UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Jorge Luiz Franco

Graph machine learning for flight delay prediction due to holding manouver

São Carlos

2024

Jorge Luiz Franco

Graph machine learning for flight delay prediction due to holding manouver

Monograph presented to the Undergraduate Program in Computer Science at the Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, as part of the requirements for obtaining a bachelor's degree.

Advisor: Prof. Filipe Alves Neto Verri, Ph.D.

São Carlos
2024

S856m

Franco, Jorge Luiz

Graph machine learning for flight delay prediction due to holding manouver / Jorge Luiz Franco ; orientador Filipe Alves Neto Verri. – São Carlos, 2024.

36 p. : il. (algumas color.) ; 30 cm.

1. Machine Learning. 2. Flight Delay. 3. Graphs I. Filipe Alves Neto Verri. II. Universidade de São Paulo. III. Instituto de Ciências Matemáticas e de Computação. IV. Graph ML for Flight Delay Prediction due to Holding Manouver

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my family—Raul, Barbara, Sandra, and Ester—for their unwavering support throughout this journey, especially my father, João Luiz Franco, who has always encouraged my studies. It is an honor to be graduating from the same Institute and in the same course where my hero graduated in 1987.

I extend my deepest gratitude to my advisor, Filipe Alves Neto Verri, who has not only guided me this semester but has been a constant source of wisdom and inspiration throughout much of my academic journey. Thank you for your patience, guidance, encouragement and the opportunity you gave me, professor.

My heartfelt thanks also go to Manoel Vilela Machado, whose collaboration and friendship were instrumental in completing this work. Our numerous discussions provided invaluable learning experiences and opportunities for growth.

I am equally grateful to the entire ICMC staff, especially the graduation service team—for their patience with my many requests—and to the library, as well as the cleaning and general services staff, whose kindness and support made ICMC feel like a second home. I owe thanks to all the professors who contributed directly to this project through their teaching, particularly in graph theory: professors Ricardo, Diego, Francisco, Rudinei, Cynthia, Luis, and João Batista.

Furthermore, I also thank my IFSP professors Renato, André, Malu, Fábio and Jorge, who encouraged me to get into college. I am sure that without their support, specially from professor Renato who passed countless days helping me in physics, I would not be here.

To my examination board, I am deeply thankful for this opportunity. Professor Solange, the first member of the board, lectured me Artificial Intelligence, and I could learn so much from someone that always brought a positive outlook and saw the best in us. Maria Carolina Monard, who advised both Solange and my father, was instrumental in my father's journey and consequently mine. Though she is no longer with us, I dedicate this project to her in recognition of her incredible contributions to all our lives. It is also a privilege to have my dear friend André Fakhoury as the second member of the board. Thank you for everything, André—it's interesting that this project is about graphs, but you've been helping me with graphs even before I entered at USP, back when I would bother you in GEMA's group.

Lastly, I thank the love of my life, Isis, for making me happier every day, for all the care and understanding during this period, my friends for their unwavering support, and God, who has never abandoned me.

“All people are born as originals but many die as photocopies. ”
Carlo Acutis

ABSTRACT

FRANCO, J. L. **Graph ML for Flight Delay Prediction due to Holding Manouver**. 2024. 36p. Monografia (Trabalho de Conclusão de Curso) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

This project models the prediction of flight delays due to holding maneuvers as a graph problem, leveraging advanced Graph Machine Learning (Graph ML) techniques to capture complex interdependencies in air traffic networks. Holding maneuvers, while crucial for safety, cause increased fuel usage, emissions, and passenger dissatisfaction, making accurate prediction essential for operational efficiency. Traditional machine learning models, typically using tabular data, often overlook spatial-temporal relations within air traffic data. To address this, we model the problem of predicting holding as edge feature prediction in a directed (multi)graph where we apply both CatBoost, enriched with graph features capturing network centrality and connectivity, and Graph Attention Networks (GATs), which excel in relational data contexts. Our results indicate that CatBoost outperforms GAT in this imbalanced dataset, effectively predicting holding events and offering interpretability through graph-based feature importance. Additionally, a web-based tool, Airdelay, allows users to simulate real-time delay predictions, demonstrating the model's potential operational impact. This research underscores the viability of graph-based approaches for predictive analysis in aviation, with implications for enhancing fuel efficiency, reducing delays, and improving passenger experience.

Keywords: Graph Neural Networks. Graphs. Machine Learning. Complex Networks.

