability of go-around.

Table 10. PCLR model estimation results.

Dimension	Est./Std.	Dimension	Est./Std.	Dimension	Est./Std.
Constant	-6.560*** (0.088)				
PC1	0.252*** (0.018)	**PC5**	0.152*** (0.036)	**PC8**	0.177*** (0.046)
PC2	0.207*** (0.039)	**PC6**	0.407*** (0.040)	**PC13**	0.453*** (0.036)
PC4	0.149*** (0.043)	**PC7**	0.144*** (0.033)	**PC15**	0.180*** (0.022)

Variables are significant at the 0.1% level***, 1% level**, 5% level*.

The PC4 captures the effect of approach procedure deviations. The positive coefficients PC4 in Table 10 implies that flights that are deviated from the optimum approach procedure (3-degree glideslope, runway alignment, proper speed control) are more likely to initiate go-arounds. The landing runway indicators 04R and 22L are also captured in this PC,

perhaps indicating that approaches to these two runways are correlated with the other PC4 covariates. It suggests that flights landing on runway 04R/22L would be less likely to initiate go-arounds than other runways, perhaps because 04R/22L has the most advanced landing aids of the JFK runways. As reported by [96], Runway 4R is a Category IIIB ILS runway, permitting landings with as little as 600 feet of visibility; Runway 22L is equipped with a Precision Approach Path Indicator (PAPI) and allows landings down to visibility of less than a half-mile (2640 feet), while other runways at JFK require more than half-mile visibility for landing. These technologies make it easier for pilots to land, alleviate the operational risks, and avert go-arounds.

The arrival queue, departure queue, go-around clustering effect, and the number of objects occupying the runway during the landing process are captured by PC5, which has a positive impact on the go-around occurrence. This may indicate that increased controller workload or pressure to maximize throughput increases the go-around probability. PC5 also picks up the clustering effect whereby go-arounds are more likely in the time period surrounding a given go-around.

The PC6 captures the aircraft characteristics – fixed effects of international airliners and wide-body aircraft, which is found a significant positive impact on go-around occurrence. The daytime operations and strong winds (PC8) increases the likelihood of go-around occurrences, as does a change of runway configuration (PC13). This could reflect how the configuration change interrupts traffic patterns, increasing pilot and controller workload.

Besides the landing runway 04R/22L loaded by PC4, the fixed effects of other landing runway variables for capturing different approach patterns are loaded in different PCs – 22R in PC5, 13R/31L in PC7, and 13L/31R in PC13. All of these PCs are statistically significant which suggests that, all else equal, this is a greater proclivity toward go-arounds in certain landing runways, but further investigation is required to interpret the relationship between runway configuration and go-around occurrence.

6.5.2. Transformation of coefficients

In the above section, we interpret how the original derived features impact go-around occurrence using the assigned semantic labels for each PC based on factor analysis. This section further quantifies the impacts of the original derived features by reconstructing their estimates using Equation (30). The results are presented in Table 11. The coefficient estimates of the original features are based on the assumption that the features affect go-around occurrence through their effects on the factors included in the model. Compared to the standard logit model estimation results in Table 8, the coefficients in Table 11 are quite different, and the standard errors of the coefficient estimates are much lower. Nearly all the coefficients become statistically significant at the 1% confidence level and have expected signs. The PCLR model removes collinearity without eliminating any of the original variables and reduces the variance of the estimated coefficients. The majority of the estimates (β) are consistent with the discussions in the factor loading analysis. Note,

however, that individual coefficient estimates are biased and the main value of the PCLR method is in the estimates of the latent variable coefficients. Practice [97] has shown that strong collinearity induces the conditions under which the PCLR method is beneficial, in that the PCLR allows for minor bias for the sake of substantially smaller variance and improved model interpretability [98].

Table 11. Reconstructed coefficients of original features.

Variable	Reconstructed Coef. β (std.)	Variable	Reconstructed Coef. β (std.)	Variable	Reconstructed Coef. β (std.)	Variable	Reconstructed Coef. β (std.)
Constant	-6.560*** (0.088)	**Visibility$_1$**	-0.164*** (0.030)	**ArrQue**	0.016*** (0.003)	**RwyChange**	0.562*** (0.004)
AltDev	0.015*** (0.006)	**Visibility$_2$**	-0.061*** (0.005)	**DepQue**	0.013*** (0.002)	**Rwy04R**	-0.036 (0.029)
Speed	0.010*** (0.001)	**Visibility$_3$**	-0.052*** (0.005)	**RwyCnt**	0.011 (0.008)	**Rwy13L**	-0.209** (0.073)
Energy	0.001*** (0.000)	**Ceiling$_1$**	-0.190*** (0.038)	$\widehat{ROB}$	-0.005*** (0.001)	**Rwy13R**	2.758*** (0.180)
Angle	0.001** (0.000)	**Ceiling$_2$**	-0.040*** (0.003)	**IMC**	0.244*** (0.006)	**Rwy22L**	-0.094** (0.035)
LOS	2.101*** (0.215)	**Ceiling$_3$**	-0.008*** (0.001)	**Daytime**	0.201*** (0.023)	**Rwy22R**	0.383*** (0.068)
SpeedDiff	0.004*** (0.000)	**Ceiling$_4$**	-0.001** (0.000)	**AirlineIntl**	0.463*** (0.014)	**Rwy31L**	-0.420*** (0.049)
AltDiff	0.023*** (0.005)	**GaCnt**	0.549*** (0.046)	**BodyWide**	0.445*** (0.015)	**Rwy31R**	0.107 (0.070)
Wind	0.058*** (0.007)	**GaGap**	-0.001*** (0.000)	**NoLead**	-0.473*** (0.005)		

Variables are significant at the 0.1% level***, 1% level**, 5% level*.

We here plot the coefficients of different visibility and ceiling discretized variables in Figure 21. The green bar represents visibility (in statute miles), and the blue bar represents the ceiling (in 100 feet). We observe that go around occurrence is more sensitive to visibility and ceiling variation when these values are less than 3 statute miles and less than 500 feet, respectively, and that this sensitivity declines markedly as these conditions improve.

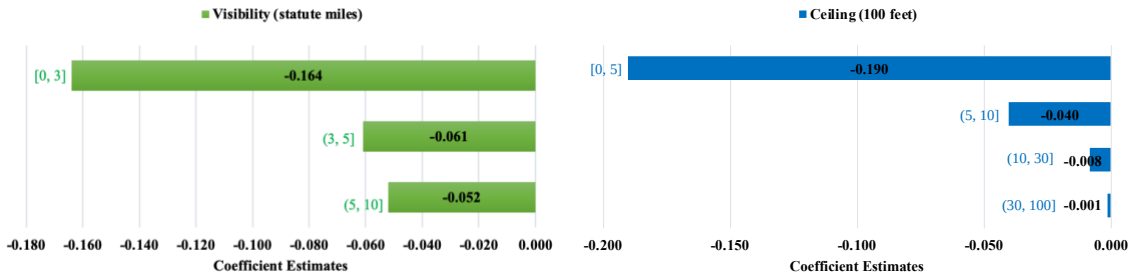

Figure 21. Bar plot of visibility and ceiling effects.

6.5.3. Counterfactual Analysis

In this section, we directly measure the contributions of different factors to the go-around occurrence by conducting a counterfactual analysis. Each counterfactual scenario is constructed by setting a particular feature to its "best" value while leaving the other features unchanged. For example, model estimates suggested that the ceiling has a negative effect on go-around occurrence. To construct the counterfactual scenario for ceiling, we set the ceiling to 10,000 feet for each observation in the data set. Based on this assumption we reset the values of the various ceiling-related valuables, while leaving all other values unchanged. Then, we use the estimated PCLR model to predict the corresponding go-around probability for each flight. The variable contribution is calculated by measuring the percentage reduction between the baseline go-around rate and the expected go-around rate using Equation (35). Note that we assume individual feature does not directly impact go-around occurrence but via their effects on the PCA factor scores F.

$$\%reduction = \frac{P_{GA} - E(P_{GA}|X')}{P_{GA}} \times 100\% \tag{35}$$

where $E(P_{GA}|X')$ is the expected go-around rate given the counterfactual input X'; P_{GA} is the baseline go-around rate.

Table 12 reports the scenario value for each variable, that is the value found in the data that, based on the sign of its coefficient, would minimize go-around occurrence (labeled "Expected GA%"), and the percentage reduction in go-around occurrence relative to the observed baseline of 0.343%.

Table 12. Counterfactual analysis results.

Baseline go-around rate: 0.343%			
Variable	**Scenario value**	**Expected GA%**	**% Reduction in GA's under scenario**
AltDev	0	0.240%	29.97%
Speed	77.28		
Energy	1318.72		
Angle	0		
LOS	0	0.248%	27.59%
SpeedDiff	-86.95		
DepQue	-15	0.256%	25.46%
$Ceiling_1$	5	0.257%	25.12%
$Ceiling_2$	10		
$Ceiling_3$	30		
$Ceiling_4$	100		
MC	VMC	0.258%	24.86%
Body	Narrow	0.259%	24.60%
$Visibility_1$	3	0.272%	20.62%
$Visibility_2$	5		
$Visibility_3$	10		
Wind	0	0.273%	20.47%
NoLead	1	0.276%	19.47%
$\widehat{ROB}$	146.79	0.277%	19.39%
ArrQue	-15	0.290%	15.51%
RwyChange	0	0.306%	10.86%
GaGap	1440	0.307%	10.41%
GaCnt	0		
AirlineIntl	0	0.310%	9.52%
RwyCnt	0	0.313%	8.88%

The relative importance of the variables on the reduction of go-around probability is shown in Figure 22, where each row represents one variable. The length of the color bar indicates the variable contribution in percentage. The stability of an approach, flight lead-trail spacing, departing traffic and airport ceiling are the most important factors of go-around occurrence. If the flight aligns the extended runway centerline properly at 5 nm, strictly follows the 3-degree glideslope, and maintains the effective speed control and energy management, the go-around rate would potentially decline about 30%. For aircraft forming in-trail relationships, the go-around rate would also decline by about 28% by maintaining appropriate following speed and keeping safe spatial separation, while the absence of a lead aircraft (*NoLead*) would result in a 19% drop of the go-around rate. Managing and optimizing the departure queue seems to have a more substantial contribution to decreasing go-around rates (25%) than reducing the arrival queue (16%). The go-around rate would drop by more than 25% if the airport ceiling were set to its ideal scenario value, and 21% if there were high visibility. If all the flights in the observation dataset are narrow-body aircraft or operated by domestic airliners, the go-around rate decreases by 25% and 9%, respectively. The wind speed effect contributes 20% to the reduction of go-around occurrence. Ensuring that there are no aircraft or vehicles on the runway safety area when a flight is 5 nm from its landing runway threshold, would result in a 9% reduction of go-around occurrence. Finally, eliminating the clustering effect would reduce the go-around occurrence by 10%.

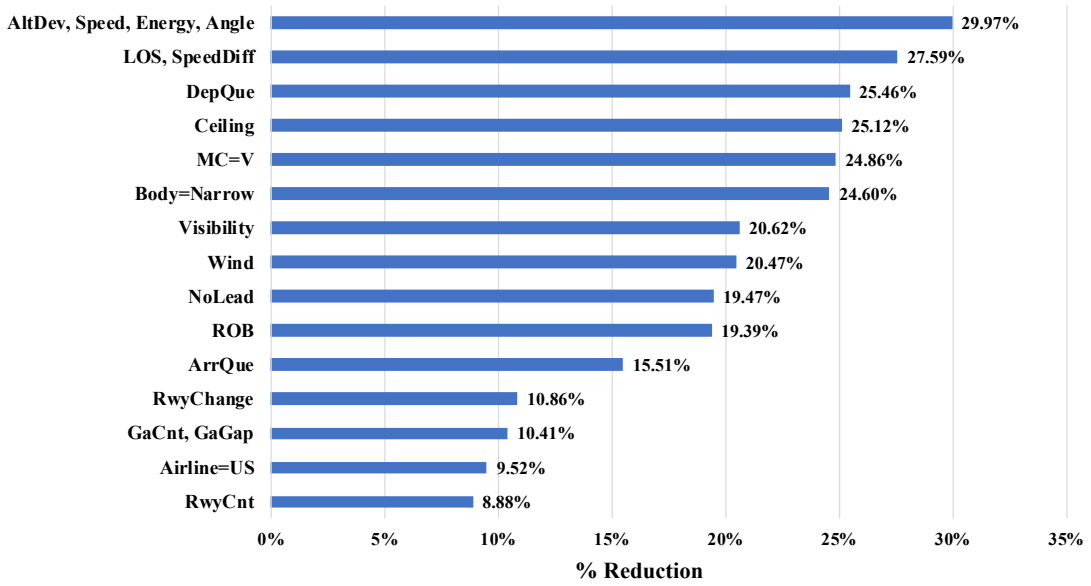

Figure 22. Relative variable contribution in reducing go-arounds.

6.6. Chapter Summary

In this section, we develop the principal component logistic regression model to quantify the contribution of a wide range of factors to go-around occurrence. Specifically, we have designed a trajectory-based go-around detection algorithm and applied it to JFK arrival flights in 2018. Multiple datasets have been fused to capture features that may influence flight approach procedures and therefore help understand the causes of go-around occurrence. In developing the feature engineering, we have collected features from the dataset directly, and used domain knowledge to derive features, such as loss of separation and runway occupancy buffer. We then established statistical relationships between go-around occurrence with those derived features and estimated their effects using PCLR model and factor loading analysis. Lastly, we quantify the contribution of various features to go-around occurrence through counterfactual analysis. Conclusions are in line with research using full-flight simulator trials [64], interviews with ATC controllers and pilots [26], and a realized trajectory dataset [16].

As far as the authors know, this is the first work to detect, model, and interpret go-around occurrence from surveillance data, considering a broad set of environmental and operational variables. This enables us to assess the relative importance of a wide range of factors in determining go-around probability. We find that there is no single dominant factor. Factors in the top tier of importance include the state of the subject aircraft, its separation and speed difference from the aircraft in front, and factors related to visibility, cloud ceiling, and the subject aircraft type. Among these factors the first two are, in principle, subject to improvement through pilot and controller training, and thus inviting targets for initiatives to reduce go-arounds. These conclusions must be qualified by the strong assumption that individual features influence go-around occurrence via their contributions to factors. Larger data sets are required to overcome the multi-collinearity between features so that feature coefficients can be estimated directly rather than through principal components.

In addition to the scientific contribution, it has a variety of practical applications. This research could lead to a real-time monitoring tool that can anticipate, and perhaps remediate, situations in which there is a substantial risk of go-arounds (Chapter 7). The model can also supply tactical instructions for controllers and pilots about the probability of go-arounds under varying conditions during approach procedures. It would be helpful for decision support monitoring and prediction-based alerting in advance to improve flight approach safety and airport efficiency. Our results can also inform strategies to reduce go-arounds by identifying the most salient contributing factors, some of which may be mitigated. Also, by summarizing historical patterns of go-around occurrence, our study can augment the limited individual experience of air traffic controllers and pilots, and this informs their judgment about whether a go-around is warranted.

Several improvements can be built upon the presented work. As noted above, one important direction is to overcome multi-collinearity among different features, presumably by employing a larger data set across multiple airports. Another interesting extension is to

develop models at other distances and explore the evolutionary impacts of feature contributions on go-around occurrence. In this manuscript, the PCLR model is based on the features that are available when a flight is at 5 nm from its landing runway threshold. The algorithms and the estimation procedures can be applied to features spaces defined at different information cutoff gates from 10 nm to 1 nm. Thirdly, our methods can be extended to other types of atypical flight approach events, such as the unstabilized approach, short approach, dogleg approach, etc. Lastly, other contributing factors, such as crew-controller communications [99] and commercial pressure to maintain flight schedules, may be considered in future work.

7. Sequential Prediction of Go-Around Occurrence

7.1. Overview

While the contribution analysis in Chapter 6 helps decision-making at a strategic level, being able to predict go-around probabilities for each landing aircraft over the entire approach could provide tactical guidance to foresee and perhaps prevent go-arounds. If the probability of a go-around can be predicted in advance and be imparted to operators (e.g., air traffic controllers and pilots) in time, more proactive mitigations can be taken to alleviate the operational risks and increase airport efficiency as go-arounds might be averted. Toward this end, in this chapter, we leverage the go-around detection results in Chapter 3, derived features in Chapter 4 and Chapter 5, and observation-driven insights from Chapter 6 to develop machine learning models from making sequential predictions of go-around probabilities of individual flights as they approach the airport.

This chapter aims to develop predictors that can incorporate the operational conditions and environmental measures for the real-time prediction of go-arounds. This work enables a real-time system-wide safety assurance (Chapter 9) that establishes a practical safety-enhancing risk detection tool for ANSPs, airports, airlines, and other stakeholders.

7.2. Related Work

There are two common approaches to sequence prediction: one is to use snapshot (point) features extracted from the sequence to build multiple static supervised learning models at every timestamp; The other is to use time series of relevant data for temporal models that can learn the inherent temporal structures of the entire sequence.

For the applications of using snapshot features, Martinez et al. [100] apply gradient boosting methods to predict the runway occupancy times at Vienna airport. For each flight sequence, the instant in which the prediction is made has been set at [10, 9.5, 9, …, 2.5, 2] nautical miles before the landing runway threshold – every 0.5 nm between the landing runway threshold and 10 nm from that. In Chapter 5, we also apply the similar regime to train several machine learning models to predict the ROB, when the trailing aircraft is at [10, 9, …, 3, 2] nm from the landing runway threshold.

Temporal models capture the temporal dynamics in a more flexible way and allow a "memory" of the previous inputs to persist in the internal hidden units, which then influences the prediction result. Markovian models and recurrent networks, in which the structure is a directed acyclic graph, are often employed to learn the inherent temporal dependence structure of the sequence. Both methods have shown promise in the aviation field, especially when dealing with flight trajectory data. HMM has gained more attention

in recent years, particularly for predicting flight trajectories. Ayhan and Samet [58] propose an HMM to predict full four-dimensional trajectories, given the observed weather cube sequence (temperature and wind). In this model, the positions of the aircraft are regarded as hidden states, and the weather cube observed around the track point is a realization of the hidden states. To evaluate airport system operational behavior for safety control, Rodriguez-Sanz et al. [101] have applied the HMM framework to access airspace-airside turnaround operations. This paper defines hidden states with performance thresholds (e.g., delay, throughput) and expert knowledge, but does not undertake sensitivity analysis on the selected threshold values to ensure model robustness. Liu and Hansen [59] developed an encoder-decoder LSTM network for four-dimensional aircraft trajectory prediction in a real-time setting. The model is applied to a dataset including 1,679 flight trajectories and achieves good predictability.

7.3. Problem Formulation

For each flight approaching the airport, our goal is to predict whether or not the flight will initiate a go-around during the remainder of its approach, given the sequence of the *realized* track points and their associated feature vectors (currently available information) gleaned from the datasets at a certain point in time. The underlying question we seek to answer is that, by solely observing the current flight status and its surrounding traffic and weather environment, how well do the predictive models learn to predict the probability of go-around initiation, after the flight passes a certain information cutoff gate?

This task can, therefore, be viewed as a "many-to-many" sequential prediction problem, with the first "many" implying that the observed feature space could involve more than one input time step, and the second "many" indicating that our final prediction will consist of labels at one or more time steps based on a sequence of recent observations. Notably, the number of input time steps (the first "many") does not have to match the number of output time steps (the second "many"). We treat the flight approach procedure as a stochastic process, and model the complex dynamical flight approach procedures in a recurrent processing style.

Recall from Section 4.2 that we applied linear extrapolation to discretize the flight trajectories at information cutoff gates, and we will only derive features at those gates. For each labeled go-around flight, we treated the timestamp/trackpoint at the start of ascent as the initial time/location for the go-around procedure, and further truncated the trajectory after that point. Therefore, each flight trajectory considered to be a go-around can be represented by *at most* ten track points, which end before the flight altitude increases. Before a go-around is initiated, all the go-around labels $G_{i,d}$ for flight i at the information cutoff gate d will be set to 0. Afterward, only the last closest information cutoff gate is labeled as $G_{i,d} = 1$. For example, consider a flight that initiated a go-around at 5.3 nm from the landing runway threshold. This flight sequence spans 10 nm to 6 nm, meaning that 6-

nautical-mile to 10-nautical-mile information cutoff gates will be considered for feature engineering. Information for gates between 1 nm and 5 nm will not be considered.

After data cleaning and matching, there were 100,032 arrivals at JFK airport during the analysis period. We identified and validated 371 go-arounds within the period, accounting for 0.371% of all JFK arrivals. All the features discussed in Chapter 4 are derived for every nautical mile from the landing runway threshold (10, 9, 8, …, 1 nm). Features are of two types – attributes and time series features – as defined in Section 4.4. In addition, continuous features are standardized to alleviate variation in magnitudes of the feature values and improve the convergence speed of the models.

In this chapter, we seek to investigate several methodologies for predicting the occurrence of a go-around at each timestamp of a landing aircraft sequence, using the realized trajectory data and its surrounding environment information available at different times of the approach. The go-around prediction is formulated as a transformation of multivariate sequences in the feature space into a sequence of go-around probabilities at each timestamp. The methodological framework, including data processing, modeling, and performance evaluation, is presented in Figure 23. In the remainder of this chapter, we will discuss three modeling approaches – classical machine learning models trained independently at different timestamps, a recurrent network model, and a Markovian model trained on sequential data. We then demonstrate the performance evaluation of the single-step-ahead prediction and the multi-step-ahead prediction and show the experimental results of the go-around prediction with a real-world dataset.

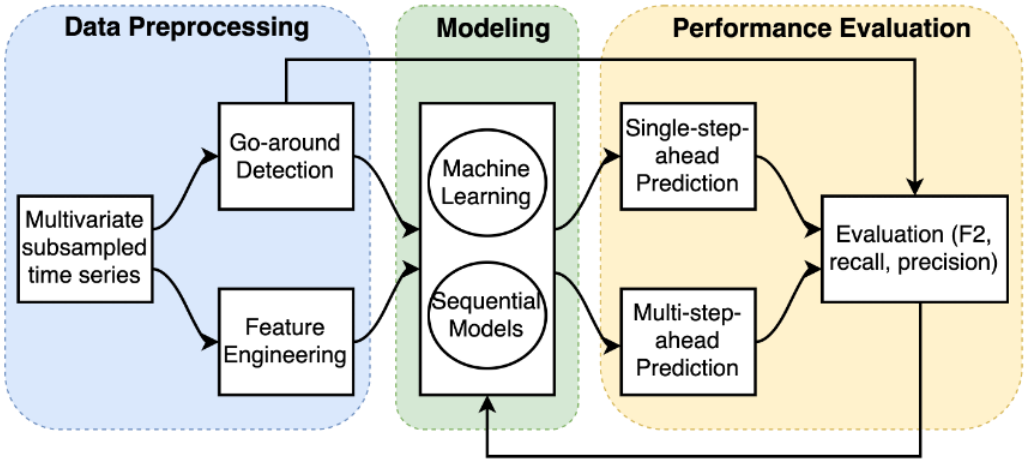

Figure 23. Predictive analytics framework.

7.4. Classical Machine Learning

One issue with using the classical machine learning models for the go-around sequential prediction task is that the training data consist of sequences of feature-response

pairs, which exhibit significant sequential correlation and do not quite fit the classical machine learning paradigm. Therefore, we employ the divide-and-conquer strategy and sliding window method to partition the overall sequential learning problem (i.e., predicting the go-around labeling sequence $\boldsymbol{G_i}$ of flight i given feature sequences $\boldsymbol{F_i}$) into subproblems (i.e., predicting individual output labels $G_{i,d}$ of flight i at the information cutoff gate d given subsets of information from $\boldsymbol{F_i}$).

Table 13. Tuning hyperparameters.

Model	Hyperparameter	Search Range
Logistic Regression	Penalty term for the l_2 regularization	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
SVM	Penalty term for the l_2 regularization	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
Random Forest	The maximal depth of the tree	[5, 10, 15, 20]
	The minimal number of samples required to split an internal node	[2, 5, 10, 15, 20]
	The number of features to consider when looking for the best split	[sqrt, log2]
XGBoost	Learning rate	[10^{-6},10^{-5},10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.3, 0.8]
	Minimum loss reduction required to make a further partition on a leaf node of the tree	[1, 3, 5, 10, 15, 20]
	Maximum depth of a tree	[6, 10, 15, 20]
	The maximum number of steps we allow each leaf output to be. It might help when class is extremely imbalanced	[2, 6, 10, 15, 20]
	l_2 regularization term on weights	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
	Subsample ratio of the training instances	[0.2, 0.4, 0.6, 0.8, 0.9, 1]
IO-HMM	The l2 regularized term in linear regression	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
	The l2 regularized term in logistic regression	[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]
	The number of hidden states	[2, 3, 4, 5, 6]

The sliding window method enables any classical machine learning algorithms to be applied to sequential prediction problems since it retains the information from previous

timestamps by stacking the feature spaces with a sliding window. It is predicated on the assumptions that (1) the training samples are drawn independently and identically from joint distributions $P(\boldsymbol{F}, G)$ specific to each information cutoff gate, and (2) only a fixed-sized window of features is relevant for predicting output values at each gate. In the current stage, we assume a sliding window size of *one*, in which the lag-one features from the last information cutoff gate will be included to predict $G_{i,d}$ given $\boldsymbol{F_{i,d}}, \boldsymbol{F_{i,d-1}}$. A longer-range interaction might not be necessary as the effects yielded long ago are mostly manifested in the flight status from the previous gate already. In future work, we will investigate the sliding window size by subsampling the flight sequences with a more refined frequency to have more information cutoff gates or by expanding the prediction horizons.

Following a similar modeling regime and experimental processes as [102], we train and compare four kinds of classical machine learning classifiers: logistic regression, kernelized support vector machine (SVM), random forest (RF), and extreme gradient boosting (XGBoost). Specifically, for each type of classical machine learning algorithm, we train eight models to predict $G_{i,d}$ for each information cutoff gate, given varying feature matrices that are known when the flight reaches each information cutoff gate. The gate-specific predictions are then concatenated to form the predicted go-around label sequence $\boldsymbol{G_i} = [G_{i,9}, G_{i,8}, G_{i,7}, G_{i,6}, G_{i,5}, G_{i,4}, G_{i,3}, G_{i,2}]$ for flight i. To balance bias and variance, we have fine-tuned the hyper-parameters for each model using five-fold cross-validation. Table 13 summarizes the descriptions of hyper-parameters and their tuning ranges.

7.5.Long Short-Term Memory

Although using classical machine learning algorithms is straightforward and elegant, it is not well-suited for the current setting because: (1) the training data comprises sequences of feature-label pairs $(\boldsymbol{F}, G)$ with temporal correlations that may not be effectively represented by simple stacking. (2) The classical machine learning approach is also computationally expensive since each type of classifier must be trained and fine-tuned for each distance-variant dataset. (3) In addition, the classical machine learning algorithms can only make single-step, single-output predictions one time step into the future based on the current conditions. Without the feature vectors for the following timestamp(s), we cannot obtain the probabilities of go-arounds further in the future.

In our use case, we want the model to provide a range of predicted probabilities of go-around occurrence in the future (e.g., at 5, 4, …, 1 nm), given a certain length of inputs (e.g., information at 10, 9, …, 6 nm is known). Unlike classical machine learning algorithms, which can only predict a single future value, the model needs to learn to predict a sequence of future values. Recurrent models, which can learn to make predictions in the future based on a lengthy history of inputs, are well-suited to our case. RNNs process a time series step-by-step, keeping an internal state from timestamp to timestamp. In this study, we will employ a variant of RNNs, called Long Short-Term Memory (LSTM), to map the multivariate input sequence to the output sequence. LSTM networks have been

shown to be an effective tool for learning representations from sequential data with temporal dependencies [103].

Since our flight sequences are of varying lengths, we first pad the samples to ensure that all the sequences are consistent in length and can be encoded into contiguous batches. A masking layer is then added to inform the model (i.e., the subsequent sequence-processing layers) what part of the input data is actually padding and should be skipped while processing the data or computing the loss. Under the hood, the masking layers will create a boolean tensor which will be propagated through the network for downstream layers. Each individual "False" entry specifies that the masked timestamp corresponding to it should be ignored during processing. Then, a two-layer LSTM network learns to map the given padded and masked input sequence to a sequence of hidden states that function as a summary/representation of the input sequence. The hidden states at each timestamp are learned by the LSTM network with weights and bias (parameters). Finally, we use the fully connected dense layer with sigmoid activation to transform the outputs from the LSTM layer to model predictions.

We use the Adam optimizer [104] with early stopping [105] to train our networks. All the parameters (weights and bias) of the network are learned by optimizing the binary focal cross entropy loss function [106] proposed by Facebook AI Research in 2018. For comparison, Equation (36) and (37) denote the commonly used binary cross entropy loss and the binary focal cross entropy loss, respectively, between the true go-around labels G and predictions $\hat{G}$.

$$L(G, \hat{G}) = -Gln(\hat{G}) - (1 - G)ln(1 - \hat{G}) \tag{36}$$

$$L(G, \hat{G}) = -G(1 - \hat{G})^{\gamma} ln(\hat{G}) - (1 - G)\hat{G}^{\gamma} ln(1 - \hat{G}) \tag{37}$$

Where $G \in \{0, 1\}$ is the binary class label obtained from the anomaly detection algorithm in Chapter 3; $\hat{G} \in (0, 1)$ is a probability estimate for the positive class (i.e., go-arounds). The focal loss function adds a factor of $(1 - \hat{G})^{\gamma}$ to the standard form of cross entropy loss. With the focusing parameter γ set to be positive, the focal cross entropy reduces the relative loss for well-classified samples (majority class), allowing the model to place a greater emphasis on difficult, misclassified samples. In other words, the conventional form of binary cross entropy loss requires the model to be confident in its predictions, while the focal cross entropy loss gives the model a bit more freedom to take some risks when making predictions. We currently set $\gamma = 2$ as recommended in the original paper on focal loss [106]. In addition, we weight the samples by the inverse of the class frequency for the class to which they belong, for penalizing the misclassification of the minority class by an amount proportionate to its underrepresentation. These two strategies are particularly useful in our case, where there is a highly imbalanced dataset.

7.6. Input-Output Hidden Markov Model

In this section, we develop the Markovian model to examine its ability of predicting go-around occurrence, as compared to the classical machine learning methods and the recurrent neural networks. The Markovian model relies on statistics and distributions, and therefore likelihood maximization. It is fundamentally different from the two methods that we presented in the previous sections – classical machine learning and recurrent neural networks, which do loss minimization.

For our use case, we employ a variant of the input-output hidden Markov model (IO-HMM), an extension to the HMM that can better capture the sequential structure inherent in our problem to model and predict the go-around occurrence for an approaching flight. Experiments on artificial tasks [51] have shown that IO-HMM, which uses EM recurrent learning, can deal with time dependencies more effectively than backpropagation through time and other alternative algorithms. It can be applied to achieve our goal of fully exploiting both input and output portions of the flight sequence data, as required by the go-around prediction task.

7.6.1. Model architecture

In order to deal with the go-around prediction problem, we regard the flight approach procedure as a discrete state dynamical process based on the following state-space description:

$$z_d = f(z_{d-1}, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}) \tag{38}$$

$$[\boldsymbol{x_d}, G_d] = g(z_d, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}) \tag{39}$$

Equation (38) describes the transition function between different states, where $z_d \in \{1, 2, \dots, s\}$ is a discrete hidden state variable that encodes symbols representing flight approach status (locations, speeds, etc.), s is the total number of hidden states, which is set *a priori* for IO-HMM. Researchers typically associate those hidden states with semantics. In our specific context, those hidden states those hidden states may be representations of how stable the flight approaches are, but the semantic interpretation is not critical; rather they provide a means of capturing patterns of evolution of the output variables.

$\boldsymbol{u_d}$ is the input variable at the information cutoff gate d, including contextual information that can be known before the aircraft reaches the information gate d, such as flight-specific characteristics (e.g., operated airline, aircraft type, landing runway), weather conditions (e.g., visibility, ceiling, wind speed), and airport information (e.g., airport arrival rate, runway configuration change). The unique advantage of the IO-HMM is that it incorporates the input vector, allowing contextual variables to affect transition probabilities and emission variables. In other words, flight sequences with similar input vectors—i.e., having similar contextual variables--will have similar estimated parameters, and thus a high probability of being "clustered" in the same latent state.

Equation (39) describes the emission function relating the output variables $[\boldsymbol{x_d}, G_d]$ and the state variables z_d. To be more specific, in the equation, $[\boldsymbol{x_d}, G_d]$ are both the output variables at the information cutoff gate d. $\boldsymbol{x_d}$ contains dynamic features described in Table 3, such as flight altitude and loss of separation. G_d is the go-around label obtained from the go-around detection algorithm in Chapter 3. According to Equation (39), the G_d is determined by the current state of the system z_d, input features at the current information cutoff gate $\boldsymbol{u_d}$and the lag-one output features in the previous information cutoff gate $\boldsymbol{x_{d-1}}$. The output variables are available only after the aircraft passes the information cutoff gate d.

In contrast to the input variables, the output variables contain information that is not available at the transition to a new approach state. In other words, output variables can be observed when training the models but must be inferred when we predict the go-around probability. As dynamic features from the previous information cutoff gates also contribute to part of the context information for predicting what will happen to the aircraft in the following information cutoff gates, we link the lag-one output variables $\boldsymbol{x_{d-1}}$ to the next information gate by incorporating $\boldsymbol{x_{d-1}}$ in the input vector layer.

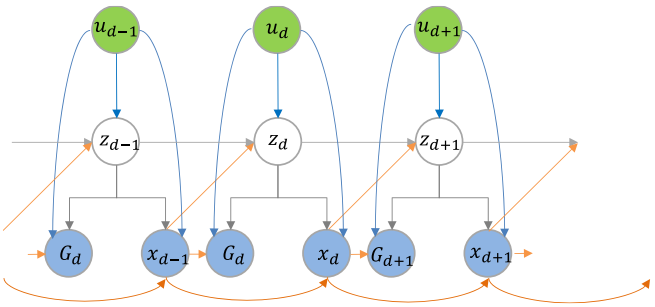

Figure 24. IO-HMM architecture.

Such discrete state dynamical system defined by Equation (38) (39) can be modeled by the graph depicted in Figure 25, in which the white nodes represent hidden state variables z_d, the green nodes represent observed input variables $\boldsymbol{u_d}$, and the blue nodes contain output variables $\boldsymbol{x_d}$ and G_d.

7.6.2. Model specification

We assume a multinomial distribution for the state variable z_d and use a Bayesian network to characterize the probabilistic dependencies among these state variables, input variables, and output variables. The IO-HMM captures the dynamics of the Markovian chain by using the current inputs and current state distribution to estimate the state variable and the output variable distributions for the next information cutoff gate. According to the connections shown in the IO-HMM graph shown in Figure 25, three probability models need to be specified: an initial probability model which outputs a vector of initial state probabilities at the start of the sequence, a transition probability model conditioned on the input sequence which outputs a square matrix of state transition probabilities at each

information cutoff gate, and an emission probability (a.k.a. output probability) model governing the distribution of the output variables – including go-around probability – at a particular time given the hidden state and input features. Model details and formulas are described in the following sections.

A. Initial model

We develop a multinomial logistic regression model to estimate the initial probability parameters $\widehat{\boldsymbol{\theta}}_{\boldsymbol{initial}}$.

$$P(z_1 = i \mid \boldsymbol{u_1}; \boldsymbol{\theta_{initial}}) = \frac{\exp\left(\theta_{initial_i} \cdot \boldsymbol{u_1}\right)}{\sum_k^s \exp\left(\theta_{initial_k} \cdot \boldsymbol{u_1}\right)} \tag{40}$$

where i is the state label at the initial ($d = 1$ represents the first information cutoff gate of the sequence, which is 9 nm before the landing threshold) information cutoff gate; s is the total number of hidden states.

B. Transition model

At the information gate d, the hidden state $\boldsymbol{z_d}$ is related to the input features $\boldsymbol{u_d}$, lag-one output features $\boldsymbol{x_{d-1}}$, and the previous hidden state $\boldsymbol{z_{d-1}}$, subject to some Gaussian noise. The multinomial logistic regression model is used to estimate the transition probability parameters $\widehat{\boldsymbol{\theta}}_{\boldsymbol{trans}}$.

$$P(z_d = j \mid z_{d-1} = i, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta_{trans}}) = \frac{\exp\left(\theta_{trans}{}_i^j \cdot\right)}{\sum_k^s \exp\left(\theta_{trans}{}_i^k \cdot \begin{bmatrix} \boldsymbol{u_d} \\ \boldsymbol{x_{d-1}} \end{bmatrix}\right)} \tag{41}$$

where i is the state label at the previous information cutoff gate $d-1$, j is the state label at the current information cutoff gate d; s is the total number of hidden states. The estimated $\widehat{\boldsymbol{\theta}}_{\boldsymbol{trans}} = \begin{bmatrix} \theta_{trans}{}_1^1 & \cdots & \theta_{trans}{}_1^s \\ \vdots & \ddots & \vdots \\ \theta_{trans}{}_s^1 & \cdots & \theta_{trans}{}_s^s \end{bmatrix}$, $\widehat{\boldsymbol{\theta}}_{\boldsymbol{trans}} \in \mathbb{R}^{s \times s}$ is the transition probability matrix for the approach state transited from the previous state. The transition probabilities are heterogeneous and depend on contextual information u_d. This can improve the accuracy of state inference.

C. Emission model

To gain interpretability, we choose linear models for continuous outputs represented as Gaussian random variables ($\boldsymbol{x_d}$), thus:

$$P(\boldsymbol{x_d}|\, z_d = j,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^c_{emis}}) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(- \frac{\left(\boldsymbol{x_d} - \boldsymbol{\theta^c_{emis_j}} \cdot \begin{bmatrix} \boldsymbol{u_d} \\ \boldsymbol{x_{d-1}} \end{bmatrix}\right)^2}{2\sigma_j^2} \right) \tag{42}$$

where j is the state label at the current information cutoff gate d. The estimated $\widehat{\boldsymbol{\theta}}^{\boldsymbol{c}}_{\boldsymbol{emis}} = \left[\widehat{\boldsymbol{\theta}}^{\boldsymbol{c}}_{\boldsymbol{emis_1}} \quad \cdots \quad \widehat{\boldsymbol{\theta}}^{\boldsymbol{c}}_{\boldsymbol{emis_s}}\right]$, $\widehat{\boldsymbol{\theta}}^{\boldsymbol{c}}_{\boldsymbol{emis}} \in \mathbb{R}^{m \times s}$ is the emission coefficient matrix where its column $\boldsymbol{\theta^c_{emis_j}}$ denotes the coefficients of m output variables in the linear model when the hidden state is j. σ_j represents the standard deviation of the linear model when the hidden state is j. Therefore, the number of coefficients to be estimated in the emission probability model is equal to the product of the number of hidden states s and the number of output variables m.

For binary random variable G_d, the logistic regression model is used as the output emission model. The probability is as follows:

$$P(G_d = 1|\, z_d = j,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^G_{emis}}) = \frac{1}{1 + \exp\left(-\boldsymbol{\theta^G_{emis_j}} \cdot \begin{bmatrix} \boldsymbol{u_d} \\ \boldsymbol{x_{d-1}} \end{bmatrix}\right)} \tag{43}$$

When implementing the maximum likelihood method, a Gaussian may be fit onto a single data point and lead to a singular covariance matrix. Whenever the covariance matrix is singular, the log-likelihood function will go to infinity. Thus, the maximization of the log-likelihood function is ill-posed. Ridge regularization [107] is employed to objective functions of both linear and logistic regression. The regularized term C_{ols}, C_{logit} will be fine-tuned from the range specified in Table 13 using five-fold cross-validation.

Based on the model architecture and model specifications, the likelihood of a sequence in our IO-HMM can be written as:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{u}) = \sum\nolimits_{s} [\, & P(z_1 \mid \boldsymbol{u_1}; \boldsymbol{\theta_{initial}}) \cdot \\ & \prod_{d=2}^{D} P(z_d \mid z_{d-1}, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta_{trans}}) \cdot \\ & \prod_{d=1}^{D} P(\boldsymbol{x_d}|\, z_d,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^c_{emis}}) \cdot \\ & P(G_d = 1|\, z_d,\, \boldsymbol{u_d}\,, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^G_{emis}})] \end{aligned} \tag{44}$$

The temporal dependency is captured by the transitions between the hidden approach states $\boldsymbol{z}$. The direct dependency between input features and output features can capture

relationships that are not fully mediated by the hidden state learned for the current timestamp.

7.6.3. Model estimation

As illustrated in the previous section, the IO-HMM includes three groups of unknown parameters to be estimated: initial probability parameters $\boldsymbol{\theta_{initial}}$, transition probability parameters $\boldsymbol{\theta_{trans}}$, and emission probability parameters $\boldsymbol{\theta_{emis}}$. In this study, we implement the Expectation-Maximization (EM) algorithm to optimize the parameter set. Explicitly, in the E-step, we compute the expected value of the complete log-likelihood as in Equation (45), given the observed data and parameters estimated (or initialized) at the previous (or initialization) step. In the M-step, parameters are updated to maximize the expected data log-likelihood.

$$\begin{aligned} Q(\boldsymbol{\theta},\, \boldsymbol{\theta^r}) = & \sum_{i=1}^{s} \gamma_{i,1} ln\, P(z_1 = i \mid \boldsymbol{u_1}; \boldsymbol{\theta_{initial}}) + \\ & \sum_{d=2}^{D} \sum_{i=1}^{s} \sum_{j=1}^{s} \xi_{ij,d} \ln \mathrm{P}(z_d = j \mid z_{d-1} = i, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta_{trans}}) \\ & + \sum_{d=1}^{D} \sum_{i=1}^{s} \gamma_{i,d} \left[\ln P(\boldsymbol{x_d} \mid z_d = j,\, \boldsymbol{u_d}\, , \boldsymbol{x_{d-1}}; \boldsymbol{\theta^c_{emis}}) \right. \\ & \left. + \ln P(G_d = 1 \mid z_d = j,\, \boldsymbol{u_d}\, , \boldsymbol{x_{d-1}}; \boldsymbol{\theta^G_{emis}}) \right] \end{aligned} \tag{45}$$

where r represents the iteration; s is the total number of hidden states; D is the total number of information cutoff gates in each flight sequence; $\boldsymbol{u_d}, \boldsymbol{x_{d-1}}, \boldsymbol{x_d}$, G_d, and z_d are the input variables, lag-one output variables, output variables, go-around binary labels and hidden state variables at information cutoff gate d, which were introduced in Section 7.6.1; $\boldsymbol{\theta_{initial}}, \boldsymbol{\theta_{trans}}, \boldsymbol{\theta^c_{emis}}, \boldsymbol{\theta^G_{emis}}$ are parameters to be estimated in initial probability model, transition probability model, and emission probability model which were discussed in Section 7.6.2.