RESUMO

FRANCO, J. L. **Graph ML for Flight Delay Prediction due to Holding Manouver**. 2024. 36p. Monografia (Trabalho de Conclusão de Curso) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Este projeto modela a previsão de atrasos de voo devido a manobras de espera como um problema de grafo, aproveitando técnicas avançadas de Aprendizado de Máquina com Grafos (Graph ML) para capturar interdependências complexas em redes de tráfego aéreo. Manobras de espera, embora essenciais para a segurança, causam aumento no uso de combustível, emissões e insatisfação dos passageiros, tornando a previsão precisa essencial para a eficiência operacional. Modelos tradicionais de aprendizado de máquina, que geralmente utilizam dados tabulares, muitas vezes não consideram as relações espaço-temporais nos dados de tráfego aéreo. Para abordar essa questão, modelamos o problema da previsão de espera como uma predição de características de arestas em um (multi)grafo direcionado, onde aplicamos tanto o CatBoost, enriquecido com características de grafo que capturam centralidade e conectividade da rede, quanto Graph Attention Networks (GATs), que são eficazes em contextos de dados relacionais. Nossos resultados indicam que o CatBoost supera o GAT nesse conjunto de dados desbalanceado, prevendo eficazmente eventos de espera e oferecendo interpretabilidade por meio da importância dos recursos baseados em grafo. Além disso, uma ferramenta web chamada Airdelay permite que os usuários simulem previsões de atraso em tempo real, demonstrando o impacto operacional potencial do modelo. Esta pesquisa destaca a viabilidade de abordagens baseadas em grafos para análise preditiva na aviação, com implicações para aumentar a eficiência do combustível, reduzir atrasos e melhorar a experiência do passageiro.

Palavras-chave: Redes Neurais de Grafos. Grafos. Aprendizado de Máquina. Redes Complexas.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

GNN	Graph Neural Network
GAT	Graph Attention Network
ML	Machine Learning
GCN	Graph Convolutional Network
T-GCN	Temporal Graph Convolutional Network
GRU	Gated Recurrent Units
ICEA	Instituto de Controle do Espaço Aéreo / Airspace Control Institute
SCNN	Spectral CNN
METAR	METeorological Aerodrome Report
METAF	METeorological Terminal Aerodrome Forecasts
MLP	Multilayer Perceptron
XAI	Explainable AI
SVM	Support Vector Machine
SP	São Paulo
RJ	Rio de Janeiro
MG	Minas Gerais

CONTENTS

1 INTRODUCTION

The aviation industry increasingly relies on data-driven approaches to improve operational efficiency and reduce delays. Among the pressing challenges in air traffic management is the prediction of ‘holding’ maneuvers, where aircraft are instructed to delay landing due to factors such as airport congestion, adverse weather, or airspace limitations. While holding patterns are necessary for safety, they contribute to increased fuel consumption, emissions, and passenger dissatisfaction. This project aims to enhance the accuracy of holding predictions using machine learning (ML) models based on graph-structured data, specifically employing advanced methodologies in Graph Machine Learning (Graph ML) and Graph Neural Networks (GNNs).

Traditional machine learning applications in aviation have primarily focused on flight delay prediction and air traffic flow management. For instance, delay predictions based on weather conditions, airport congestion, and flight schedules have been widely studied (LAMBELHO *et al.*, 2020; GUI *et al.*, 2019). However, these models often rely on tabular data representations, which limit their ability to capture complex relational patterns among airports and other influencing factors. Additionally, research specifically focused on holding maneuvers is limited and generally lacks machine learning and network-based approaches that can model the spatial and temporal dependencies intrinsic to air traffic data (LEE *et al.*, 2020; SMITH; BATEMAN, 2008).

The use of graph-based machine learning methods is rapidly advancing in the field of intelligent transportation, where graph structures effectively capture complex spatial and temporal relationships across networks. A recent survey on GNNs in intelligent transportation systems, Rahmani *et al.* (2023), highlights their application across a variety of domains, including traffic forecasting, demand prediction, and urban planning. This survey underscores the power of GNNs in applications where data is inherently interconnected, such as autonomous vehicle routing and intersection management. By organizing studies within these domains, they identify distinct opportunities and challenges, particularly in multi-modal models and reinforcement learning applications. Similarly, Zhao *et al.* (2020) demonstrate the value of GNNs in the specific context of real-time traffic forecasting with their T-GCN (Temporal Graph Convolutional Network) model. By combining Graph Convolutional Networks (GCN) and Gated Recurrent Units (GRU), the T-GCN model captures both spatial and temporal dependencies, achieving state-of-the-art accuracy in urban traffic prediction tasks. These studies exemplify the increasing role of GNNs in transportation-related decision-making, showing potential for improved accuracy and efficiency in complex, dynamic systems.