$\xi_{ij,d}$ is the posterior transition probability, which defines the probability of being in state i at the information cutoff gate d and state j at the information cutoff gate $d+1$. $\gamma_{i,d}$ is the posterior state probability for state i at the information cutoff gate d. $\xi_{ij,d}$ and $\gamma_{i,d}$ are computed from forward probability α and backward probability β under the forward-backward algorithm [108], which involves three steps:

- Computing the forward probability which provides the probability of ending up in any particular state i given the first d observations in the sequence;
- Computing the backward probability which provides the probability of seeing the observations from information cutoff gate $d+1$ to the end given we are in a particular state at the time;

- These two sets of probabilistic distributions can then be combined to obtain the distribution over states at any specific point $d \in \{1, \cdots, D\}$ given the entire observations sequence.

$$\xi_{ij,d} = P(z_d = i \mid z_{d-1} = j, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta_{trans}}) \cdot \alpha_{i,d} \cdot \beta_{j,d} \cdot P(\boldsymbol{x_d} \mid z_d = j, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^c_{emis}}) \cdot P(G_d = 1 \mid z_d = k, \boldsymbol{u_d}, \boldsymbol{x_{d-1}}; \boldsymbol{\theta^G_{emis}}) / \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{u}) \quad (46)$$

$$\gamma_{i,d} = \frac{\alpha_{i,d}\beta_{i,d}}{\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{u})} \quad (47)$$

The whole estimation and inference process is summarized in Figure 25. The forward-backward algorithm starts with some initial estimates of the IO-HMM parameters $\boldsymbol{\theta_0}$. We then iteratively run the Expectation-step and the Maximization-step. In the E-step, we compute the expected value of the complete data log-likelihood, the posterior state probability $\gamma_{i,d}$, and the posterior transition probability $\xi_{ij,d}$ for each training sequence, given the entire observed data sequence and parameters estimated at the previous (or initialized) step. In the M-step, we use the computed distribution over states to update the transition probability matrix and the emission likelihood matrix for maximizing the expected data likelihood.

During the inference time, we apply the estimated parameters on unseen data in the test set. It is intuitive to imagine the inference process in this way: as an aircraft approaches to its landing runway, it will experience different states during the approach procedure. Before the flight reaches the next nautical-mile point, we can obtain context information u_t such as aircraft type, weather conditions in advance. Given the input features available at the moment, the heterogeneous transition probability matrix dependent on input features is calculated. We can thus infer which state the flight will be transited to in the next 1-nautical-mile according to the *highest posterior state probability* among all candidate states. Based on the input layer and the hidden state layer, all the variables in the output layer can be predicted via $\theta^{predicted}_{emission}$ of the selected state, including the go-around probability. During flight approach procedure, the relevant context information u_t will be updated, and the next approach state will be selected given the newly obtained transition probability matrix. This process continues until the full sequence of approached states has been inferred. Under this mechanism, the sequential probability of go-arounds can be predicted for any flight approach procedures given known features prior to a certain time.

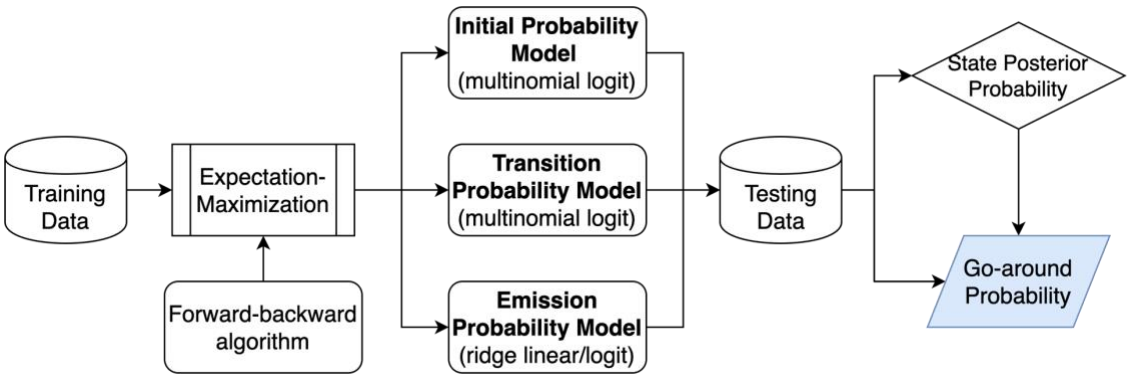

Figure 25. IO-HMM model estimation and inference.

7.7. Model inference

The goal of this section is to predict the dynamic probability of go-arounds in the output layer as the flight approaches its landing runway, given the known input features at a certain time prior to the go-around. Once the models are trained, we will use them to generate predictions against previously unseen data and evaluate the model performance. With pre-trained models, the inference process can be completed in constant time for each flight. Models can also be re-trained, if needed, with newly available observations.

The simplest way is just to predict a single value (one step ahead) in the future, and no feedback is used to continue the prediction. We call this the single-step-ahead prediction. The feature space must be updated with the most recent information in order to predict the values in the next step. Figure 26 shows a simplified version of the single-step-ahead prediction procedure when the flight passes the 6 nm information cutoff gate (i.e., the flight is 6 nm away from the landing runway threshold). The feature vectors are available at 10, 9, …, 6 nm ($\boldsymbol{F_{10}}$, $\boldsymbol{F_9}$, $\boldsymbol{F_8}$, $\boldsymbol{F_7}$, $\boldsymbol{F_6}$). The model is trying to predict the go-around probability in the next gate. The classical machine learning models (left figure) only utilize the information at the current timestamp and the lag-one step to make a prediction. the models trained for every timestamp are completely independent of one another. Without knowing the features at 5 nm $\boldsymbol{F_5}$, we cannot produce predictions at 4 nm, and hence cannot predict the probability of go-arounds before the flight passes the 5 nm gate, so on for the succeeding timestamps.

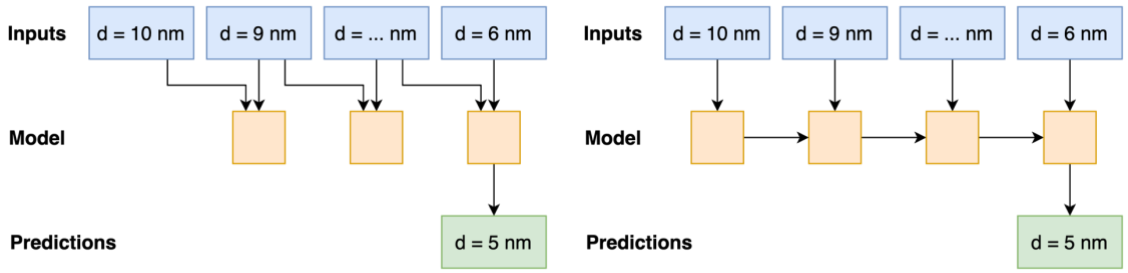

Figure 26. Single-step-ahead prediction with classical machine learning (left) and sequential models (right).

The other prediction procedure is termed multi-step-ahead prediction, which provides the outcomes of the rest of the steps or multiple steps ahead in a single shot or in an autoregressive way. We demonstrate the difference between these two approaches in Figure 27 and Figure 28 below using an example of predictions made when the flight just passes the 6 nm information cutoff gate. For multi-step-ahead prediction, we are trying to answer the questions like: if we know all required information (features) at 10, 9, …, 6 nm gates, what are the probabilities of go-around occurrence at 5, 4, …, 1 nm will be? After accumulating the internal state for 10 to 6 nm, the LSTM networks produce the output sequence for the remaining information cutoff gates in a single shot. The IO-HMM makes autoregressive predictions, in which the model makes single-step predictions of both input and output sequences, and then feeds them forward as input to make further predictions conditioned on the previous one for the required number of output timestamps.

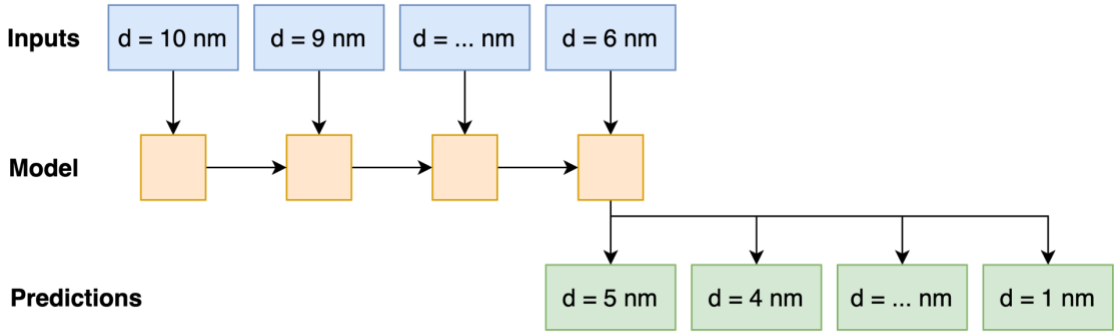

Figure 27. Multi-step-ahead prediction in a single short.

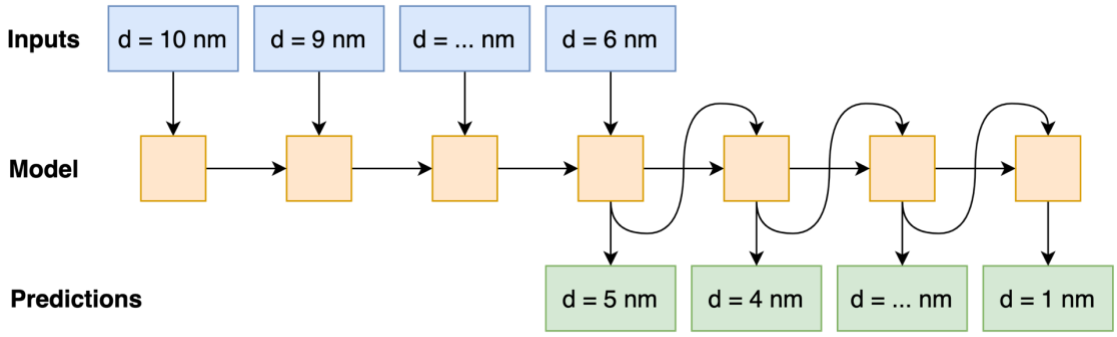

Figure 28. Multi-step-ahead prediction with autoregression.

Assuming that we are able to constantly update the observed information at the previous *1 nm* in real time, each time the flight reaches a certain gate d, we will acquire a sequence of predictions for the subsequent steps $d-1, d-2, \ldots, 1$, $\widehat{\boldsymbol{G}}^{\boldsymbol{d}} = \left[\hat{G}^{d}_{d-1}, \hat{G}^{d}_{d-2}, \ldots, \hat{G}^{d}_{1}\right]$. Note that we will obtain another new sequence of go-around predictions once the flight passes the next gate, and more information becomes available. For example, after the flight passes the 5-nm gate, we obtain the sequence of go-around predictions $\widehat{\boldsymbol{G}}^{\boldsymbol{5}} = \left[\hat{G}^{5}_{4}, \hat{G}^{5}_{3}, \hat{G}^{5}_{2}, \hat{G}^{5}_{1}\right]$. A new sequence of go-around predictions $\widehat{\boldsymbol{G}}^{\boldsymbol{4}} = \left[\hat{G}^{4}_{3}, \hat{G}^{4}_{2}, \hat{G}^{4}_{1}\right]$ is produced once the flight passes the next 4 nm gate. To facilitate model comparison and practical use, we adopt the complement rule of probability to transform a sequence of go-around probabilities at a certain gate d, $\widehat{\boldsymbol{G}}^{\boldsymbol{d}}$, into a single probability $\hat{G}^{d_{multi}}$:

$$\hat{G}^{d_{multi}} = 1 - \prod_{r=1}^{d-1}\left(1 - \hat{G}^{d}_{r}\right) \tag{48}$$

As a result, every time the flight passes through a particular information cutoff gate d, we obtain a single value prediction $\hat{G}^{d_{single}}$ from the single-step-ahead prediction, and another single value prediction $\hat{G}^{d_{multi}}$ converted from a sequence of predicted go-around probabilities using multi-step-ahead prediction and Equation (48). The single-step prediction $\hat{G}^{d_{single}}$ represents the probability of go-around prior to the next nautical mile, while the multi-step prediction $\hat{G}^{d_{multi}}$ can represent the probability of go-around prior to landing. These two go-around probabilities will be updated accordingly and can be transmitted to pilots or controllers as a signal or alert for go-around occurrence during the approach and landing phase.

7.8. Experimental Steps

The three methods – classical machine learning, LSTM, and IO-HMM – were applied to a historical dataset that contains 371 go-arounds out of 100,032 arrival flights at JFK airport in the second half-year of 2018. We first split the data into a training set (80%) and a testing set (20%), respecting to the class distribution. Then, we train, validate, and test

the aforementioned models on these datasets. The experimental steps and model selection process follow a similar regime in [102], including data processing, parameter tuning, and performance evaluation with statistical hypothesis tests. For classical machine learning, each kind of model is trained eight times separately over different distant-specific datasets, using only features up to the present information cutoff gate. For the sequential models, the LSTM and IO-HMM are trained once for all the flight sequences.

In addition, we observe that the dataset is extremely imbalanced, with go-around flights significantly less than those non-go-around flights - only 0.3% of arrivals are go-around flights. With so few go-arounds relative to non-go-arounds, the learning algorithm often struggles to generalize the behavior of the go-around flights well, and tends to classify all observations as majority class; hence the algorithm performs poorly on the minority class. Therefore, the loss functions of all the above models are modified to deal with the class imbalance issues – penalizing the misclassification of the minority class by an amount proportional to how under-represented it is.

We report five metrics to evaluate the performance of the models: F2 score, the area under the receiver operating characteristic curve (AUC), precision, recall, and accuracy. F-score, which is the weighted harmonic mean of the precision and recall, is chosen to be the evaluation criteria through the experiments. Based on the accuracy measures in the confusion matrix [109], $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$ are defined (TP: true positive, FP: false positive, TN: true negative, FN: false negative). We want to have high confidence that observations predicted as go-arounds are actually correct (high precision), as well as a high detection rate of the go-arounds (high recall). However, high precision comes at the cost of the low recall and vice versa. Referring to [29] that the cost of not detecting a go-around highly outweighs the cost of getting a false alarm warning in the system, we set the parameter, which determines the weight of recall in the F-score, as $\boldsymbol{2}$. In the inference process, we empirically select the threshold that maximizes the F2 score on the test set.

$$F_2 = \frac{(1+2^2)\cdot precision \cdot recall}{2^2 \cdot precision + recall} \tag{49}$$

7.9.Model Performance

Figure 29 shows the F2 score at each timestamp for the four classical machine learning algorithms and the two sequential models of LSTM and IO-HMM. Table 14 reports the five metrics obtained on the testing set for these different models. We include the single-step prediction of the two sequential models in comparison with the classical machine learning models. The corresponding metrics scores are comparable across different models, for a specific information cutoff gate, in the same dataset.

The sequential models consistently outperform the classical machine learning models except for the early stage of the flight approach. This suggests that incorporating temporal structures in the model can improve go-around prediction performance. With multi-step predictions, the IO-HMM slightly improves over the single-step predictions for every distance-variant dataset in terms of F2 score, precision, recall, and AUC, while the LSTM improves significantly. As for the classical machine learning models, the random forest model is comparable to the IO-HMM, yet it still slightly falls behind the IO-HMM, especially during the later stage of a flight approach process.

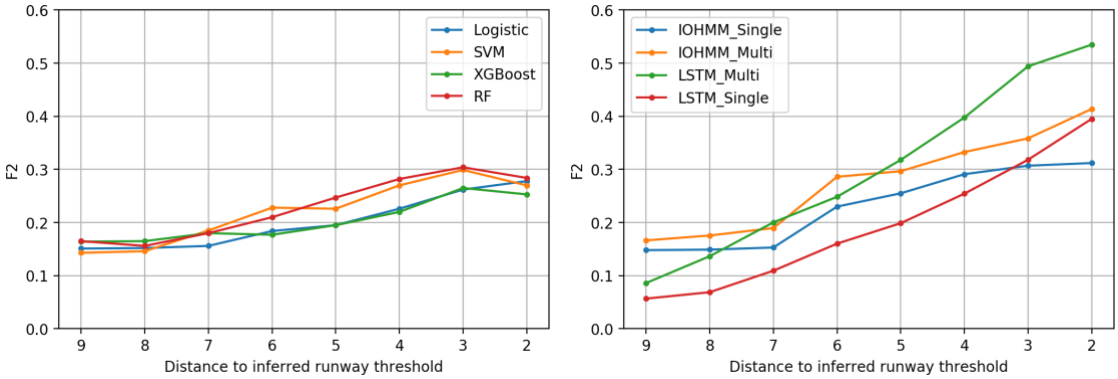

Figure 29. Model performance, in terms of F2 score, of classical machine learning models (left) and sequential models (right).

We also notice that sequential models (LSTM, IO-HMM) exhibit a consistent and monotonically increasing performance, in terms of F2 score, as the approach progresses, while other models do not. The prediction performance improves as the flight gets closer to the airport. The main reason for this may be that most go-arounds happen closer to the airport, and these sequential models can maintain more information in their internal state as the given input sequence gets longer (flights getting closer to the airport). In other words, more information in the temporal sequences is preserved, and the feature space used for multi-step predictions becomes more reliable. The classical machine learning models are trained separately using independent input features, and are thus incapable of learning a whole flight sequence that inherently captures the temporal dependency. For the LSTM, the most confident prediction is achieved when the flight is at 2 nm before the runway threshold, with a recall score of 0.59 and a precision score of 0.39, suggesting that 59% of go-arounds are correctly identified, and 39% of the go-arounds predicted by the model are correct. LSTM is capable of automatically extracting features from past events and is able to apply point-wise nonlinearities to the output at every timestamp, which Markovian models do not. This is why the LSTM model is more expressive and adaptive to learning from real-world data.

7.10. Chapter Summary

In this chapter, we have shown how go-around prediction can be addressed as a multivariate sequential prediction problem. We view it as a sequence classification problem and compare different learning algorithms with two prediction strategies. A set of supervised learning models serve as the benchmarks of the go-around prediction task. The experiments were also carried out with the IO-HMM designed to evolve model estimates (learned weights) through discrete state space. It was found that the both the LSTM and IO-HMM can capture the dependency structure in the flight sequence, and the multi-step prediction strategy achieves better prediction performance. LSTM is capable of automatically extracting features from past events and is able to apply point-wise nonlinearities to the output at every timestamp, which Markovian models do not. This is why the LSTM model is more expressive and adaptive to learn from real-world data. Further research performed with more real-world datasets and additional sequential models is required to generalize and improve the results. A major challenge of go-around prediction is the imbalanced class distribution of the data. The go-around rate is (thankfully) very low – FAA reports that the average go-around occurrence across the core 30 airports in the US from 2012 to 2018 is at a rate of 0.4% [25]—but the resulting class imbalance in go-around data sets makes predictive modelling difficult. In the next chapter, we will investigate different techniques to address the class imbalance issue.

Aside from prediction, another product of the IO-HMM model is the pattern recognition capability. The hidden state variables $\boldsymbol{z}$ inferred from data are categorical labels corresponding to unobserved activity patterns, or flight approach types, or other semantic meanings that can be associated to it following an in-depth analysis. First, a set of decision rules based on the spatial-temporal representations of features (e.g., speed/altitude profile) need to be designed to identify the latent semantics of each of the hidden state variables. Second, we would need to compare the annotated activity patterns with additional data sources (e.g., ground truth or simulated approach patterns identified by domain experts via manual inspection), that are independent of the in-use dataset, to validate the pattern recognition results and the activity transition chains. Although it may prove to be difficult in light of the highly specialized nature of the aviation domain (as compared, for example, with urban activity modeling), the pattern recognition capability can help "annotate" flight activity types (in the approach phase, for this work), which may simplify scenario evaluation for aviation operators, and thus improve the safety and efficiency of air traffic control.

Table 14. Go-around prediction model performance.