The project employs two main approaches:

- **Tabular-based approach:** We use the CatBoost model, leveraging graph features—such as centrality and connectivity metrics—that capture the significance of directed edges (flights) within the network (PROKHORENKOVA *et al.*, 2018).
- **Graph Attention Network (GAT) approach:** We compare with GATs that have proven effective in applications where relational data is essential, making them a promising choice for capturing the interconnected nature of air traffic (VELIČKOVIĆ *et al.*, 2018).

The contributions of this project are twofold. First, we introduce a novel application of graph-based ML models for predicting holding events, providing a refined view of airport interdependencies. Second, by comparing CatBoost with the GAT model, we evaluate which approach better captures the graph topology of the air traffic and have a better predictive performance. This research has potential implications for improved fuel efficiency, reduced delays, and enhanced passenger experiences by refining model selection and feature engineering strategies tailored to aviation applications. For the best of our knowledge there is no previous work that approached the problem of flight delay due to holding maneuver using Graph ML.

The structure of this work is as follows. In Section 2, we review relevant literature and build the theoretical background needed. Section 3 details the dataset and the modelling. Section 4 discusses the experimental setup, with results and comparison analysis presented and model deployment. Finally, Section 5 provides concluding remarks about my Computer Science Bachelor's.

2 THEORETICAL FRAMEWORK AND RELATED WORKS

Graph machine learning can be tracked backwards to the problem of 'learning' on data that is inherently a graph (SILVA; ZHAO, 2016; CHAMI *et al.*, 2022) or can be modeled as a graph (VERRI; ZHAO, 2013; YOU *et al.*, 2020). This field encompasses a variety of tasks, including node/edge classification, network construction, link prediction, graph classification, graph cut/partitioning, network embeddings, graph coarsening/reduction, which rely on learning representations from graph-structured data. Over the last decades, researchers have developed numerous approaches to tackle these challenges, initially these techniques were most developed by complex networks researchers. However, in the last decade with the advancements in deep learning, the field has seen a significant shift towards the merging of three main communities: graph signal processing, deep learning and complex nets.

As described, defining the field of graph machine learning is not straightforward, as it encompasses a broad range of methods and applications. The tasks mentioned above are just a few examples of the many challenges that can be addressed through graph-based learning techniques. For clarity, these tasks can be categorized into three main learning paradigms: supervised, unsupervised, and semi-supervised learning. In this study, we are interested on the (semi-)supervised learning paradigm, which encompasses a variety of techniques designed to leverage learning to (partially-)labeled data (VERRI; URIO; ZHAO, 2018; AMANCIO *et al.*, 2014). But we can refine even more, in fact, this work will focus in the subset of graph elements prediction(classification/regression) methods.

In this chapter, we provide an overview of the theoretical framework of graph machine learning for node/edge prediction. Here we consider the division of the field into `classical` graph learning and `deep` graph learning, where here 'classical' refers to the machine learning techniques applied to graphs before the advent of graph neural networks, where standard ML algorithms were applied to graph data and the topological information measures were encoded as features together with the tabular data (COSTA, 2007; SILVA; ZHAO, 2016). This bipartition is what will pave the way of our explanation, since the last decade has seen a complex interplay between these two approaches. The field's evolution can be traced back to when Bruna *et al.* (2013) introduced one of the first GNN architectures leaned on the theory of graph signal processing. Concurrently, researchers were developing node embedding techniques like DeepWalk (PEROZZI; AL-RFOU; SKIENA, 2014) and node2vec (GROVER; LESKOVEC, 2016), which bridged classical and deep approaches while remaining using complex networks concepts. The subsequent years saw a surge in GNN architectures, including Graph Convolutional Networks(GCNs) (KIPF; WELLING, 2016) and GraphSAGE (HAMILTON; YING; LESKOVEC, 2017), marking a shift towards more sophisticated deep learning approaches for graphs and the unification of the field.

In the following sections, we explain each subset, their theory and applications, and how they have evolved over time. We also discuss the challenges and limitations of these methods.