After the flight passes gate	Model	F2	Precision	Recall	AUC	Accuracy
	Logistic	0.151	0.064	0.227	0.848	0.985
	SVM	0.143	0.062	0.213	0.829	0.985
	XGBoost	0.164	0.085	0.213	0.860	0.988
	RF	0.165	0.073	0.240	0.841	0.986
9	IO-HMM (single)	0.148	0.067	0.213	0.777	0.978
	LSTM (single)	0.057	0.013	0.321	0.826	0.953
	IO-HMM (multi)	0.166	0.049	0.410	0.817	0.968
	LSTM (multi)	0.086	0.028	0.179	0.780	0.973
	Logistic	0.152	0.167	0.149	0.843	0.994
	SVM	0.146	0.056	0.243	0.814	0.982
	XGBoost	0.165	0.047	0.446	0.846	0.964
	RF	0.156	0.054	0.297	0.846	0.978
8	IO-HMM (single)	0.149	0.062	0.230	0.711	0.984
	LSTM (single)	0.069	0.022	0.146	0.820	0.985
	IO-HMM (multi)	0.176	0.070	0.282	0.815	0.983
	LSTM (multi)	0.137	0.058	0.205	0.824	0.985
	Logistic	0.156	0.071	0.222	0.857	0.987
	SVM	0.185	0.066	0.333	0.828	0.981
	XGBoost	0.180	0.063	0.333	0.856	0.980
	RF	0.180	0.052	0.472	0.864	0.967
7	IO-HMM (single)	0.153	0.077	0.203	0.728	0.988
	LSTM (single)	0.109	0.031	0.308	0.803	0.961
	IO-HMM (multi)	0.190	0.216	0.188	0.903	0.999
	LSTM (multi)	0.200	0.158	0.215	0.897	0.995
	Logistic	0.184	0.063	0.352	0.862	0.979
	SVM	0.228	0.091	0.366	0.854	0.985
6	XGBoost	0.177	0.068	0.296	0.851	0.983
	RF	0.210	0.091	0.310	0.860	0.987

After the flight passes gate	Model	F2	Precision	Recall	AUC	Accuracy
	IO-HMM (single)	0.230	0.092	0.368	0.808	0.985
	LSTM (single)	0.160	0.064	0.256	0.845	0.983
	IO-HMM (multi)	0.286	0.142	0.385	0.867	0.989
	LSTM (multi)	0.248	0.096	0.410	0.882	0.983
5	Logistic	0.195	0.065	0.391	0.862	0.978
	SVM	0.226	0.146	0.261	0.855	0.992
	XGBoost	0.195	0.073	0.333	0.863	0.983
	RF	0.247	0.122	0.333	0.856	0.989
	IO-HMM (single)	0.255	0.100	0.417	0.828	0.986
	LSTM (single)	0.199	0.062	0.449	0.908	0.981
	IO-HMM (multi)	0.297	0.155	0.385	0.926	0.990
	LSTM (multi)	0.318	0.089	0.897	0.947	0.965
4	Logistic	0.226	0.080	0.413	0.891	0.983
	SVM	0.270	0.153	0.333	0.884	0.992
	XGBoost	0.220	0.088	0.349	0.874	0.987
	RF	0.282	0.131	0.397	0.854	0.990
	IO-HMM (single)	0.291	0.117	0.463	0.836	0.989
	LSTM (single)	0.254	0.118	0.359	0.947	0.987
	IO-HMM (multi)	0.333	0.097	0.846	0.925	0.970
	LSTM (multi)	0.398	0.257	0.462	0.964	0.993
3	Logistic	0.262	0.127	0.357	0.861	0.991
	SVM	0.299	0.129	0.446	0.862	0.990
	XGBoost	0.265	0.205	0.286	0.874	0.995
	RF	0.304	0.410	0.286	0.877	0.997
	IO-HMM (single)	0.307	0.141	0.435	0.839	0.990
	LSTM (single)	0.318	0.089	0.897	0.947	0.965
	IO-HMM (multi)	0.358	0.239	0.410	0.938	0.993

After the flight passes gate	Model	F2	Precision	Recall	AUC	Accuracy
	LSTM (multi)	0.494	0.295	0.590	0.957	0.993
2	Logistic	0.278	0.115	0.429	0.873	0.989
	SVM	0.270	0.102	0.457	0.878	0.989
	XGBoost	0.253	0.151	0.304	0.867	0.994
	RF	0.284	0.126	0.413	0.848	0.992
	IO-HMM (single)	0.312	0.155	0.417	0.861	0.990
	LSTM (single)	0.395	0.191	0.538	0.944	0.990
	IO-HMM (multi)	0.414	0.214	0.538	0.924	0.991
	LSTM (multi)	0.535	0.390	0.590	0.930	0.995

8. Imbalanced Learning

8.1. Overview

A major challenge of the go-around prediction problem, or for any rare event prediction, is the imbalanced class distribution of the data. The capability to deal with imbalanced data holds significant promise for the applicability of AI, as this is the crux of many real-world problems, especially in the safety-related fields. The go-around prediction task is a canonical example of imbalanced data problems, which often results in increased difficulty in classification as many learning algorithms are best suited for a balanced dataset. The go-around rate is often very low, with only 0.3% of arrivals being go-arounds in our case. The ensuing class imbalance would significantly degrade the performance of any learning model, specifically with regard to the minority class we are interested in. With so few go-arounds relative to non-go-arounds, the learning algorithm is often unable to generalize the behavior of the go-around flights well, and thus tends to classify minority class observations as majority class.

In the models presented in Chapter 7, we have modified the loss function to penalize the misclassifications of minority class by an amount proportional to how under-represented it is, and picked the performance metric in a way that emphasizes the recall score of the minority class (details are in Section 7.8). The other approach to mitigate the class imbalance problem is to augment the data samples for the minority class, either by resampling the data or generating synthetic samples in the training set. The change to the class distribution is only applied to the training set, intending to influence the model fit (estimated/learned parameters). This motivates this part of thesis research to explore several strategies for minority class augmentation in order to tackle the issue of imbalanced learning. While the minority class augmentation will be employed specifically for go-arounds in this study, it is a promising technique for many other transport (and non-transport) safety threats in which sequential data is employed to identify the risk of non-nominal event occurrence.

The overarching question for the work in this chapter is: can we generate high-fidelity synthetic go-around sequences to augment the minority class, and employ these synthetic sequences to train models that are better at predicting go-around occurrence? Following a review of existing methods for tackling this problem in Section 8.2, we develop a GAN model (Section 8.5) that can satisfy the requirements of generating multivariate sequence data with variable length and mixed data types. Furthermore, we benchmark this model against the simple downsampling method (Section 8.3) and the autoencoder method with ADASYN (Section 8.4). The performance of the go-around prediction model is also compared with various proportions of synthetic go-arounds added to the training data.

8.2. Related Work

The two main approaches to augmenting the minority class are to delete samples from the majority class, referred to as downsampling, and add more minority samples, referred to as oversampling. The random sampling method takes the original training set and resamples it to obtain a more balanced distribution of the minority and majority classes. While this simple strategy changes the *data sizes* of the minority and majority samples, generating synthetic samples based on the original training data could further increase the *data variety* and might lead to better model performance. In our case, sequential data generation is typically more challenging than non-sequential data generation, since sequential data has time dependencies. Furthermore, our data is multivariate, with multiple features at each time point. This adds additional complexity to synthetic data generation as we not only need to generate reasonable values for features at different timestamps, but also need to reproduce similar sequence lengths, temporal dependencies, and feature correlations.

There are three common ways to generate synthetic data – simulation approach, sampling-based approach and deep generative models. The simulation approach generates data samples by developing a simulator that replicates the behavior of a real system and event occurrence. For example, Total Airspace and Airport Modeler (TAAM) is a widely used fast-time, gate-to-gate simulation tool for analyzing aviation operations both in the airspace and on the airport surface. If the simulator is highly similar to real systems, the simulation approach has a high degree of fidelity. In practice, however, configuring the settings to simulate a specific target dataset is often difficult and expensive. While several data-driven approaches to parameter configuration have been presented recently [110, 111, 112], it remains challenging to ensure that the simulator generalizes all possible scenarios.

The sampling-based methods are straightforward, but the situations they can be applied are limited. SMOTE (Synthetic Minority Oversampling TEchnique) [113] is a frequently used technique for oversampling minority samples by interpolating between their nearest neighbors. However, it does not account for the sequential dimension. One extension to SMOTE is the ADASYN (ADAptive SYNthetic) sampling algorithm [114], which investigates the composition of the nearest neighbors and adaptively shifts the classification decision boundary toward the minority samples. Neither SMOTE nor ADASYN captures the sequential correlation in time because they generate synthetic feature vectors by interpolating between the real data points at each timestamp independently. Recently, autoencoders have been utilized to learn representations of the input sequence, allowing for the application of these sampling methods in the latent space to capture the structure of the sequence [115].

Another approach, generative adversarial networks (GANs), have been widely used to oversample image data [116, 117, 118, 119, 120], and has recently become an active research topic for oversampling sequence data [121, 122, 123, 124, 125]. We consider leveraging recent advances in GANs to generate synthetic go-around sequences. In comparison to the preceding two approaches, GANs, when appropriately developed,

provide three significant advantages. First, the discriminator acts as a universal agent responsible for accessing the authenticity of generated samples. Thus, the discriminator requires only the realistic samples and no further information about the system producing the samples. Second, GANs can be used to generate both static and time series features with mixed data types, as well as their cross-correlations. Finally, GANs are capable of capturing the complicated structure of the data, as evidenced by their applications in the generation of images [126, 127], texts [125, 128, 129], and music [130, 131].

However, canonical GANs often perform badly when it comes to extracting temporal dependencies and the correlations among multiple features with mixed data types. Additionally, Mode collapse [62] is a well-known issue in GANs where they generate only a few modes from the underlying distribution. It is compounded further in time series applications, such as when we generate go-around sequences, because of the wide range of feature values. In this study, we will synthesize domain-specific insights with concurrent advances in the GAN literature to design a proof-of-concept GAN architecture for our go-around use case. RCGAN [121], TimeGAN [122] and DoppelGANger (DG) [123] are the three most relevant studies we refer to. RCGAN generates time series using RNNs, but does not evaluate the correlations among features. As with RCGAN, TimeGAN uses RNNs for both the generator and discriminator. However, TimeGAN further trains an additional neural network that maps time series to vector embeddings, such that the generator produces sequences of embeddings rather than original features. It is usual to learn to generate transformed or embedded time series, both using GANs and using a different class of generative models such as variational autoencoders (VAE) [132]. An extension of the TimeGAN, DG is proposed to deal with network data generation. The innovative part of this GAN architecture is that it introduces an auxiliary discriminator for the generation of static features (i.e., attributes), which are then conditioned by the generation of time series features.

8.3. Downsampling

Considering the size of the data and computational complexity, we choose to downsample the majority class in the training set before learning a classifier. The assumption behind this strategy is that there are many redundant observations in the majority class, and that randomly removing some of them does not affect the estimation of the within-class distribution [133].

Specifically, the downsampling will be utilized as a pre-processing step to rebalance the two classes before any algorithm is applied. For each distance-variant dataset, we first split the data into a training set (80%) and a testing set (20%), corresponding to the second and third sub-columns in Table 15. Second, while the number of go-arounds remains unchanged in both the training set and the testing set, we randomly sample the non-go-around flights without replacement until the ratio of non-go-arounds to go-arounds ratio is 10 to 1 (269:1 in the full dataset). We do not set the two classes equally balanced; otherwise,

models would be prone to overfitting and be much more sensitive to outliers owing to the small number of data samples. After merging the downsampled majority class with the original minority class, we obtain the balanced dataset, reported as the fourth and fifth sub-columns in Table 15. We train, validate, and test the aforementioned models on the full dataset and the downsampled dataset separately. We also apply the same treatment to deal with these two imbalanced datasets – penalizing the misclassification of the minority class by an amount proportional to how under-represented it is.

Table 15. The size of the full dataset and the downsampled dataset.

Dist. (nm)	**Full dataset (G-A / arrivals)**		**Downsampled set (G-A / arrivals)**	
	Train (80%)	***Test (20%)***	***Train***	***Test***
9	296 / 80,025	75 / 20,007	296 / 3,256	75 / 825
8	293 / 80,022	74 / 20,006	293 / 3,223	74 / 814
7	287 / 80,016	72 / 20,004	287 / 3,157	72 / 792
6	281 / 80,010	71 / 20,003	281 / 3,091	71 / 781
5	273 / 80,002	69 / 20,001	273 / 3,003	69 / 759
4	252 / 79,981	63 / 19,995	252 / 2,772	63 / 693
3	221 / 79,950	56 / 19,988	221 / 2,431	56 / 616
2	180 / 79,909	46 / 19,978	180 / 1,980	46 / 506

With a less imbalanced dataset, all the models have better predictability regarding F2 score, precision, recall, and AUC. Nevertheless, no universal model performs best for all distance-variant datasets. The performance of IO-HMM is poorer, probably due to the small size of the training data. Other research work [134] has found that the larger the training data, the more accurate the estimation of the probability models (especially for the transition matrix) of the IO-HMM, and the smaller the bias in the Markovian predicted residuals. Even though IO-HMM learns temporal structures, the small training data makes it incapable of learning a hidden state effective for go-around prediction.

8.4. Sampling-Based Augmentation

As discussed in Section 8.2, the sampling-based approach (e.g., SMOTE, ADASYN) does not capture the sequential correlation in time because they generate synthetic feature vectors by interpolating between the real data points at each timestamp independently. In order to use the sampling-based approaches, we implement an extended ADASYN with autoencoder to capture the representation of multi-dimensional go-around sequences, such that the oversampling is done in a lower dimension space.

The original go-around data is first used to train an autoencoder. After the data representation is learned in a lower dimension space, the ADASYN algorithm is run to generate more (synthetic) go-around samples in the lower dimensional space. Finally, the

decoder is used to lift the generated samples back to the dimensional space of the original data. In this way, the encoded minority data captures the structure of the sequence. This sampling-based augmentation method serves as a baseline for the GAN-based augmentation that we will discuss in the next section. We found that the GAN-based augmentation is superior to the sampling-based augmentation in terms of the fidelity of generated samples. We thus do not retrain predictors on these sampling-based augmented samples.

8.5. Generative Adversarial Network

8.5.1. Problem formulation

For our go-around use case, the goal is to develop a GAN model capable of generating variable-length, multi-dimensional sequences with mixed data types. Following the notation from Chapter 7, the GAN takes part of the original dataset $\mathcal{D}$ as input and learns a model (generator) that can generate a new dataset $\mathcal{D}'$ as output. The synthetic dataset $\mathcal{D}'$ preserves trends and feature distributions as the original data with enough fidelity that a predictor trained on synthetic data $\mathcal{D}'$ can still make meaningful predictions on real data (in the test set). Each sample i in the input dataset $\mathcal{D}$, is a multi-dimensional go-around sequence with label $G = 1$, and features $\boldsymbol{F_d}$ representing the feature vectors available/known after the flight passes the information cutoff gate d, including both the static features $\boldsymbol{u}$ and dynamic features $\boldsymbol{x_d}$.

8.5.2. Model architecture

The primary difficulties encountered when applying existing GAN architectures to our go-around use case are: the complex correlations among multiple features with mixed data types, the temporal dependencies within sequences that are not present in images, and the variable sequence lengths associated with the abnormality of the multivariate distributions. While these difficulties specifically stem from our attempts in using GANs to generate go-around sequences, they are broadly applicable to other use cases as well.

To tackle these challenges, we synthesize domain-specific insights with concurrent advances in the GAN literature [121, 122, 123] to design a proof-of-concept GAN architecture shown at the bottom of Figure 30. We also present the conventional GAN architecture on the top to illustrate how our model architecture compares to it. Empirically, we find that this architecture improves the fidelity of our generated go-around data samples significantly.

First, to better capture the correlations among multi-dimensional features with mixed data types (i.e., attributes and time series features), we decouple the generation of attributes (e.g., aircraft type, airline) and time series features (e.g., groundspeed, altitude), each using a dedicated generator. In our go-around use case, attributes can strongly influence the time series features. For example, aircraft of different types would have different approach

speed/altitude profiles; flight approaches also vary under different visibility conditions. Therefore, we need a mechanism to model the joint distribution between dynamic time series features and static attributes. We found that the typical strategy of concurrently generating attributes and time series features using a single generator is ineffective in learning the correlations between these two types of data. Several studies have addressed this issue by training a variant of GANs, called conditional GANs (CGANs) [135], which learns to generate data in response to a user-defined input label. For example, earlier works [121, 136] develop a conditional model where the user specifies the desired attributes, and the GAN generates time series features conditioned on the attributes. Our approach is conceptually similar to this idea, but rather than conditioning on the manual process of user-specified inputs, we obtain such "user-specified inputs" (attributes) using a standard multi-layer perceptron (MLP) network.

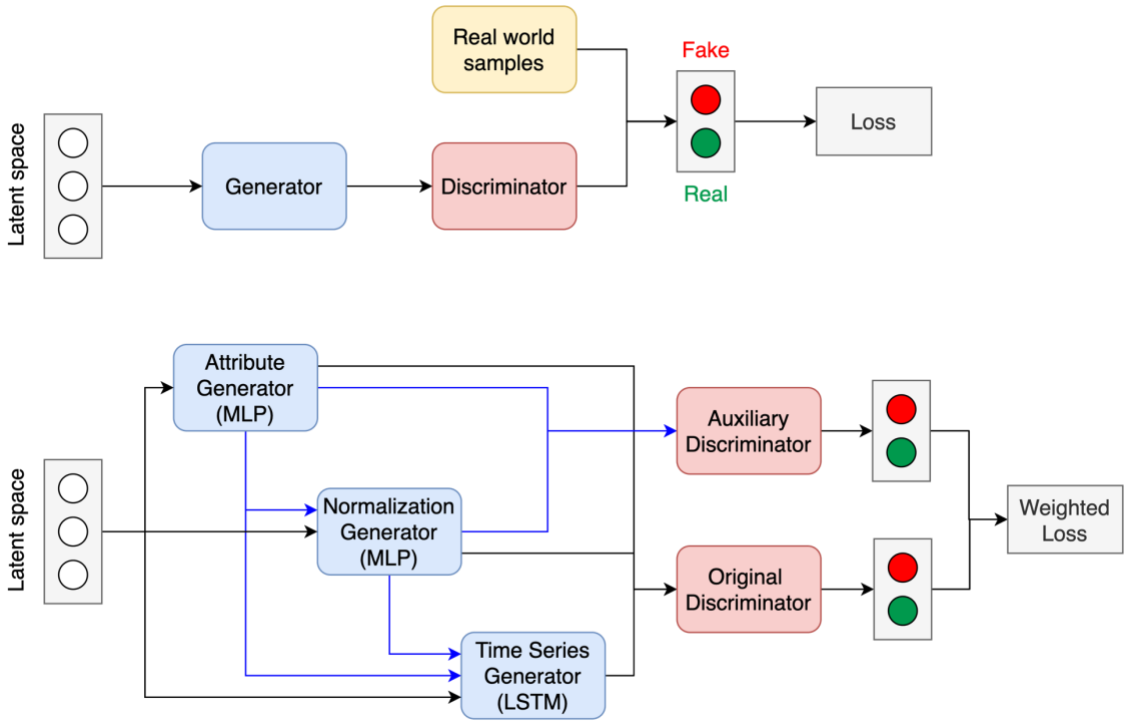

Figure 30. The architecture of the conventional GAN (top) and our proposed GAN (bottom).

Second, to better capture the temporal dependencies of the series, we employ a variant of RNNs, long short-term memory (LSTM), as the generator for time series features. The conventional GAN generator is a fully connected multi-layer perceptron (MLP), which is unsuitable for time series. A preferable option is to use RNNs, which are intended to model time series and have been extensively used to produce time series in the GAN literature [121, 122, 123]. At a high level, instead of producing the whole time series at once, RNNs

generate one record $\boldsymbol{F_d}$ (i.e., features at d nm information cutoff gate) at a time, and then run multiple passes (equals to the sequence length) to construct the entire time series step by step. RNNs can also learn correlations across the dimensions of a time series and produce multi-dimensional output. The critical distinction between RNNs and classic neural networks is that RNNs contain an internal state that implicitly encodes all past states of the signal. Thus, when generating $\boldsymbol{F_d}$, the RNN unit can incorporate the patterns from all the previous records $\boldsymbol{F_{10}, F_9, \ldots, F_{d-1}}$. At each timestamp, the generated attributes from the MLP network are added as an input to the LSTM network in order to retain the hidden relationships between the attributes and time series.

Third, we add another MLP generator to implement an auto-normalization heuristic to mitigate the problem of mode collapse, in which the GAN produces homogeneous samples while being trained on a diverse dataset. Mode collapse mitigation is a hot issue of study in the GAN community. Unlike image or medical data, which often displays similar value ranges across samples, our go-around sequential data exhibits much more range variability. Datasets with a more extensive value range tend to exacerbate mode collapse by having a more diversified variety of modes, making the data more difficult to be learned by GANs. In addition, due to the anomalous nature of the go-around sequence, some flight sequence samples may have extreme values that are beyond the typical rage. A standard normalization approach, which simply normalizes the data sequence by the global minimum and maximum values, may not be well-suited for the go-around use case. The mode collapse continues to occur since GAN learns basically the same thing - just scaling and shifting the feature values by a constant. This motivates us to normalize each time series individually [123], rather than normalizing over the entire dataset. The maximum and minimum values of each time series are considered as random variables (i.e., static features or attributes) to be learned by GAN. Thus, the GAN first learns to generate the maximum and minimum values defining the range for each time series individually, then rescales the sequence features generated by the LSTM network to fit inside this range. In this way, all the time series have their own range during generation, which alleviates the mode collapse issue.

Putting it all together, we generate all the features in the following three steps:

(1) Generate attributes using the MLP generator.

(2) With the generated attributes as inputs, generate the maximum and minimum values defining the range for each time series individually using another MLP generator.

(3) With the generated attributes and the maximum and minimum values as inputs, generate the time series features using the LSTM network.

Along with the primary discriminator, we introduce an auxiliary discriminator that discriminates only on attributes. A tunable weighting parameter is used to combine the Wasserstein losses [137] of two discriminators. The generator effectively learns from this

auxiliary discriminator to generate attributes with high fidelity. Further, with the help of the original discriminator, the generator can learn to generate time series features well.

8.5.3. Fidelity Analysis

Evaluating the fidelity of the generated samples is notoriously difficult. The most widely accepted metrics were developed for image pixels [138, 139, 140] and hence cannot be used in our datasets. In line with the recommendations of Lin et al. [123], we will assess the fidelity of the generated synthetic go-arounds in terms of sequence length, feature distributions, and serial correlation. These are also the criteria through which we select the model. Once the model is picked, we will apply the trained model to generate synthetic go-arounds and evaluate the go-around prediction model performance with varying fractions of synthetic samples added to the training set in the next section.

For each GAN model that is converged, we use the generator to "create" the same number (371) of synthetic go-around sequences. Below we present some comparisons between the 371 synthetic go-around sequences and the 371 real go-around sequences in order to evaluate the fidelity of the model. We benchmark this GAN-based data generation technique against the autoencoder sampling-based approach in Section 8.4.

A. Sequence Length

One aspect of evaluating the fidelity of data is to examine whether the algorithm generates time series of the appropriate length, particularly for our variable-length go-around sequences. In real-world scenarios, a go-around might occur near the airport (with a full sequence length of 9 timestamps in our analysis), or in the middle of the final approach. Our GAN model should be able to capture such a masking effect and generate synthetic go-around sequences of similar lengths. The side-by-side histogram in Figure 31 below compares the real go-around sequence length in blue, and the synthetic sequence length in orange. Compared to the distribution of sequence length for the real go-around sequences, the GAN generates most of the go-around sequences with nine timestamps, but also a considerable number of shorter sequences.

Note that the sampling-based generator can only generate time series of fixed length. Presumably, we could further truncate the generated sequences according to the empirical length distribution and compare them to the GAN-generated sequences. However, such a comparison is meaningless - the generated sequential data would perfectly reproduce the real length distribution, but it results from our human intervention, and not because of the algorithm learning to reproduce time series lengths. Hence, we only show the comparison between the GAN-generated sequences and the real sequences.

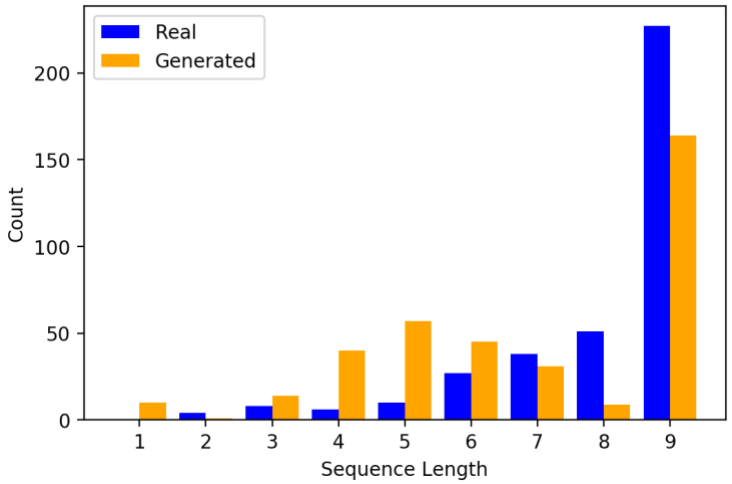

Figure 31. Histogram of the sequence length for real samples (blue) and synthetic samples (orange).

B. Feature Distribution

In this section, we will evaluate how well the generators learn to generate reasonable feature values. For every feature of each sequence, we calculate the representative values of the feature of the sequence as (max + min) / 2. The distribution of this value over all the synthetic sequences implicitly reflects how well the generator reproduces the range of time series values in the dataset. Instead of visually inspecting the histograms, we conduct the non-parametric test to determine whether the two data samples (real vs. synthetic) are drawn from the same distribution. The Kolmogorov-Smirnov test (K-S test) quantifies the distance between the empirical distribution functions of two samples with the null distribution stating that both samples are drawn from the same distribution. The K-S test is sensitive to differences in both location and shape of the empirical cumulative distribution functions of the two samples. We consider the two distributions to be the same if the K-S statistic is small or the p-value is above 0.05, where we fail to reject the null hypothesis at a 95% confidence level.

The figures below depict the distribution of the representative values for some features for the real go-around sequences and the synthetic go-around sequences, using the GAN generator (Figure 32) and the sampling-based generator (Figure 33), respectively. Both continuous features and categorical features are investigated. We observe that GAN much more closely mirrors the true feature distribution compared to the sampling-based approach, particularly in the tails of the true distribution. The figure title includes the p-values of the K-S test. Among the 46 features (including on-hot encoding dummy variables), 33 features generated by GAN have the same distributions as the real feature value distribution. In comparison, only 8 features generated by the sampling generator pass the K-S test. The GAN generator can learn to generate features well, and significantly improves the fidelity of the generated distributions.

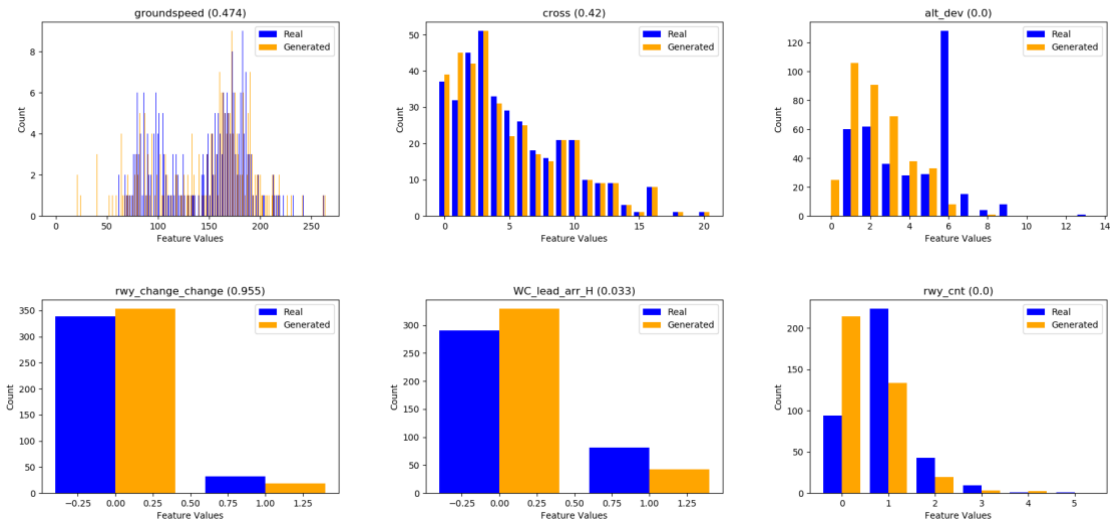

Figure 32. Clockwise, distribution of the feature distribution for real go-around sequences (blue) and synthetic go-around sequences (orange) generated by the GAN generator for continuous features: groundspeed, crosswind speed, altitude deviation, the number of objects on the runway; and categorical features: weight class of leading aircraft (heavy), runway configuration change.

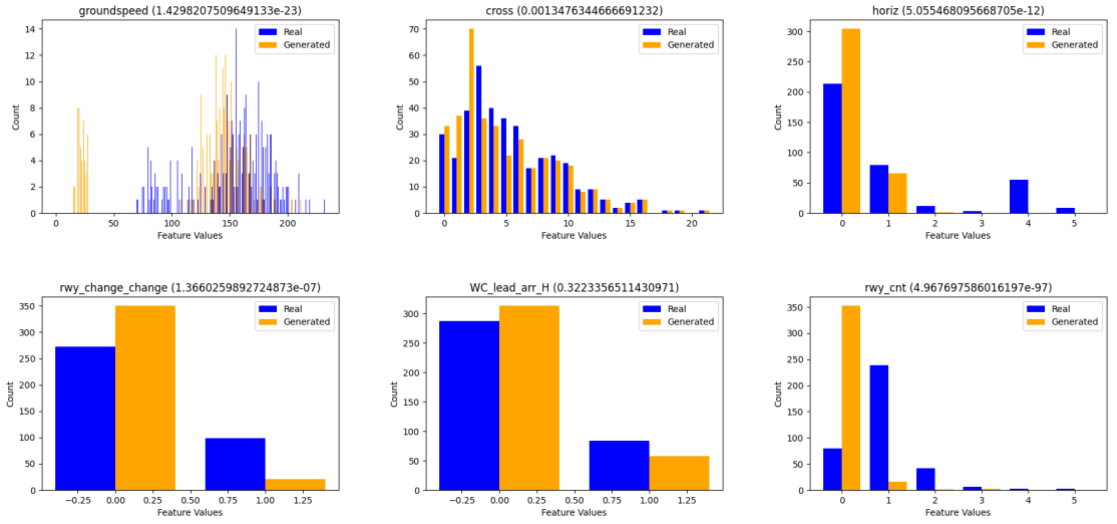

Figure 33. Clockwise, distribution of the feature distribution for real go-around sequences (blue) and synthetic go-around sequences (orange) generated by the sampling-based generator for continuous features: groundspeed, crosswind speed, altitude deviation, the number of objects on the runway; and categorical features: weight class of leading aircraft (heavy), runway configuration change.

C. Temporal Dependency

The specialty of sequential data is the time dependence between timestamps. In order to validate how the generators capture temporal correlations of the sequential data, we calculate the autocorrelation for each synthetic sequence for each feature. Autocorrelation, also known as serial correlation, is the correlation of a time series with a delayed copy of itself as a function of time lag. Informally, it is the similarity between observations as a function of the time lag between them. The figures below shows the autocorrelation for the real go-around sequences in blue and the learned autocorrelation for the synthetic go-around sequences in orange, for each selected feature. By comparing the serial correlation curves generated by GAN (Figure 34) to those generated by the sampling-based generator (Figure 35), the GAN model is capable of capturing the variant time dependence and not just simply generate the average scenarios. This is most likely due to the RNNs module we implemented in the GAN generator networks. All of the synthetic go-around sequences generated by the sampling-based generator exhibit the exact same serial correlation, which is shown in the Figure 35 as a single overlapping orange line. This may be because the autoencoder included in the sampling-based generator learns about time dependency in the latent space, but incorporates too little randomness, causing it to generate simplified correlations.

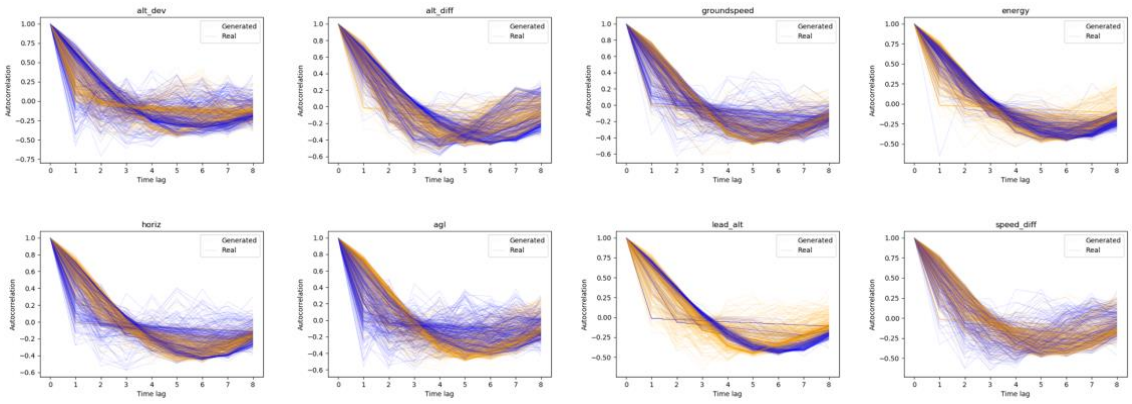

Figure 34. Clockwise, autocorrelation of real go-around sequence (blue) and synthetic go-arounds (orange) generated by GAN for features: altitude deviation, altitude difference, groundspeed, kinetic energy height, speed difference, altitude of leading aircraft, angle with the extended runway centerline, horizontal deviation.

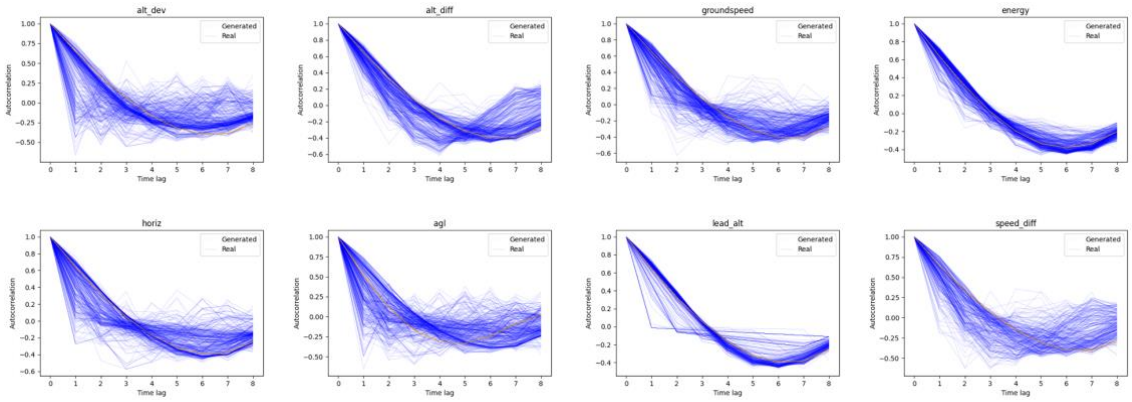

Figure 35. Clockwise, autocorrelation of real go-around sequence (blue) and synthetic go-arounds (orange) generated by sampling-based generator for features: altitude deviation, altitude difference, groundspeed, kinetic energy height, speed difference, altitude of leading aircraft, angle with the extended runway centerline, horizontal deviation.

8.5.4. Downstream Performance

The question to be answered in this section is that, with the high-fidelity synthetic go-around sequences generated to augment the minority class for model training, how the models trained on the synthetic data improve on the baseline models, and what will be the appropriate portions of synthetic data should be added for model training?

We utilize the generated data samples for the go-around prediction task, handling the class imbalance issue, and validating that models trained on synthetic data are generalizable to real data. As seen in Figure 36, we begin by reconstructing our dataset. Initially, we only have 0.3% of the go-arounds in the training set. We train the GAN networks on all the go-around flights, and then employ the model to generate synthetic go-around samples (in blue). Then, the original go-around and non-go-around samples (in green), together with the synthetic go-around samples (in blue) are formed to be a nearly class-balanced training set for predictors – depending on how many synthetic go-around sequences are generated. We apply the trained GAN model to generate 8854 go-around sequences combined with the 371 real go-arounds to increase the minority class in the training set to 10% of the total flight sequences, and generate 79,392 go-around sequences in addition to the 371 real go-arounds to increase the minority class in training set to 50% of the total flight sequences. Finally, we evaluate the model performance on a collection of metrics, including F2 score, recall score, precision score, and accuracy.

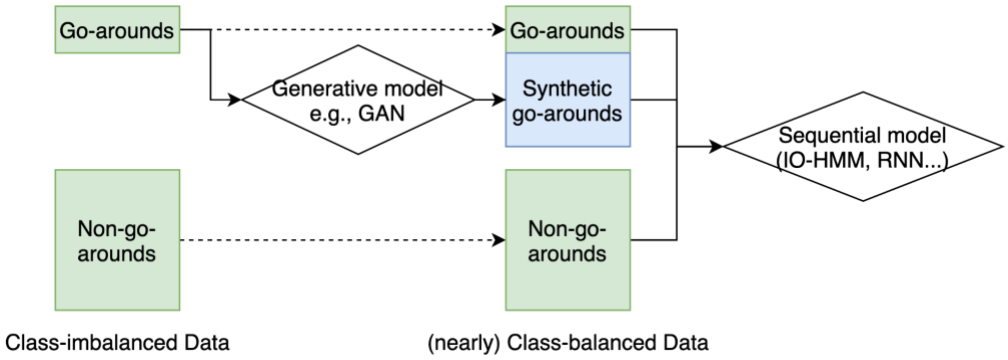

Figure 36. Reconstructing a more balanced dataset with synthetic go-around samples to augment the minority class.

In order to provide a pure comparison of the real and synthetic training set (without introducing effects produced from different predictors), we pick the LSTM model with multi-step-ahead prediction over the entire approach, suggested as the best predictor in Chapter 7, to train on different datasets. The following Table 16 summarizes the testing results of the LSTM predictor when trained on three different datasets (10%, 30%, and 50% of go-arounds) and tested on the same real data. As illustrated in Figure 37, F2 score is generally higher for the training set with 30% go-arounds. This is most likely because the class imbalance issue is eased by the additional amount of minority class samples added for training. The model trained on 50% synthetic go-arounds underperforms the models trained on 10% and 30% go-arounds, most likely due to the model learning an excessive number of "fake" samples. Future work is needed to determine the appropriate portions of synthetic data for training.

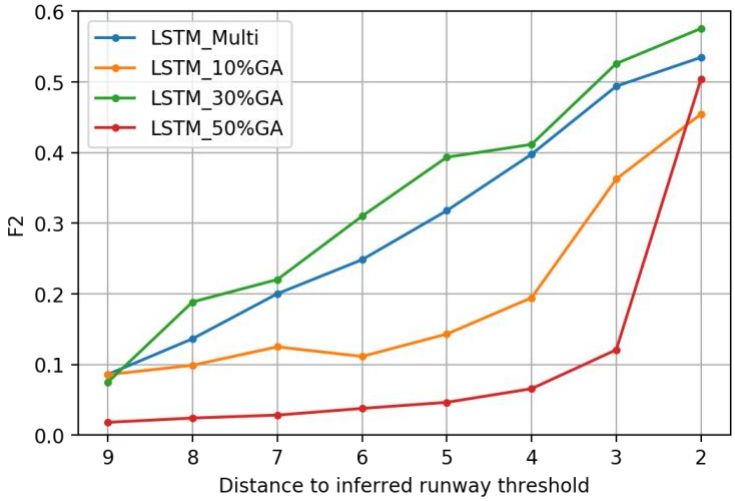

Figure 37. Model performance at different information cutoff gates with the original portion, 10%, 30% and 50% of go-arounds in the training set.

While we have demonstrated a promising path towards augmenting the minority class for imbalanced learning, further efforts on model development and generalization are required for researchers to confidently use such workflows. Dealing with imbalanced datasets is the crux of many real-world classification problems, especially in safety-related fields. The capability to generate complex multivariate time series data with variable length to augment the minority class not only benefits the go-around prediction in this study, but more generally, would be helpful for other problems that require the augmentation of rare events. We hope that such initial promise and open questions inspire more and further research by theoreticians and practitioners to help break the impasse in imbalanced learning.

Table 16. Model performance at different information cutoff gates with the original portion, 10%, 30% and 50% of go-arounds in the training set.