2.1 Classical graph learning

These early efforts focused on shallow learning techniques such as feature engineering, graph traversal algorithms, and spectral methods, which laid the foundation for understanding graph structure and dynamics. Methods like community detection, centrality measures, and link prediction (SILVA; ZHAO, 2016) became key tools for analyzing large-scale networks in areas such as social science, biology, and infrastructure systems (NEWMAN, 2018; BOCCALETTI *et al.*, 2006). A significant focus of these techniques was to develop graph-based features that could be integrated into traditional machine learning models, effectively transforming graph data into a format compatible with standard algorithms like logistic regression, decision trees, and support vector machines. By encoding graph topology through hand-crafted features, such as connectivity and centrality, researchers could leverage these features for tasks like classification, regression, and clustering in tabular machine learning frameworks.

Among these features, centrality measures became particularly important due to their ability to capture the relative importance or influence of nodes in a graph, not just nodes (BONACICH, 1987), but other graph elements such as edges (LU; ZHANG, 2013; BRÖHL; LEHNERTZ, 2019) and hyperedges (TUDISCO; HIGHAM, 2021). Centrality measures, such as degree, betweenness, and closeness, served as input features in machine learning pipelines, helping to predict outcomes based on the structural role of nodes within the network.

Spectral centrality, particularly eigenvector centrality (BONACICH, 1987), has proven valuable in machine learning applications due to its ability to identify globally influential nodes. Eigenvector centrality assigns a score to each node by considering not only its direct connections but also the centrality of its neighbors, which results in a recursive definition. Mathematically, the eigenvector centrality x of a node in a graph can be defined as the solution to the equation $Ax = \lambda x$, where A is the adjacency matrix of the graph, and λ is the largest eigenvalue, thus x is the eigenvector associated with the largest eigenvalue. This relationship arises from the fact that the centrality of a node is proportional to the sum of the centralities of its neighbors, if we normalize the adjacency we get an stochastic matrix and then $\lambda = 1$ is the largest eigenvalue, named the `Perron vector`. The eigenvector centrality captures both local and global structure in a network, making it a powerful feature for tasks such as node classification, ranking, and recommendation systems. A related and widely used spectral measure is PageRank

(BRIN, 1998), which extends the idea of eigenvector centrality by introducing a damping factor to model random surfing behavior,

$$PR(v) = \frac{1-d}{N} + d \sum_{u \in \mathcal{N}(v)} \frac{PR(u)}{\deg(u)},$$

where $PR(v)$ is the PageRank score of node v, d is the damping factor, and $\mathcal{N}(v)$ represents the neighbors of node v. This iterative computation converges to a stationary distribution of scores, which can be interpreted as the probability of landing on a given node after a long random walk, in this sense the `Perron vector` signifies the convergence of the process in the infinite. PageRank has been widely used in ranking tasks, such as identifying important websites in search engines or recommending influential users in social networks.

However, these spectral-based centralities come with limitations. Eigenvector centrality requires the computation of the principal eigenvector of the adjacency matrix, which involves finding the largest eigenpair problem. This has a time complexity of $\mathcal{O}(n^2 d)$ for exact methods, where n is the number of nodes in the graph and d is the ratio of convergence for the power method. Furthermore, spectral methods can suffer from limitations rooted in the Perron-Frobenius theorem, which guarantees the existence of a unique largest eigenvalue only for irreducible, non-negative matrices. For graphs that are disconnected or have negative weights, these conditions are violated, and the eigenvector centrality may not be well-defined or interpretable. That is, the adjacency matrix should be non-negative and irreducible, where we could use the Perron test $\sum A^k > 0$ to see if the graph is strongly connected. These centralities also tend to be node-centric, lacking a direct extension to edge importance. For edge centrality, betweenness remains crucial, particularly in directed graphs, where the structural role of links (edges) must be considered to capture flow dynamics. Additionally, spectral centralities can be sensitive to noise and small perturbations in the graph structure, leading to instability in the centrality scores. Despite these challenges, spectral centrality remains a powerful tool for machine learning tasks that benefit from capturing global graph structure, provided that the computational and stability issues can be managed.

2.2 Deep graph learning

The rise of deep learning has revolutionized the field of graph machine learning, enabling the development of more powerful and scalable models for graph data. Graph neural networks can be divide in two main categories: spectral-based and spatial-based. Here is a trick thing, the GCN architecture (KIPF; WELLING, 2016) is commonly divulgated as a spatial-based method, since it is more intuitive talking about the convolution operation in the spatial domain, where we simply aggregate information from the immediate neighbors. However, the GCN is a spectral-based method, in fact, it can be thought as a simplification of the first spectral GNN (BRUNA *et al.*, 2013) proposed and that builds the math behind

GCNs. That said, first we introduce the spectral-based GNNs and then the spatial-based ones.