After the flight passes gate	Model	F2	Precision	Recall	AUC	Accuracy
9	LSTM_Multi	0.086	0.028	0.179	0.973	0.780
	10%GA	0.086	0.028	0.179	0.780	0.973
	30%GA	0.075	0.022	0.195	0.768	0.982
	50%GA	0.018	0.005	0.044	0.701	0.983
8	LSTM_Multi	0.137	0.058	0.205	0.985	0.824
	10%GA	0.099	0.087	0.103	0.720	0.993
	30%GA	0.189	0.092	0.256	0.823	0.988
	50%GA	0.024	0.005	0.238	0.723	0.905
7	LSTM_Multi	0.200	0.158	0.215	0.995	0.897
	10%GA	0.125	0.030	0.587	0.927	0.941
	30%GA	0.220	0.089	0.349	0.927	0.988
	50%GA	0.029	0.006	0.414	0.760	0.845
6	LSTM_Multi	0.248	0.096	0.410	0.983	0.882
	10%GA	0.112	0.031	0.308	0.796	0.962
	30%GA	0.310	0.144	0.436	0.844	0.988
	50%GA	0.038	0.008	0.432	0.797	0.872
5	LSTM_Multi	0.318	0.089	0.897	0.965	0.947
	10%GA	0.143	0.046	0.308	0.826	0.973
	30%GA	0.394	0.283	0.436	0.922	0.994
	50%GA	0.047	0.010	0.467	0.832	0.881

After the flight passes gate	Model	F2	Precision	Recall	AUC	Accuracy
4	LSTM_Multi	0.398	0.257	0.462	0.993	0.964
	10%GA	0.195	0.081	0.299	0.896	0.989
	30%GA	0.412	0.212	0.538	0.935	0.991
	50%GA	0.066	0.014	0.605	0.869	0.885
3	LSTM_Multi	0.494	0.295	0.590	0.993	0.957
	10%GA	0.363	0.246	0.410	0.953	0.993
	30%GA	0.526	0.420	0.538	0.911	0.995
	50%GA	0.121	0.028	0.714	0.914	0.924
2	LSTM_Multi	0.535	0.390	0.590	0.995	0.930
	10%GA	0.455	0.358	0.487	0.938	0.995
	30%GA	0.576	0.421	0.615	0.951	0.995
	50%GA	0.504	0.293	0.615	0.952	0.993

9. Real-Time Risk Predictive Framework

9.1. Overview

In this chapter, we introduce the Go-Around Prediction (GAP) software service and demonstrate its ultimate feasibility and practicality by testing it on real-time emulated traffic scenarios. The GAP service encapsulates the work developed in previous Chapters into the form of a software system, enabling stakeholders to readily assess the go-around probability in real-time during actual arrival operations in the NAS. Potential environments for the use of GAP include air traffic terminal automation systems, pilot displays, and NASA capabilities such as the In-Time Aviation Safety Management System (IASMS) and Digital Information Platform (DIP). Through the combination of a real-time data input stream and ML-based predictive models, the service allows for the continuous computation of the probability of a go-round. As each arrival flight approaches the airport, the prediction results can be updated and displayed to operators (i.e., air traffic controllers and pilots). This additional information will give operators enhanced situational awareness throughout the approach phase of flight, allowing them to mitigate growing risks earlier and, if needed, provide more time to safely execute go-arounds.

The GAP service provides ANSPs, airports, airlines, and other ISSA stakeholders with a practical risk detection tool that enhances safety. Specifically, we would like to develop a proof-of-concept tool that can be distributed to Air Traffic Control facilities and airport-focused personnel with the goal of identifying the probability of a go-around due to hazards in the approach domain in time for effective mitigation. Figure 38 depicts the overall concept and architecture of the GAP service. As seen in the figure, the GAP service is composed of two primary components: (1) on the left side of the figure is the machine learning model, having been trained on historical data, and (2) on the right side, the real-time data stream injects live air traffic positions, weather, and other data sources into the service, computes the probability of a go-around, and outputs that information from the service. In summary, the GAP service performs the following functions:

- It continuously monitors the arrival domain of the NAS and fuses disparate data to identify risk.
- It makes use of several novel machine learning techniques to predict the occurrence of a go-around or missed approach.
- It combines custom data collection capabilities with existing data dissemination frameworks to feed real-time data into state-of-the-art machine learning methods to provide real-time warnings.

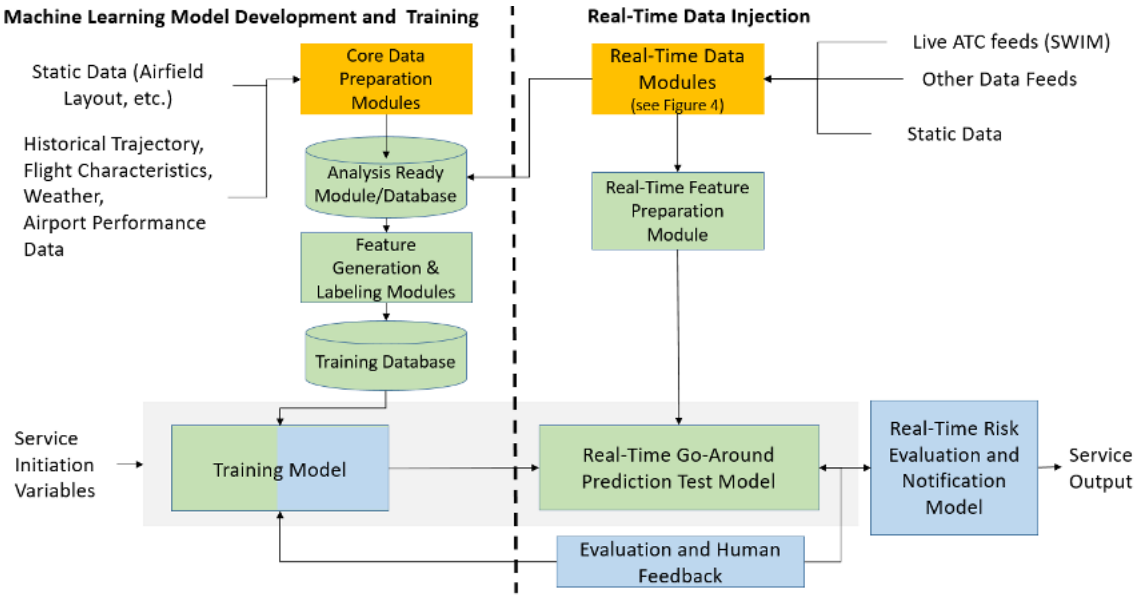

Figure 38. The proof-of-concept GAP service.

In the current stage, we will use real-time emulation data to demonstrate the ability to reliably predict the likelihood of go-arounds ahead of time. The prediction could help in risk mitigation at the pre-go-around stage and lessen inherent risks and uncertainties associated with the disruption of an airport when a go-around is executed. In order to achieve this capability, we identify the following requirements for the GAP service components, which will be discussed in further depth in the following subsections:

- There is a need to emulate the real-time data feed using historical data so we can effectively test the service on go-around situations. It is for demonstration purposes to assure that a go-around occurred during the model run.
- The models would need to be saved in order to be promptly called, accessed, and executed in response to real-time messages.
- The web interface tool should include a moving map display and indicators of go-around probability, as well as supplement information that might be helpful for decision making, such as predicted runway occupancy buffer, separation, wind speed.
- Several test scenarios of go-arounds need to be identified to demonstrate the proof-of-concept.

9.2. Real-Time Data Ingestion

The first key module of the GAP system is the real-time data injection mechanism. The GAP system requires key messages like trajectory messages from the FAA's SWIM feeds. Alternatively, this information can also come from an upstream source such as NASA's ATD-2/DIP Fuser, which fuses various inputs into a data stream ready for analysis. For our purposes, we introduced two additional processes providing further data curation as well as a real-time filtering layer that narrows the emphasis of the incoming data on the domain for GAP (within approximately 10 nm from the airport). The system can be operated in one of the two modes, as shown in Figure 39. The upper mode enables the process to be performed on actual live data coming from the data sources. The bottom emulated data stream is for demonstration purposes to assure that a go-around occurred during our model run. The modular architecture of the data injection components is a technically sound approach that facilitates easier integration with other systems.

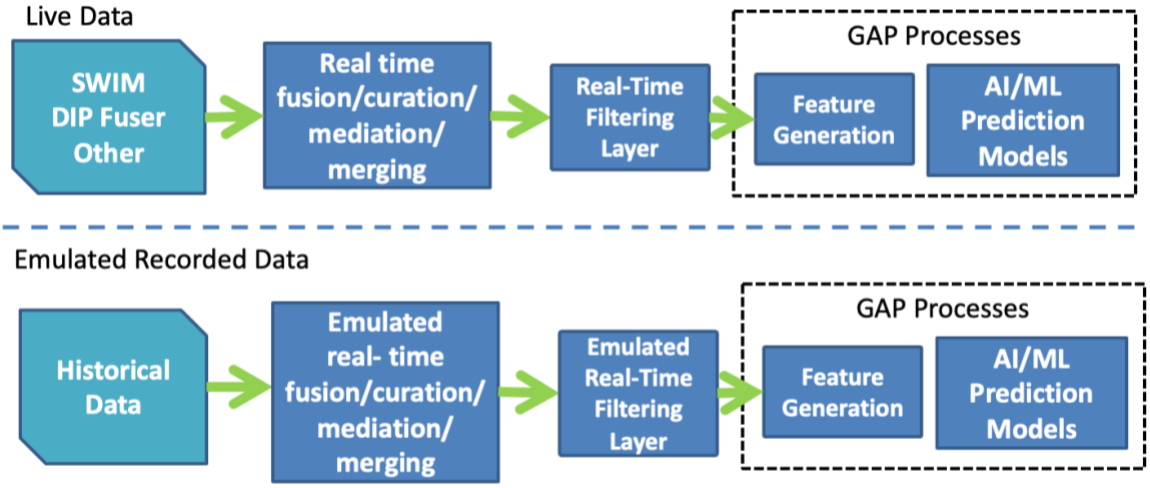

Figure 39. Real-time and emulated data streams and how we prepare them for input to the GAP processes.

Our real-time data pipeline is built on top of the Java Message Service (JMS, a Java API that enables applications to create, send, receive, and read messages) and Apache Software Service, which includes ***Apache Kafka*** (a distributed event streaming platform for handling real-time data feeds), ***Apache Flink*** (a stream processing framework with stream- and batch-processing capabilities), ***Apache ZooKeeper*** (a centralized service for maintaining configuration information, naming, providing distributed synchronization and group service), ***Apache NiFi*** (an integrated data logistics platform for automating the flow of data between software systems), and ***Apache ActiveMQ*** (a message broker written in Java to send messages between different applications).

To begin, the SWIM/DIP fuser or the historical data files are put in a designated repository in preparation for data ingestion into Kafka. Before launching the Kafka server,

we run the ZooKeeper instance that will keep track of the status of the Kafka cluster nodes, topics, partitions, etc. When Kafka is ready, we connect it to Flink clusters and create needed topics for storing, organizing, and messaging the input and output data.

Second, after the environment is set up, we initialize ActiveMQ and NiFi. We use the ActiveMQ source connector to read messages and write them to the created Kafka topics. NiFi is needed to publish and consume messages to and from ActiveMQ queues. Additionally, we configure two JMS processors – TapClient and TapProducer – to make this process more efficient.

Third, Flink will subscribe to the topics and consume data from the Kafka stream once everything is up and running. We let Flink periodically print out the number of messages published to the output Kafka topics. As long as messages are printed out, the emulation is working properly.

9.3. Offline Models

Due to the relative rarity of go-around operations, it is challenging to accumulate large enough samples set for reliable training of an online machine learning model. We thus opt for offline training, which requires saving all the models so they can be called and executed when new observations arrive. All the models have already been built, trained, and fine-tuned on a large store of historical datasets for production use, and they can be retrained periodically as needed.

There are two kinds of models we need to save for the GAP service – distance-variant models to predict ROB at each information cutoff gate, and the model to predict go-around occurrence. While the features remain the same, the coding frameworks used to derive them need to be adjusted to operate with the real-time data sources. In the current development stage, we retrain the ROB model with the XGBoost algorithm, and retrain the go-around prediction model using IO-HMM. In addition, we alter the sampling rate of the information cutoff gates with 0.5 nm spacing and rederive the features in order to train a more refined model capable of providing predictions at every 0.5 nm. The whole year of 2018 operations at the JFK airport are used for model training, albeit two months (April 27th – June 27th) are excluded from the analysis due to missing data in the APTC profile.

With the same feature space as we derived before, we retrain the XGBoost algorithm with the new dataset to model ROB at every 0.5 nautical miles. In total, 20 models are trained independently, each with its own training and testing set. These 20 models are then used to generate predicted ROB values ($\widehat{ROB}$) as one of the features for the go-around prediction model for the GAP service to provide the probability of go-around occurrence as the flight approaches the airport. In general, 27% – 70% of go-arounds are correctly classified by the model, and 4% – 20% of predicted go-arounds are correct. As the flight gets closer to the airport, the prediction becomes more reliable.

9.4. Test Scenarios

In order to identify suitable test scenarios for the real-time emulation demonstration, we undertook an exploratory analysis of historical go-around data at JFK. We defined two types of scenarios: *clusters*, where multiple go-around events occur in one hour, and *isolated events*, where only one go-around event occurs in at least an hour. Go-around clusters were detected by calculating the *cluster density* for each go-around event. Cluster density is the number of go-arounds that occur from 30 minutes before to 30 minutes after the event in question. According to these two types of scenarios, specific go-around events were chosen and investigated further.

We filter out some go-around events from testing consideration due to the current constraints imposed by our GAP service at this early stage of development. First, go-arounds at runways 13L and 13R were omitted due to the present limitations of the inferred runway algorithm (a critical component of the go-around prediction model). Second, the scenarios involving mixed-use runways are less desirable for test scenarios because the ROB prediction presently only considers arrivals. We further utilize the airport runway configuration data and historical departure data to determine if any of the runways in question were mixed-use, which means they were used concurrently for both departures and arrivals. Finally, by visually inspecting a replay of the trajectories, we validate that all selected go-around events were confirmed to be true positives.

A web-based user interface is developed for the GAP service to display the results of the go-around prediction. We can employ a centralized collection and processing infrastructure for both real-time operational data and predictive models through the web-based interface. In addition, the web interface is developed using open-source software tools, which enable potential customers to freely use the GAP capabilities during proof-of-concept demonstrations, as well as on a more regular basis once the capability is ready for commercial use. Figure 40 below shows a snapshot of the GAP viewer. In addition to the probability prediction, the display has the real-time representation of the actual operational traffic within the vicinity of the airport, including arrival, departure, and surface operations.

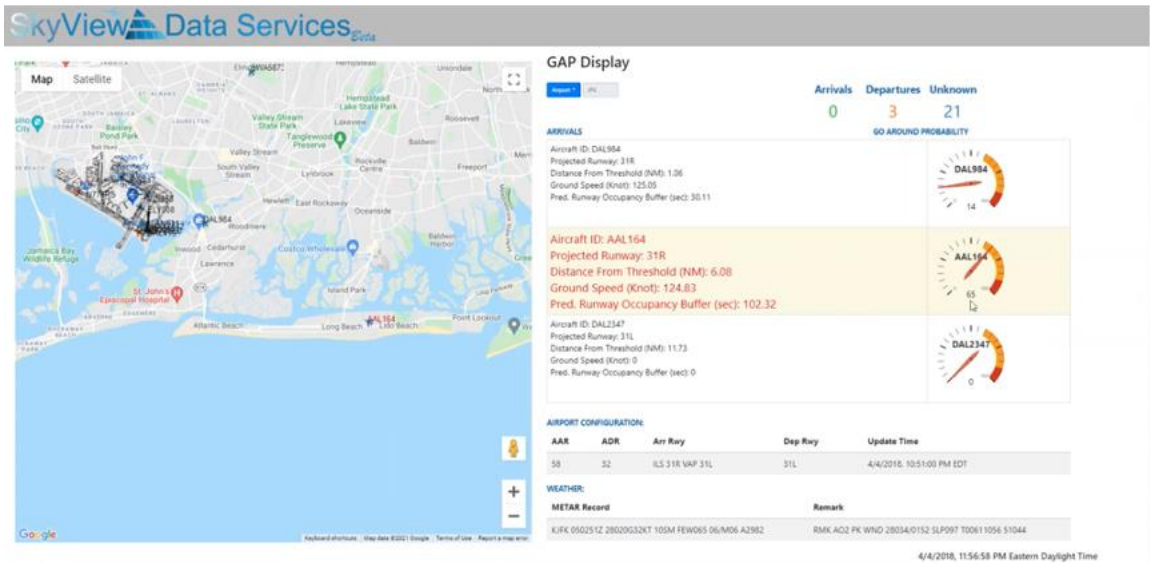

Figure 40. The GAP service provides a visualization display where subject aircraft positions and related metrics are continuously updated along with go-around predictions.

9.5. Real-Time Deployment

We integrate all the technical components and modules presented above to create a proof-of-concept GAP service demonstration that can be illustrated via the viewer and emulated using historical data. The overall architecture for the proof-of-concept GAP service is shown in Figure 41. The top set of processes (above the horizontal dashed lines) encompasses the training of the model on historical flights. The bottom set of processes illustrates the components required for the real-time prediction and outcome display.

As an example, we choose a test scenario in early April 2018 to demonstrate the end-to-end capability of the GAP service by emulating the historical data sets into real-time data streams. On this date, AAL164 is on approach to JFK runway 31R. The meteorological conditions include high wind gusts over 30 knots. A link to the video of the GAP software system in action for this scenario can be found at https://youtu.be/EVjPvEyp3g0. In order to construct this proof-of-concept demonstration, a number of processes are executed on the emulated data stream behind the scenes. We end this section by summarizing the whole workflow of the backend of the demonstration for the GAP capability, as demonstrated in Figure 42.

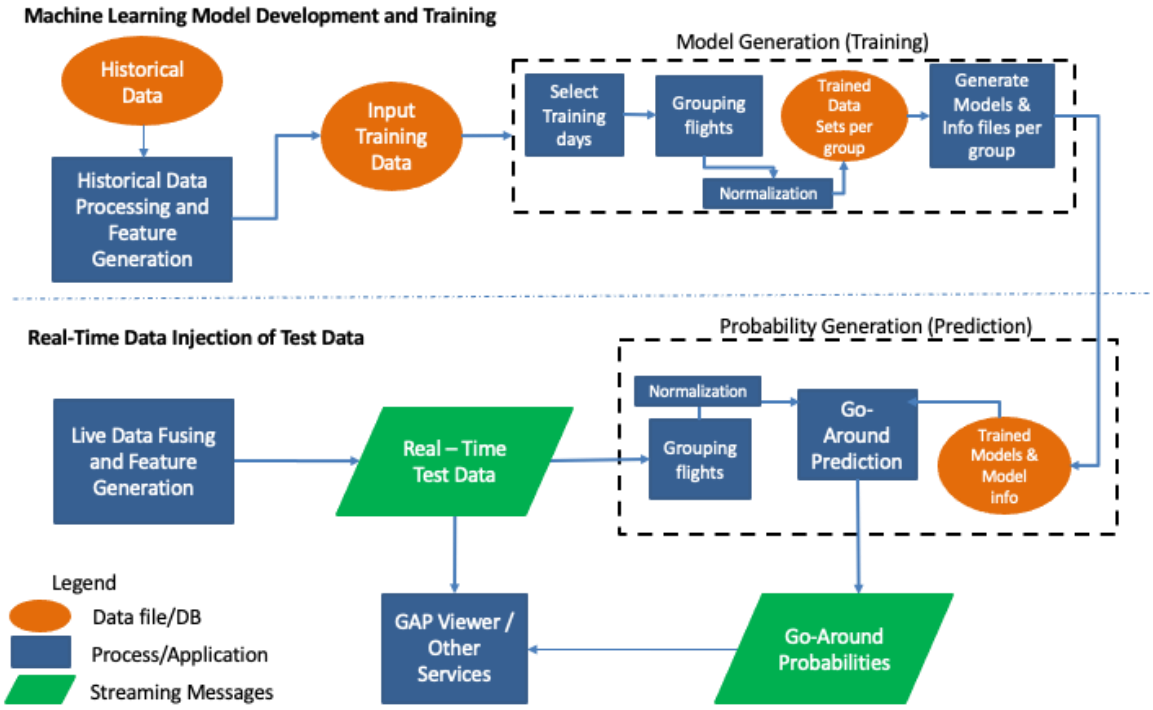

Figure 41. The overall architecture for the proof-of-concept GAP service.

After the models are well-trained on historical data, we save the models and leverage the real-time data pipeline which is built on top of JMS and Apache software service to obtain the real-time prediction of go-around occurrence. Specifically, when the JMS TapClient serializes real-time data streams (trajectory data, etc.) and publishes them to the Kafka server as topics, we are able to decode the messages using the Avro Schema and transform them to the desired data format. We set a few seconds (5 seconds in the current setting) of sleep time to allow for the accumulation of incoming data. The data acquired throughout the time window, in conjunction with the logging information kept in memory (if any), will be used to project the landing runway first and decide which information cutoff gate to be analyzed and predicted. Next, the Python scripts for deriving all of the required features will be executed. Once we obtain the predicted runway occupancy buffer (ROB) using the pre-trained model at the corresponding information cutoff gate, we include this additional feature into the design matrix, and use it for the go-around prediction. The predicted go-around probability, along with any other information that may be useful for operators (e.g., runway occupancy buffer, separation), will be displayed on the visualization platform. The whole workflow (except for the display service) has been tested on a Windows-x64 10 computer (256 GB RAM, Intel Xeon E5-2643v4 3.40 GHz) running Python 3.7.6, JAVA 1.8.0, and all supporting packages installed. The average running time for getting the go-around probability result for 5-second collected data is 0.839 seconds. This suggests that it is feasible to use the trained models to provide real-time guidance.