2.2.1 Spectral-based GNNs

Spectral methods are rooted in graph signal processing. The core idea is that a signal on a graph can be represented as node features, where each feature vector at a node corresponds to a 'signal' defined over the graph. In this context, the graph Laplacian $\mathcal{L} = D - A$, where D is the degree matrix and A is the adjacency matrix, plays a crucial role. It captures the structure of the graph and can be used to perform operations analogous to Fourier transforms in classical signal processing. Spectral methods can be categorized into two types: eigenvalue-based, where the focus is on creating a graph filter in the Fourier domain, and eigenvector-based, where the goal is to use a spectral basis to decompose the signal (BO *et al.*, 2023).

Bruna *et al.* (2013) introduced the first spectral Graph Neural Network (GNN), termed the Spectral CNN (SCNN), which aimed to translate ideas from standard Convolutional Neural Networks for images to graphs. The SCNN leverages the spectral decomposition of the graph Laplacian $\mathcal{L} = U\Lambda U^T$ to define a filter convolution operation in the Fourier domain. In this framework, the graph Fourier transform of a signal f is represented as $\hat{f} = U^T f$, and the convolution operation $(\star)$ is defined as $g_\theta \star f = U g_\theta U^T f$, where g_θ is a learnable filter parameterized by θ. While powerful, the SCNN faces significant challenges: it requires $\mathcal{O}(n^3)$ computational complexity to calculate the entire graph spectrum, which is prohibitively expensive for large graphs. Moreover, the non-localized nature of eigenvectors means global information can overshadow local structural details, leading suboptimal balance between local and global information aligned with a huge parameter complexity (CHEN, 2020).

To address these limitations, ChebNetDefferrard, Bresson and Vandergheynst (2016) introduces Chebyshev polynomials to approximate spectral filters, effectively reducing computational complexity while preserving the ability to capture localized patterns in the graph structure. The main ideia is to redefine our previous filtering operation to $g_\theta(\mathcal{L})f = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{\mathcal{L}})f$, where $T_k(\widetilde{\mathcal{L}}) =$ is the Chebyshev polinomial of order k evaluated at the scaled Laplacian $\widetilde{\mathcal{L}} = 2\frac{\mathcal{L}}{\lambda_{\max}} - I_n$. This innovation not only makes spectral GNNs more scalable to larger graphs, since we just need to calculate the first eigenpair ($\mathcal{O}(n^2)$ through the power method) for the approximations, but also enhances their ability to balance local and global information processing. In fact, the filters are K-localized for polinomials of order K, that is intuitive by remembering that $\mathcal{L}^K$ represents the paths with length less or equal to K. The ChebNet laid the foundation for GCNs (KIPF; WELLING, 2016). Although GCNs are commonly referred to as spatial methods, their underlying principle is rooted in the truncation of the Chebyshev expansion to $K = 1$, which limits the filter to first-order neighbors. This simplification reduces computational complexity significantly while preserving effectiveness. Instead of requiring the full spectral decomposition of the Laplacian matrix, GCNs use a localized approximation of the graph convolution, expressed as: $g_\theta \star f \approx \theta(I_n + \widetilde{A})f$, where $\widetilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized adjacency matrix, where A is the adjacency matrix, and D is the degree matrix. This approximation results in an efficient propagation rule that aggregates information from a node's immediate neighbors while updating the node's features. This propagation mechanism is often confused as a spatial method because it effectively propagates information from adjacent nodes—akin to a spatial neighborhood aggregation. Although its already a simple model, results have shown that GCNs can achieve state-of-the-art performance on a variety of tasks with even more simplifications (WU *et al.*, 2019). However, as we can note, all these spectral methods works just in undirected graphs, since it needs the spectral decomposition. Furthermore, these methods are 'node centric', that is, they focus just on node features and the topology of the nodes, most of this is because the adjacency matrix maps the dimension of nodes to nodes, thus leaving *edge features* out of the scene.

2.2.2 Spatial-based GNNs

Spatial-based GNNs differ from spectral-based approaches by directly leveraging the graph structure to perform convolutions in the spatial domain, rather than relying on the spectral decomposition of graph operators like the Laplacian. In spatial-based methods, the convolution operation is interpreted as an aggregation of node features from a node's local neighborhood, akin to how standard convolutional neural networks aggregate pixel information from nearby regions in image data. These methods operate by iteratively updating node representations by propagating information between neighboring nodes, making them intuitive and highly scalable for large-scale graphs.

The general framework for message passing in spatial-based GNNs can be described as follows. For each node i, at layer t, we aggregate the features of its neighbors $\mathcal{N}(i)$ to produce an updated node embedding: $\mathbf{m}_i^{(t+1)} = \text{AGGREGATE}^{(t)}\left(\left\{\mathbf{h}_j^{(t)} : j \in \mathcal{N}(i)\right\}\right)$, where $\mathbf{h}_j^{(t)}$ is the feature of neighboring node j at layer t. Then, we update the node i's representation: $\mathbf{h}_i^{(t+1)} = \text{UPDATE}^{(t)}\left(\mathbf{h}_i^{(t)}, \mathbf{m}_i^{(t+1)}\right)$, where $\text{AGGREGATE}^{(t)}$ is a neighborhood aggregation function, and $\text{UPDATE}^{(t)}$ is the node update function.

The general idea behind spatial-based GNNs is that, for each node, we aggregate the features of its neighbors to produce an updated node embedding. A key example of this is the GraphSAGE architecture (HAMILTON; YING; LESKOVEC, 2017), which computes node representations by sampling and aggregating features from the node's neighbors. The GraphSAGE model employs several types of aggregation functions, including mean, LSTM-based, and pooling aggregators, which allow for flexible and inductive learning on large graphs. In particular, GraphSAGE enables the generation of embeddings for unseen nodes, making it suitable for inductive learning tasks, where the model needs to generalize to new nodes that were not present during training. Unlike spectral-based methods, which are constrained to a fixed graph size and structure due to their reliance on the graph Laplacian, spatial-based GNNs are inherently more flexible and can be applied to dynamic and evolving graphs. These models perform neighborhood aggregation locally, and therefore do not require the global knowledge of the graph structure that spectral methods need. This flexibility makes them particularly useful for large-scale graphs and for graphs where the structure may change over time, such as social networks or knowledge graphs.

Another prominent spatial-based GNN is the Graph Attention Network (GAT) (VELIČKOVIĆ *et al.*, 2018), which introduced attention mechanisms into graph learning. GAT models learn to assign different weights to the neighbors of a node, allowing the model to focus more on the most relevant neighbors during the feature aggregation process. This is achieved using a self-attention mechanism, where the importance of neighboring nodes is learned through a shared attention coefficient, $e_{ij} = \text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i||\mathbf{W}\mathbf{h}_j])$, where e_{ij} represents the attention coefficient between nodes i and j, $\mathbf{W}$ is a learnable weight matrix, $\mathbf{h}_i$ and $\mathbf{h}_j$ are the feature vectors of nodes i and j, and $||$ denotes concatenation. The attention coefficients are then normalized across all of a node's neighbors using the softmax function, $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}(i)}\exp(e_{ik})}$, this normalization ensures that the attention coefficients sum to 1, allowing the model to perform a weighted aggregation of the neighbors' features, $\mathbf{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}\mathbf{W}\mathbf{h}_j\right)$, here $\mathbf{h}'_i$ is the updated representation of node i, and σ is a non-linear activation function. By learning attention coefficients, GATs can capture both the importance and the structure of the graph, making them particularly effective in tasks where the relationships between nodes are not equally important, such as in citation networks or social media graphs.

3 MATERIALS AND METHODS

This chapter details the materials and methods used in our study. Specifically, we will cover the dataset, the model based on CatBoost using graph-derived features, and our approach using Graph Attention Networks (GATs) for predictive modeling. Our dataset includes a range of meteorological, geographical, and flight variables.

Our objective is to detail and model how to predict if a given aircraft is going to delay due to holding maneuver in a supervised learning setting.

3.1 Datasets

The study utilizes two distinct datasets, each containing 42,336 observations, with identical meteorological, geographical, and flight-related features. These datasets were constructed from a combination of METAR and METAF weather reports, airport and flight specifications sourced from ICEA. Each dataset serves a different predictive modeling purpose—binary classification and regression—and includes tailored labels to reflect these objectives

One of the primary challenges in the classification dataset is its class imbalance, which could impact model performance. Common approaches to address imbalance, such as oversampling or undersampling, have been shown to introduce limitations like overfitting and poor generalization ((ZHAO; ZHANG; WANG, 2021), Graph SMOTE). For graph machine learning tasks, oversampling is generally problematic due to the risk of introducing artificial connectivity patterns, while undersampling can lead to loss of critical structural information. Therefore, we opted to explore model-based techniques without applying these rebalancing methods.