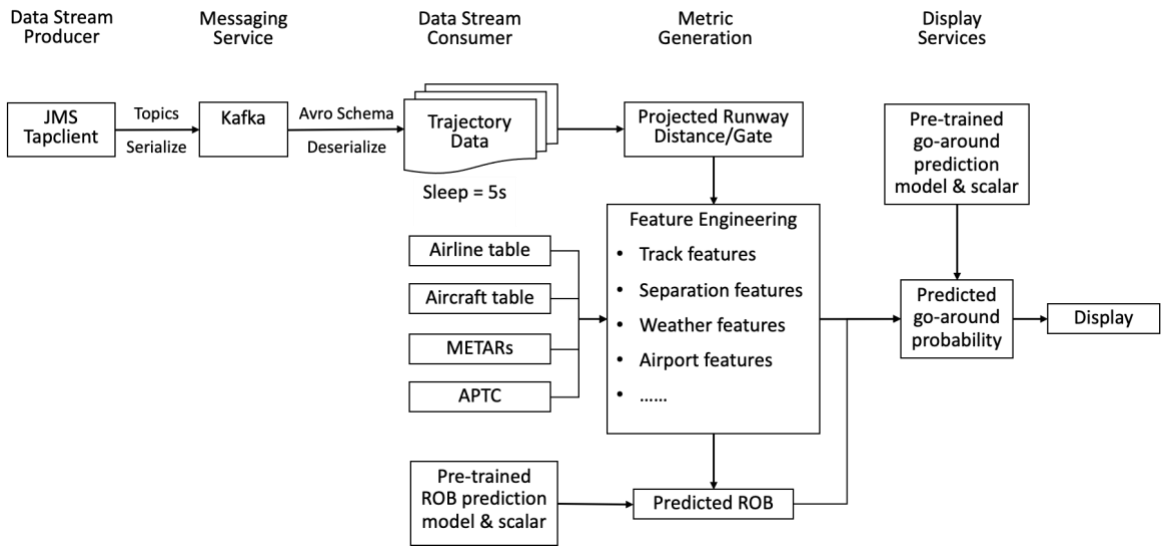

Figure 42. The workflow of the backend of the GAP service.

10. The Flight Plan

We close now with a "flight plan" outlining future research paths. Drawing on the present dissertation endeavor, we see great opportunities to leverage modern advances in AI for the study of go-arounds, or other application domains. As such, we discuss specific directions of promising research in the following areas.

(1) Complement the modules required to enhance the GAP service to handle more complex traffic situations and operations at various airports, thereby broadening the scope of scenarios covered by the developed framework.

(2) Overhaul the machine learning models at the core of the go-around prediction in order to improve their accuracy and reduce potential false positives. This enhances the value proposition of GAP to the airlines, FAA, and other stakeholders by enabling them to not only achieve increased situational awareness during the crucial approach phase of flight, but also to prevent the go-around from occurring in the first place resulting in reduced workload and efficiency gains.

(3) Integrate the GAP software service with several currently used operating platforms, including NASA's IASMS and DIP capabilities, to offer an easy access point for system users.

(4) Refine the use cases and conduct a deeper examination of the value created by the technology to narrow the problem space and enable us to focus on challenges faced by ISSA, such as real-time predictive analytics and how it translates to increased safety margins and economic benefit for operators.

(5) Extend the work beyond the identified use cases and re-prototype for other safety-related areas, such as autonomous driving and urban air mobility (UAM), leveraging its design principles and methods for studying rare events in a real-time arena.

10.1. Broaden The Scope

Thus far, the dissertation work has demonstrated a proof-of-concept containing an end-to-end process flow that combines a real-time data stream with machine learning models to achieve prediction results. We successfully built the GAP service for arrival operations at JFK airport. However, due to resource limitations, several operational scenarios were omitted. The operational envelope of the present proof-of-concept was limited in scope and operational complexity. In the future, generalization to other airports and the addition of mixed-use runway operations will broaden the breadth of operating coverage with the GAP service.

This work may also be extended to other atypical flight approach procedures, such as (1) Unstabilized Approach, in which an aircraft does not maintain an appropriate speed, descent rate, vertical/lateral flight path, or landing configuration; (2) Short Approach, in

which an aircraft shortens its downwind leg and turns earlier than normal for the base leg; (3) Dogleg Approach, in which an aircraft establishes its approach on the radar base leg for an ILS approach, to name a few.

Relevant to this opportunity area, the following questions need to be considered: (1) What additional data sources or processing steps are required to accommodate increased operational complexities such as mixed-use runway operations? (2) How can we make the system more adaptable so that we can shorten the amount of time and decrease the level of effort required to implement the system at a new airport? (3) What types of airports are candidates for expansion? (4) Will the increased complexity have a detrimental effect on the machine learning model and predictive performance, and if so, how can this problem be mitigated?

10.2. Improve The Model

We have developed predictive models that can regularly update the go-around probability as the flight continues along with its approach to the airport. Specifically, for each flight, using information from the realized own-flight and leading flight, airport surface traffic, and weather, we extract and derive associated feature vectors and use them to predict whether the flight will initiate a go-around before landing. This sequential prediction problem requires the extraction and fusing of multiple diverse data sources (flight trajectory, air traffic control, weather, surface conditions, etc.) in order to train a model and then use it in real time. In addition, we have been able to extract real-time information from the Kafka server and derive features fed into the pre-trained predictive model to estimate the probability of go-around occurrence in real time. Building on the current research effort, further improvements are needed:

(1) From the labeling perspective, unsupervised (clustering) methods can be leveraged to improve the model labeling scheme for better training. Instead of labeling the flight sequences with a binary response (go-around or not), flight activity sequences might be classified/clustered by unsupervised learning with multiple labels/classes. Transformer encoder model can be considered to learn the latent representation of the observed sequential data, and cluster the latent representation to construct an informative flight activity sequence for each aircraft. This may help ease the class imbalance issue and help with the understanding of different states of the flight sequences.

(2) From the feature engineering perspective, other than adding more features and deriving more robust signatures of the real-world scenarios, feature learning-based methods can be employed for dealing with the imbalanced pattern classification. The idea is to train an autoencoder to project the input feature space onto a learned feature space with better representation. Two or more autoencoders can be stacked together to provide more robust representations for different sets of features, and with different activation functions. Samples are then classified in the new feature space leaned in this manner instead of the original input space. Experimental results show that the autoencoder feature

learning method yields statistically significant improvement compared to resampling-based and feature projection methods for dealing with the imbalanced pattern classification problems.

(3) From the model architecture perspective, efforts could be put into designing other sequential models that outperform the current models with fewer false alarms without impacting the response time of the prediction, such as combining the generative Input-Output Hidden Markov Model with discriminative neural networks, the LSTM Fully Convolutional Network (LSTM-FCN) which works efficiently on multivariate time series classification tasks.

(4) From the model interpretability perspective, explore the possibility of root-causing the go-around occurrence. In addition to the evolving go-around probability conveyed to aviation stakeholders during the operations, one could take one step further to leverage Explainable AI (XAI) techniques to find the potential root causes (risk factors) for the high value of go-around prediction. While the prediction of go-around probability can enhance situational awareness, the root cause analysis can help uncover hidden connections and causalities behind the prediction, and thus support human decisions. The difficulty for this subtask would be that the scalability and real-time reaction need to be underpinned by automated RCA of the complex aviation system in order for them to be genuinely viable. There is a broad spectrum of techniques with the usual trade-off between tractability and expressiveness. Once the GAP service model is sufficiently mature, finding appropriate mechanisms to infer root causes of the go-around prediction in real time based on the resulting model is a promising direction to explore.

10.3. Integrate With Existing Platform

The objective is to operationalize the fundamental GAP predictive capabilities for the stakeholder application(s) with the greatest potential to affect system-wide safety or value proposition to the stakeholders. To accomplish this, GAP components should be improved to make them cloud-ready, allowing for integration with existing platforms like NASA's IASMS, DIP platform, and Skyview Data Services (SDS) platform. The integration efforts will eventually enable the GAP services to be supplied more efficiently to stakeholders, providing more robust solutions that realize benefits in safety and performance in the next-generation aerospace systems. This requires adapting the GAP services to operate with near-real-time streams coming into these platforms received through the FAA SWIM feeds and disseminating to stakeholders the GAP predictions based on this near real-time data.

The following research questions should be addressed: (1) What are the APIs or other interface mechanisms for the various platforms to be integrated with? (2) Do the platforms have access to the appropriate real-time data streams required for GAP to make accurate predictions? (3) When will these platforms be accessible for integration? (4) How will the GAP information be connected end-to-end with stakeholder systems? (5) How will they utilize it to enhance the safety and efficiency of the operation?

10.4. Closing The Loop

Closing the loop entails collaborating closely with domain experts in order to conduct a thorough evaluation of the software service. Human behavior and reasoning, as well as their interactions with the go-around decision-making assistance, are difficult to model. The difficulty is worsened further in settings where people have not yet experienced this tool. The objectives of this opportunity area are: (1) Identifying the most compelling use cases for potential users from a value proposition standpoint; (2) Comprehending how the aviation stakeholders will use the GAP service; (3) Deriving high-level requirements from stakeholder feedback in order to identify which specific technical components will need to be developed and/or enhanced. (4) Identifying other high-priority enhancements to be made to bring present methods up to date.

Before deploying the GAP software service for real-world applications, we will need to interview stakeholders, including airlines, to understand how the go-around prediction can benefit their operations in terms of both safety and efficiency. The effort is required to improve risk quantification—the estimation of the likely frequency of occurrence against the likely magnitude of impact if the risk led to an accident. Additionally, field tests in both emulations and real-world airport environments will aid in bridging the online-offline gap and establishing a fundamental understanding of which strategies perform effectively and why. The stakeholder feedback and emulation/testing data acquired during trials can be utilized to evaluate the potential for broader adoption of the GAP service for different kinds of flight anomalies and at other airports.

For this opportunity area, we seek answers to the following questions: (1) What is the value proposition of go-around prediction to each organization (e.g., airports, airlines, NASA, FAA, and other stakeholders)? (2) How much value is generated by avoiding a go-around occurrence? (3) What are the highest-priority areas for risk reduction related to go-arounds? (4) How should the information be displayed to optimize the advantages of safety and efficiency benefits? (5) How long in advance does the go-around prediction need to occur to be helpful to stakeholders? (6) How tolerant are stakeholders of false positives?

10.5. Open The Door

This opportunity area considers other potential applications and their markets beyond the identified use cases and creates an initial commercialization plan with government and industry stakeholders. The developed risk predictive intelligence can be prototyped and tested for use in other safety-related areas of transportation systems, such as autonomous vehicles (AV) and urban air mobility (UAM). The new application research can leverage the design principles and methodologies of the dissertation work so far for predicting rare event occurrences in a real-time arena, in order to improve the safety and efficiency of the system. However, when adopting this data-driven real-time risk predictive framework to other mobility settings, two significant paradigm shifts in data handling and prediction strategies need to be considered.

First, the primary obstacle is the lack of operational data to populate the risk predictive framework, since the unmanned systems are still in their early development stages. For autonomous driving, realized data from manned driving operations or AV testing offers an alternative and suitable approximations for model learning while overcoming the limitations of lacking empirical data from fully autonomous vehicles. Moreover, the insufficiency of accident/pre-crash data can be further compensated by synthetic off-normal scenarios generated by GANs, as we did in the go-around study – generating synthetic samples to augment the presence of anomalous events for better training and model generalization. In the aviation field, however, manned aircraft operations do not provide an adequate baseline for UAV operations due to the vast configuration differences and the varying operational context. While simulation data may be employed due to its scalability and convenience, the gap between simulation data and reality must be considered. Transfer Learning is a promising direction worth exploring since it has the potential to transfer the learned knowledge from simulated data to the real-world process.

Second, the current risk predictive analytics can be extended to an end-to-end interactive motion planner. The present study employs a non-interactive prediction paradigm, which typically performs acceptably in sparse traffic scenarios like commercial aircraft landing but can easily fail in dense traffic scenarios. Vehicles, pedestrians, and UAVs need to be modeled as active agents capable of not just capturing motion history and current interactions with other agents, but also reasoning about how other agents will react to their *future* behaviors. Specifically, while the sequential models (e.g., IOHMM, LSTM) are maintained for predicting the behaviors of individual agents using historical and current information, their output predictions of corresponded agents can be fed to another network layer with weights for joint training. All model parameters will be optimized simultaneously by considering the individual sequential models and their prediction interactions among multiple agents at the same time. Other interactive prediction and planning approaches such as game-theoretic planning and reinforcement learning should also be investigated.

References

[1] Doubiago, T. *Decarbonizing Road Transportation: Past the Tipping Point.* 2022.

[2] Waldek, S. *Here's How Many Planes Are in the Air at Any Given Moment.* 2022.

[3] Engineers, A.S.o.C. *America's Infrastructure Report Card.* https://infrastructurereportcard.org/#:~:text=Growing%20wear%20and%20tear%20on,over%20the%20past%20several%20years., 2021.

[4] Boeing Commercial, A. *Statistical summary of commercial jet airplane accidents*. 2020: Worldwide Operations.

[5] Office, A.C.R. *Bureau of Aircraft Accidents Archives.*

[6] Jarry, G., D. Delahaye, F. Nicol, and E. Féron. *Aircraft Atypical Approach Detection using Functional Principal Component Analysis*. in *SESAR Innovations Days 2018*. 2018. Salzburg, Austria.

[7] Oster, C.V., J.S. Strong, and C.K. Zorn. *Analyzing aviation safety: Problems, challenges, opportunities.* Research in Transportation Economics, 2013. **43**(1): p. 148-164.

[8] Li, L., R.J. Hansman, R. Palacios, and R. Welsch. *Anomaly detection via a Gaussian Mixture Model for flight operation and safety monitoring.* Transportation Research Part C: Emerging Technologies, 2016. **64**: p. 45-57.

[9] Das, S., B.L. Matthews, A.N. Srivastava, and N.C. Oza. *Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study*, in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, ACM: Washington, DC, USA. p. 47-56.

[10] Dutta, S. and J.S. Green. *Flight Mechanics Modeling and Post-Flight Analysis of ADEPT SR-1*, in *AIAA Aviation 2019 Forum*. 2019, AIAA AVIATION Forum Dallas, Texas.

[11] Hale, A. and D. Borys. *Working to rule or working safely? Part 2: The management of safety rules and procedures.* Safety Science, 2013. **55**: p. 222-231.

[12] Zolghadri, A. *Early warning and prediction of flight parameter abnormalities for improved system safety assessment.* Reliability Engineering & System Safety, 2002. **76**(1): p. 19-27.

[13] Li, G., Z. Zhou, C. Hu, L. Chang, Z. Zhou, and F. Zhao. *A new safety assessment model for complex system based on the conditional generalized minimum variance and the belief rule base*. Safety Science, 2017. **93**: p. 108-120.

[14] Leiden, K., S. Priess, P. Harrison, R. Stone, P. Strande, and M. Palmer. *Paired approach flight demonstration: Planning and development activities*. in *2018 Integrated Communications, Navigation, Surveillance Conference (ICNS)*. 2018.

[15] Wang, Z., L. Sherry, and J. Shortle. *Airspace risk management using surveillance track data: Stabilized approaches*. in *2015 Integrated Communication, Navigation and Surveillance Conference (ICNS)*. 2015.

[16] Wang, Z., L. Sherry, and J. Shortle. *Feasibility of using historical flight track data to nowcast unstable approaches*. in *2016 Integrated Communications Navigation and Surveillance (ICNS)*. 2016.

[17] Baomar, H. and P.J. Bentley. *Autonomous landing and go-around of airliners under severe weather conditions using Artificial Neural Networks*. in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. 2017.

[18] Donavalli, B. *Impact of weather factors on go-around occurrence*, in *Civil Engineering*. 2016, The University of Texas at Arlington.

[19] Dehais, F., J. Behrend, V. Peysakhovich, M. Causse, and C.D. Wickens. *Pilot Flying and Pilot Monitoring's Aircraft State Awareness During Go-Around Execution in Aviation: A*

Behavioral and Eye Tracking Study. The International Journal of Aerospace Psychology, 2017. **27**(1-2): p. 15-28.
[20] Chang, Y.-H., H.-H. Yang, and Y.-J. Hsiao. *Human risk factors associated with pilots in runway excursions.* Accident Analysis & Prevention, 2016. **94**: p. 227-237.
[21] Ross, G.E. *Human Factors Contributing to Unstabilized Approaches and Landings in Commercial Aviation Incidents: An Analysis of ASRS Reports*, in *Aeronautical Science*. 2018, Embry-Riddle Aeronautical University.
[22] Ellis, K., J. Koelling, M. Davies, and P. Krois. *In-time System-wide Safety Assurance (ISSA) Concept of Operations and Design Considerations for Urban Air Mobility (UAM)*. NASA/TM-2020-5003981, 2020.
[23] Mogford, R.H. and P. Munro. *NASA System Wide Safety (SWS) Project: Safety Metrics Research*. 2018.
[24] Ellis, K.K.E. *In-Time Terminal Area Risk Management.* 2022.
[25] FAA. *Air traffic by the numbers*, in *FAA Report*. 2019.
[26] Blajev, T. and C.W. Curtis. *Go-Around Decision-Making and Execution Project*. 2017, Flight Safety Foundation.
[27] Jou, R.-C., C.-W. Kuo, and M.-L. Tang. *A study of job stress and turnover tendency among air traffic controllers: The mediating effects of job satisfaction.* Transportation Research Part E: Logistics and Transportation Review, 2013. **57**: p. 95-104.
[28] Prats, X., V. Puig, J. Quevedo, and F. Nejjari. *Multi-objective optimisation for aircraft departure trajectories minimising noise annoyance.* Transportation Research Part C: Emerging Technologies, 2010. **18**(6): p. 975-989.
[29] Shortle, J. and L. Sherry. *A Model for Investigating the Interaction Between Go-Arounds and Runway Throughput*, in *2013 Aviation Technology, Integration, and Operations Conference*. 2013, American Institute of Aeronautics and Astronautics.
[30] Dahlstrom, N. and S. Nahlinder. *Mental workload in aircraft and simulator during basic civil aviation training.* The International journal of aviation psychology, 2009. **19**(4): p. 309-325.
[31] Dai, L., Y. Liu, and M. Hansen. *Modeling Go-around Occurrence*. in *Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)*. 2019.
[32] Dai, L., Y. Liu, and M. Hansen. *Modeling go-around occurrence using principal component logistic regression.* Transportation Research Part C: Emerging Technologies, 2021. **129**: p. 103262.
[33] Dai, L. and M. Hansen. *Real-Time Prediction of Runway Occupancy Buffers*. in *2020 International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT)*. 2020. IEEE.
[34] Dai, L., Y. Liu, and M. Hansen. *Predicting Go-around Occurrence with Input-Output Hidden Markov Model*. 2020, ICRAT.
[35] IATA. *Unstable Approaches: Risk Mitigation Policies, Procedures and Best Practices*. 2016, International Air Transport Association.
[36] FAA. *Stabilized Approach and Go-around*. 2018: FAA Safety Briefing.
[37] FAA. *Runway Safety - Runway Incursions*. 2015 [cited 2015 07-29-2019].
[38] Kennedy, Q., J.L. Taylor, G. Reade, and J.A. Yesavage. *Age and expertise effects in aviation decision making and flight control in a flight simulator.* Aviation, space, and environmental medicine, 2010. **81**(5): p. 489-497.

[39] Campbell, A., P. Zaal, J.A. Schroeder, and S. Shah. *Development of Possible Go-Around Criteria for Transport Aircraft*, in *2018 Aviation Technology, Integration, and Operations Conference*. 2018, American Institute of Aeronautics and Astronautics.
[40] Campbell, A., P. Zaal, S. Shah, and J.A. Schroeder. *Pilot Evaluation of Proposed Go-Around Criteria for Transport Aircraft*, in *AIAA Aviation 2019 Forum*. 2019, American Institute of Aeronautics and Astronautics.
[41] Zaal, P., A. Campbell, J.A. Schroeder, and S. Shah. *Validation of Proposed Go-Around Criteria Under Various Environmental Conditions*, in *AIAA Aviation 2019 Forum*. 2019, American Institute of Aeronautics and Astronautics.
[42] Shepherd, R., R. Cassell, R. Thapa, and D. Lee. *A reduced aircraft separation risk assessment model.* 1997.
[43] Sherry, L., Z. Wang, H. Kerkoub Kourdali, and J. Shortle. *Big data analysis of irregular operations: Aborted approaches and their underlying factors*. 2013. 1-10.
[44] Deshmukh, R., D. Sun, and I. Hwang. *Data-Driven Precursor Detection Algorithm for Terminal Airspace Operations*, in *Thirteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2019)*. 2019: Vienna, Austria.
[45] Bro, J. *FDM Machine Learning: An investigation into the utility of neural networks as a predictive analytic tool for go around decision making.* Journal of Applied Sciences and Arts, 2017.
[46] Figuet, B., R. Monstein, M. Waltert, and S. Barry. *Predicting airplane go-arounds using machine learning and open-source data*. in *Multidisciplinary Digital Publishing Institute Proceedings*. 2020.
[47] Soentpiet, R. *Advances in kernel methods: support vector learning*. 1999: MIT press.
[48] Rabiner, L. and B. Juang. *An introduction to hidden Markov models.* ieee assp magazine, 1986. **3**(1): p. 4-16.
[49] Bengio, Y. and P. Frasconi. *An Input Output HMM Architecture* 1995.
[50] Lee, D.-H., S. Zhang, A. Fischer, and Y. Bengio. *Difference target propagation*. in *Joint european conference on machine learning and knowledge discovery in databases*. 2015. Springer.
[51] Bengio, Y. and P. Frasconi. *Credit assignment through time: Alternatives to backpropagation*. in *Advances in Neural Information Processing Systems*. 1994.
[52] Lipton, Z.C., J. Berkowitz, and C. Elkan. *A critical review of recurrent neural networks for sequence learning.* arXiv preprint arXiv:1506.00019, 2015.
[53] Zhao, Z., W. Chen, X. Wu, P.C. Chen, and J. Liu. *LSTM network: a deep learning approach for short-term traffic forecast.* IET Intelligent Transport Systems, 2017. **11**(2): p. 68-75.
[54] Ma, X., Z. Tao, Y. Wang, H. Yu, and Y. Wang. *Long short-term memory neural network for traffic speed prediction using remote microwave sensor data.* Transportation Research Part C: Emerging Technologies, 2015. **54**: p. 187-197.
[55] Kong, D. and F. Wu. *HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction*. in *IJCAI*. 2018.
[56] Jeyakumar, J.V., E.S. Lee, Z. Xia, S.S. Sandha, N. Tausik, and M. Srivastava. *Deep convolutional bidirectional LSTM based transportation mode recognition*. in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. 2018.
[57] Zhang, P., W. Ouyang, P. Zhang, J. Xue, and N. Zheng. *Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[58] Ayhan, S. and H. Samet. *Aircraft trajectory prediction made easy with predictive analytics*. in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016.

[59] Liu, Y. and M. Hansen. *Predicting aircraft trajectories: a deep generative convolutional recurrent neural networks approach.* arXiv preprint arXiv:1812.11670, 2018.

[60] Gui, G., F. Liu, J. Sun, J. Yang, Z. Zhou, and D. Zhao. *Flight delay prediction based on aviation big data and machine learning.* IEEE Transactions on Vehicular Technology, 2019. **69**(1): p. 140-150.

[61] Fan, H., M. Jiang, L. Xu, H. Zhu, J. Cheng, and J. Jiang. *Comparison of long short term memory networks and the hydrological model in runoff simulation.* Water, 2020. **12**(1): p. 175.

[62] Srivastava, A., L. Valkov, C. Russell, M.U. Gutmann, and C. Sutton. *Veegan: Reducing mode collapse in gans using implicit variational learning.* Advances in neural information processing systems, 2017. **30**.

[63] Subramanian, S.V. and A.H. Rao. *Deep-learning based Time Series Forecasting of Go-around Incidents in the National Airspace System*, in *AIAA Modeling and Simulation Technologies Conference*. 2018, AIAA SciTech Forum: Kissimmee, Florida.

[64] Campbell, A.M., P.M.T. Zaal, J.A. Schroeder, and S.R. Shah. *Development of possible go-around criteria for transport aircraft*, in *Aviation Technology, Integration, and Operations Conference*. 2018, AIAA AVIATION Forum: Atlanta, Georgia.

[65] Lee, J.-G., J. Han, and K.-Y. Whang. *Trajectory clustering: a partition-and-group framework*, in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007, ACM: Beijing, China. p. 593-604.

[66] Kim, J., S.A. Palmisano, A. Ash, and R.S. Allison. *Pilot gaze and glideslope control.* ACM Trans. Appl. Percept., 2008. **7**(3): p. 1-18.

[67] Amelink, M.H.J., M. Mulder, M.M. van Paassen, and J. Flach. *Theoretical Foundations for a Total Energy-Based Perspective Flight-Path Display.* The International Journal of Aviation Psychology, 2005. **15**(3): p. 205-231.

[68] Gluck, J., A. Tyagi, A. Grushin, D. Miller, S. Voronin, J. Nanda, and N.C. Oza. *Too fast, too low, and too close: improved real time safety assurance of the national airspace using Long Short Term Memory*. in *AIAA Scitech 2019 Forum*. 2019.

[69] FAA. *Airport Capacity and Delay Analyses*, in *FAA Technical Center Report*. 1991, FAA: FAA Technical Center Report.

[70] FAA. *Aeronautical information manual*. 2011, US Department of Transportation Washington, DC.

[71] Dai, L. and M. Hansen. *Real-time Prediction of Runway Occupancy Buffer*. in *International Conference on Artificial Intelligence and Data Analytics for Air Transportation*. 2020. Singapore: IEEE Xplore Digital Library and Scopus.

[72] FAA. *Runway Safety Area Improvements in the United States*. 2007, International Civil Aviation Organization

[73] Tosic, V. and R. Horonjeff. *Effect of multiple path approach procedures on runway landing capacity.* Transportation research, 1976. **10**(5): p. 319-329.

[74] Simpson, R.W. *Potential impacts of advanced technologies on the ATC capacity on high-density terminal areas [microform] / Robert W. Simpson, Amedeo R. Odoni, and Francisco Salas-Roche*. NASA contractor report ; 4024., ed. A.R. Odoni, et al. 1986, [Washington, D.C.] : [Springfield, Va: National Aeronautics and Space Administration, Scientific and Technical Information Branch ; For sale by the National Technical Information Service].

[75] Ruiz, S., M.A. Piera, and I. Del Pozo. *A Medium Term Conflict Detection and Resolution system for Terminal Maneuvering Area based on Spatial Data Structures and 4D Trajectories.* Transportation Research Part C: Emerging Technologies, 2013. **26**: p. 396-417.

[76] Liu, Y., M. Hansen, G. Gupta, W. Malik, and Y. Jung. *Predictability impacts of airport surface automation.* Transportation Research Part C: Emerging Technologies, 2014. **44**: p. 128-145.

[77] Meijers, N.P. and R.J. Hansman. *A data-driven approach to understanding runway occupancy time*. in *AIAA Aviation 2019 Forum*. 2019.

[78] Herrema, H.F., B.D. Treve, R. Curran, and H.G. Visser. *A novel machine learning model to predict abnormal Runway Occupancy Times and observe related precursors*. 2017.

[79] Herrema, F., V. Treve, B. Desart, R. Curran, and D. Visser. *A novel machine learning model to predict abnormal Runway Occupancy Times and observe related precursors*. in *12th USA/Europe Air Traffic Management Research and Development Seminar*. 2017.

[80] Legge, J. *Designing a real-time ramp arrival prediction tool*. in *The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576)*. 2004.

[81] Levy, B.S. and S. Bedada. *A real-time ETA-to-threshold prediction tool*. in *2006 ieee/aiaa 25TH Digital Avionics Systems Conference*. 2006. IEEE.

[82] Nikoleris, T. and M. Hansen. *Effect of Trajectory Prediction and Stochastic Runway Occupancy Times on Aircraft Delays.* Transportation Science, 2015. **50**(1): p. 110-119.

[83] Simpson, R.W., A.R. Odoni, and F. Salas-Roche. *Potential impacts of advanced technologies on the ATC capacity of high-density terminal areas*. 1986.

[84] Kolos-Lakatos, T. *The influence of runway occupancy time and wake vortex separation requirements on runway throughput*, in *Massachusetts Institute of Technology. Department of Aeronautics and Astronautics*. 2013, Massachusetts Institute of Technology: Massachusetts Institute of Technology.

[85] Zhou, K., Q. Hou, R. Wang, and B. Guo. *Real-time KD-tree construction on graphics hardware.* ACM Trans. Graph., 2008. **27**(5): p. 1-11.

[86] Friso, H.F., C. Richard, H.G. Visser, T. Vincent, and D. Bruno. *Predicting abnormal runway occupancy times and observing related precursors.* Journal of Aerospace Information Systems, 2018. **15**(1): p. 10-21.

[87] Breiman, L. *Random forests.* Machine learning, 2001. **45**(1): p. 5-32.

[88] Strobl, C., A.-L. Boulesteix, A. Zeileis, and T. Hothorn. *Bias in random forest variable importance measures: Illustrations, sources and a solution.* BMC Bioinformatics, 2007. **8**(1): p. 25.

[89] Altmann, A., L. Toloşi, O. Sander, and T. Lengauer. *Permutation importance: a corrected feature importance measure.* Bioinformatics, 2010. **26**(10): p. 1340-1347.

[90] Holland, P.W. and D.B. Rubin. *CAUSAL INFERENCE IN RETROSPECTIVE STUDIES.* ETS Research Report Series, 1987. **1987**(1): p. 203-231.

[91] Chavent, M., V. Kuentz-Simonet, A. Labenne, and J. Saracco. *Multivariate Analysis of Mixed Data: The R Package PCAmixdata.* 2017.

[92] Aguilera, A.M., M. Escabias, and M.J. Valderrama. *Using principal components for estimating logistic regression with high-dimensional multicollinear data.* Computational Statistics & Data Analysis, 2006. **50**(8): p. 1905-1924.

[93] Kaiser, H.F. *Coefficient Alpha for a Principal Component and the Kaiser-Guttman Rule.* Psychological Reports, 1991. **68**(3): p. 855-858.

[94] Carter Hill, R., T.B. Fomby, and S.R. Johnson. *Component selection norms for principal components regression.* Communications in Statistics - Theory and Methods, 1977. **6**(4): p. 309-334.
[95] Longman, R.S., A.A. Cota, R.R. Holden, and G.C. Fekken. *A Regression Equation for the Parallel Analysis Criterion in Principal Components Analysis: Mean and 95th Percentile Eigenvalues.* Multivariate Behavioral Research, 1989. **24**(1): p. 59-69.
[96] The Port Authority of New York & New Jersey. *Runways at John F. Kennedy International Airport*, T.P.A.o.N.Y.N. Jersey, Editor. 2020.
[97] Mason, R.L. and R.F. Gunst. *Selecting principal components in regression.* Statistics & probability letters, 1985. **3**(6): p. 299-301.
[98] Jolliffe, I.T. *Principal components in regression analysis*, in *Principal component analysis*. 1986, Springer. p. 129-155.
[99] Dai, L., Y. Liu, and M. Hansen. *In Search of the Upper Limit to Air Traffic Control Communication*, in *International Conference for Research in Air Transportation*. 2018: Barcelona, Spain.
[100] Martinez, D., S. Belkoura, S. Cristobal, F. Herrema, and P. Wachter. *A Boosted Tree Framework for Runway Occupancy and Exit Prediction*, in *Eighth SESAR Innovation Days*. 2018.
[101] Rodríguez-Sanz, Á., J.M. Cordero, B.R. Fernández, F. Gómez, Comendador, and R.A. Valdés. *Assessment of the Airport Operational Dynamics Using a Multistate System Approach*, in *USA/Europe Air Traffic Management Research and Development Seminar*. 2019: Vienna, Austria.
[102] Dai, L., M. Hansen, M.O. Ball, and D.J. Lovell. *Having a Bad Day? Predicting High Delay Days in the National Airspace System.*
[103] Hochreiter, S. and J. Schmidhuber. *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
[104] Kingma, D.P. and J. Ba. *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.
[105] Prechelt, L. *Early stopping-but when?*, in *Neural Networks: Tricks of the trade*. 1998, Springer. p. 55-69.
[106] Lin, T.-Y., P. Goyal, R. Girshick, K. He, and P. Dollár. *Focal loss for dense object detection*. in *Proceedings of the IEEE international conference on computer vision*. 2017.
[107] Thisted, R.A. *Ridge regression, minimax estimation, and empirical Bayes methods*. 1976: Department of Statistics, Stanford University.
[108] Yu, S.-Z. and H. Kobayashi. *An efficient forward-backward algorithm for an explicit-duration hidden Markov model.* IEEE signal processing letters, 2003. **10**(1): p. 11-14.
[109] Townsend, J.T. *Theoretical analysis of an alphabetic confusion matrix.* Perception & Psychophysics, 1971. **9**(1): p. 40-50.
[110] Cavalcante, I.M., E.M. Frazzon, F.A. Forcellini, and D. Ivanov. *A supervised machine learning approach to data-driven simulation of resilient supplier selection in digital manufacturing.* International Journal of Information Management, 2019. **49**: p. 86-97.
[111] Goodall, P., R. Sharpe, and A. West. *A data-driven simulation to support remanufacturing operations.* Computers in Industry, 2019. **105**: p. 48-60.
[112] Holden, D., B.C. Duong, S. Datta, and D. Nowrouzezahrai. *Subspace neural physics: Fast data-driven interactive simulation*. in *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2019.

[113] Chawla, N.V., K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. *SMOTE: synthetic minority over-sampling technique.* Journal of artificial intelligence research, 2002. **16**: p. 321-357.
[114] He, H., Y. Bai, E.A. Garcia, and S. Li. *ADASYN: Adaptive synthetic sampling approach for imbalanced learning*. in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. 2008. IEEE.
[115] Islam, Z., M. Abdel-Aty, Q. Cai, and J. Yuan. *Crash data augmentation using variational autoencoder.* Accident Analysis & Prevention, 2021. **151**: p. 105950.
[116] Roy, S.K., J.M. Haut, M.E. Paoletti, S.R. Dubey, and A. Plaza. *Generative Adversarial Minority Oversampling for Spectral–Spatial Hyperspectral Image Classification.* IEEE Transactions on Geoscience and Remote Sensing, 2021. **60**: p. 1-15.
[117] Mariani, G., F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi. *Bagan: Data augmentation with balancing gan.* arXiv preprint arXiv:1803.09655, 2018.
[118] Andresini, G., A. Appice, L. De Rose, and D. Malerba. *GAN augmentation to deal with imbalance in imaging-based intrusion detection.* Future Generation Computer Systems, 2021. **123**: p. 108-127.
[119] Huang, G. and A.H. Jafari. *Enhanced balancing GAN: Minority-class image generation.* Neural Computing and Applications, 2021: p. 1-10.
[120] Dumagpi, J.K. and Y.-J. Jeong. *Evaluating gan-based image augmentation for threat detection in large-scale xray security images.* Applied Sciences, 2020. **11**(1): p. 36.
[121] Esteban, C., S.L. Hyland, and G. Rätsch. *Real-valued (medical) time series generation with recurrent conditional gans.* arXiv preprint arXiv:1706.02633, 2017.
[122] Yoon, J., D. Jarrett, and M. Van der Schaar. *Time-series generative adversarial networks.* Advances in Neural Information Processing Systems, 2019. **32**.
[123] Lin, Z., A. Jain, C. Wang, G. Fanti, and V. Sekar. *Using GANs for sharing networked time series data: Challenges, initial promise, and open questions*. in *Proceedings of the ACM Internet Measurement Conference*. 2020.
[124] Yu, L., W. Zhang, J. Wang, and Y. Yu. *Seqgan: Sequence generative adversarial nets with policy gradient*. in *Proceedings of the AAAI conference on artificial intelligence*. 2017.
[125] Fedus, W., I. Goodfellow, and A.M. Dai. *Maskgan: better text generation via filling in the_*. arXiv preprint arXiv:1801.07736, 2018.
[126] Brock, A., J. Donahue, and K. Simonyan. *Large scale GAN training for high fidelity natural image synthesis.* arXiv preprint arXiv:1809.11096, 2018.
[127] Frid-Adar, M., I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. *GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification.* Neurocomputing, 2018. **321**: p. 321-331.
[128] Zhang, Y., Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin. *Adversarial feature matching for text generation*. in *International Conference on Machine Learning*. 2017. PMLR.
[129] Wang, K. and X. Wan. *SentiGAN: Generating Sentimental Texts via Mixture Adversarial Networks*. in *IJCAI*. 2018.
[130] Mogren, O. *C-RNN-GAN: Continuous recurrent neural networks with adversarial training.* arXiv preprint arXiv:1611.09904, 2016.
[131] Li, S., S. Jang, and Y. Sung. *Automatic melody composition using enhanced GAN.* Mathematics, 2019. **7**(10): p. 883.
[132] Doersch, C. *Tutorial on variational autoencoders.* arXiv preprint arXiv:1606.05908, 2016.

[133] Liu, X.-Y., J. Wu, and Z.-H. Zhou. *Exploratory undersampling for class-imbalance learning.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2008. **39**(2): p. 539-550.
[134] Alshraideh, H. and G. Runger. *Process monitoring using hidden Markov models.* Quality and Reliability Engineering International, 2014. **30**(8): p. 1379-1387.
[135] Mirza, M. and S. Osindero. *Conditional generative adversarial nets.* arXiv preprint arXiv:1411.1784, 2014.
[136] Fu, R., J. Chen, S. Zeng, Y. Zhuang, and A. Sudjianto. *Time series simulation by conditional generative adversarial net.* arXiv preprint arXiv:1904.11419, 2019.
[137] Frogner, C., C. Zhang, H. Mobahi, M. Araya, and T.A. Poggio. *Learning with a Wasserstein loss.* Advances in neural information processing systems, 2015. **28**.
[138] Larsen, A.B.L., S.K. Sønderby, H. Larochelle, and O. Winther. *Autoencoding beyond pixels using a learned similarity metric*. in *International conference on machine learning*. 2016. PMLR.
[139] Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. *Gans trained by a two time-scale update rule converge to a local nash equilibrium.* Advances in neural information processing systems, 2017. **30**.
[140] Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. *Improved techniques for training gans.* Advances in neural information processing systems, 2016. **29**.

Appendix A: Data Sources

A.1. Asynchronous Data

We leverage large stores of historical data, including NASA's Sherlock ATM Data Warehouse, to provide training sets for model development. We limit the scope of the study to JFK airport due to a number of factors including the frequency of go-arounds, the mix of international and domestic traffic, and the complexity of arrival operations in the New York area. Our datasets range from July 1st to December 24th in 2018 at the John F. Kennedy (JFK) airport, except for four days with defective data. After data cleaning and matching, there are on average 525 arrival flights per day in the analyzed airport within the analysis period.

The first dataset is retrieved from the Integrated Flight Format (IFF) and Reduced Data (RD) summary of the NASA Sherlock Data Warehouse, which are gathered from 76 FAA facilities and formatted by ATAC corporation. Fields of interest include flight summary (e.g., time, aircraft type, origin, destination, operation type), trajectory information (timestamp, latitude, longitude, altitude, groundspeed, course, rate of climb, etc.), and landing information (e.g., runway threshold crossing time). Arrival trajectories have been filtered to 400 nautical miles centered on the analyzed airport for each flight. The RD summary includes the information of takeoff / landing runway and runway threshold crossing time. The RD summary and the IFF data have been further processed and merged on a daily basis for each flight arriving at JFK. This dataset is used to derive flight-specific characteristics (Section 4.3.1), approach stability features (Section 4.3.2), and in-trail separation features (Section 4.3.3)

The second dataset, airport surface detection equipment Model X (ASDE-X) data, allows us to determine the position of aircraft and ground support equipment in the airport surface area. Each record of raw surface track data contains the timestamp, latitude, longitude, altitude, and groundspeed. This dataset is useful in identifying key metrics for airport surface operations, which we will discuss in detail in Chapter 5.

The third dataset, which comes from the FAA aviation system performance metrics (ASPM) database, provides airport level configuration and weather information every quarter-hour. The dataset includes count of Official Airline Guide (OAG) scheduled arrivals/departures, airport arrival/departure rate (AAR/ADR, which measure airport arrival and departure throughput capacity), meteorological conditions flag (instrument or visual, reflecting whether conditions allow for pilots to operate without instruments), ceiling (in feet), visibility (in statute miles), wind angle from magnetic north (degree), wind speed (in knots), and airport supplied runway configuration. We use this dataset to derive the airport and weather features by matching flight-level operation with the 15-minute interval weather information according to the time at which the aircraft reaches a certain distance from the landing runway threshold.

A.2. Real-Time Data

With the advent of the FAA's System Wide Information Management (SWIM) system for dissemination of high-definition trajectory and air traffic control automation data, we are able to provide the path forward to provide real-time predictive analytics. Below are the data sources available in real-time and that can replace the asynchronous datasets during the development of real-time application software systems for go-around predictions.

The Aviation Weather Center (AWC) METeorological Aerodrome Reports (METARs) is used to replace the ASPM dataset to provide airport level weather information updates every hour. We validate the historical records between METARs and ASPM. It is found that these two datasets are not consistent during periods when daylight saving time is observed due to the conversion error of the ASPM dataset. We use this dataset to derive weather related features such as wind, visibility, and ceiling by matching each flight with the most-updated information (i.e., record that is available in the latest hour).

The Airport Configuration (APTC) profile provides the real-time data stream of runway configuration, airport arrival/departure rate (AAR/ADR), and meteorological conditions at the airport. The data comes from the FAA SWIM's Flight Information Service (AFIS), which provides timely and specific information on individual flights from the operators.

The ASDE-X dataset is substituted with the EV database. The EV database provides runway threshold crossing time and the runway exit time. We thus can calculate the metric we need (i.e., runway occupancy buffer that we will introduce in Chapter 5.) in a more convenient way, without inferring from the trajectory and airport surface configuration.