3.1.1 Binary Classification Dataset

The binary classification dataset is used to predict the likelihood of a holding maneuver occurring for a given flight. In this dataset, the label is a binary value:

- **Class 0 (No Holding)**: Represents flights with no holding delays, comprising 41,616 samples.
- **Class 1 (Holding)**: Indicates flights with holding delays, comprising only 720 samples.

This significant imbalance between the classes adds complexity to the classification task, as the model needs to accurately predict a rare event within the data.

3.1.2 Regression Dataset

The regression dataset aims to predict the exact duration of holding delays in seconds, thus providing a continuous label for each observation:

- **Holding Time (Seconds)**: For this dataset, each holding event is represented by a floating-point number indicating the holding time in seconds. This approach enables a finer-grained analysis and can potentially improve operational insights by quantifying delay duration rather than merely classifying its occurrence.

3.1.3 Meteorological Features

The dataset includes a comprehensive range of meteorological variables from both METAR and METAF reports:

- **Wind Direction:** `metar_wind_direction`, `metaf_wind_direction`
- **Wind Speed:** `metar_wind_speed`, `metaf_wind_speed`
- **Wind Gusts:** `metar_wind_gust`, `metaf_wind_gust`
- **Visibility:** `metar_visibility`, `metaf_visibility`
- **Cloud Coverage:** `metar_cloudcover`, `metaf_cloudcover`
- **Temperature:** `metar_temperature`, `metaf_temperature`
- **Dew Point:** `metar_dewpoint`, `metaf_dewpoint`
- **Altitude:** `metar_elevation`, `metaf_elevation`
- **Sky Levels:** `metar_skylev1`, `metar_skylev2`, `metar_skylev3`, `metar_skylev4`, `metaf_skylev1`, `metaf_skylev2`, `metaf_skylev3`, `metaf_skylev4`
- **Altimeter Setting:** `metar_altimeter`, `metaf_altimeter`
- **Weather Symbols:** `metar_current_wx1_symbol`, `metar_current_wx2_symbol`, `metar_-current_wx3_symbol`, `metaf_current_wx1_symbol`, `metaf_current_wx2_symbol`, `metaf_current_wx3_symbol`

3.1.4 Geographical Features

The geographical features include variables based on flight paths and airport information:

- **Flight Distance:** Calculated as the geodesic distance between departure and arrival airports.

- **Airport Altitude:** `departure_altitude` and `arrival_altitude`, reflecting the elevation of the airports.
- **Latitude and Longitude:** `departure_latitude`, `departure_longitude`, `arrival_-latitude`, and `arrival_longitude` for geolocation-based analysis.

3.1.5 Flight-Specific Features

These features capture specific characteristics related to the flight and any runway head changes:

- **Previous Runway Head Change:** `prev_troca_cabeceira`
- **Runway Head Change in Previous Hour:** `troca_cabeceira_hora_anterior`
- **Flight Hour:** `hora_do_voo`

3.2 CatBoost with Graph Features

This study employs the CatBoost model, a high-performance gradient boosting library, chosen specifically for its ability to handle categorical features and class imbalance effectively, as well as for its robust handling of noisy data (PROKHORENKOVA *et al.*, 2018). CatBoost has been widely recognized for its superior performance in structured data problems, particularly when compared to other boosting algorithms like XGBoost and LightGBM, thanks to its unique techniques such as ordered boosting and categorical feature encoding. These innovations help prevent overfitting and enhance generalization in class unbalanced problems.

Here, we describe how CatBoost is combined the graph-based features that are extracted of our modeled airports network. These features are derived from the weighted directed graph and enconded as tabular features that are used as input to the model as we describe in the following sections.

3.2.1 Graph Representation of the Flight Network

To model the interactions in flight data, we represented the problem as a directed graph, depicted in Figure 1, where each node represents an airport, here we represent the airports as states of Brazil: SP (São Paulo), MG (Minas Gerais), RJ (Rio de Janeiro). In this network:

- Nodes represent airports.
- Directed edges represent flights, with each edge directed from the departure airport to the destination airport.

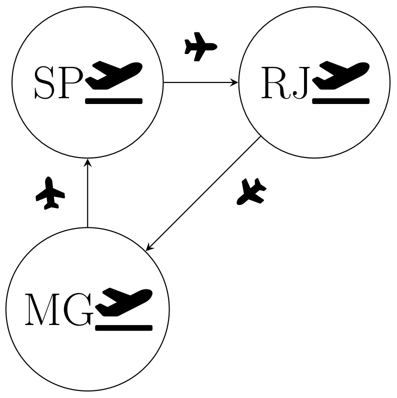


Figure 1 – Airports and Directed Flights

Given the frequent occurrence of multiple flights between the same pairs of airports (i.e., multiedges), we have in fact a multigraph, however we abstract it into a weighted directed graph as shown in 2. Here, each edge's weight corresponds to the total number of flights between a specific pair of airports, transforming multiple directed edges into a single weighted edge. This abstraction allows us to calculate key network metrics more easily, which we then used as features in the CatBoost model.

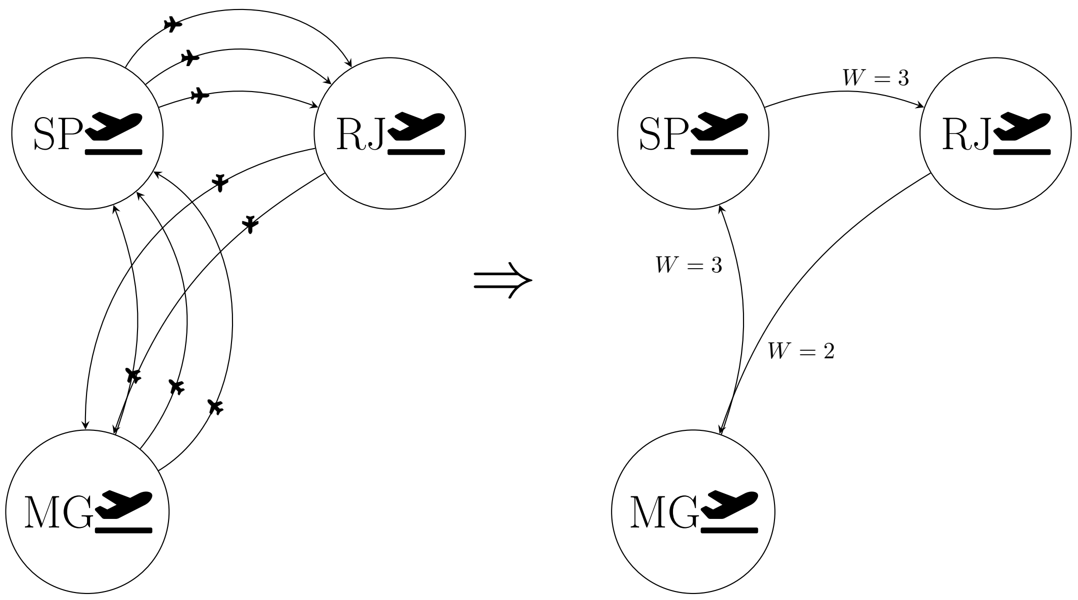


Figure 2 – Transformation of a multigraph of flights into a weighted directed graph. The multigraph (left) represents multiple flights between airports. In the weighted graph (right), edges are aggregated to show total flights as weights.

3.2.2 Graph-based Features

The graph-based features encode essential structural information about the flight network, capturing connectivity, centrality, and robustness. These features are crucial for understanding the influence of each airport within the network and its potential impact on flight holding patterns.

Although we have already made this simplification of the multigraph, transforming it into a weighted directed graph, we still need to extract the features from the graph and encode them as tabular data. However, this is not straightforward, as the graph measures are not directly compatible with the model.

The modelling will impact dramatically in the resulting graph-based features. For instance, we need to calculate edge measures, but this is not so explored as node measures, so the lack of possibilities is a challenge to be overcome. Another challenge is the direction, that is, we have to create edge measures in a directed weighted graph, which is hard, as we detailed in section 2.1, because most of the complex networks measures proposed are 'node centric' and for undirected graphs.

With this in mind, we can observe why the weighted graph transformation was so important, since the measures available for our setting are strongly dependent to the weight (as we will detail later), and our graph is almost totally connected, so in undirected unweighted setting they would be approximately equal, leaving no information. The following graph metrics were calculated from the weighted directed graph:

- **Betweenness Centrality:** Captures the relative importance of each airport in terms of the routes it controls within the network. Higher values indicate airports that serve as critical transit points.
- **Flow Betweenness:** Highlights the flow dynamics of connections, showing how flights tend to route through certain airports, which may correlate with congestion.
- **Edge Connectivity:** Indicates the robustness of airport connections, with higher values signifying more resilient routes between airports that could better handle rerouting needs.
- **Degree Difference:** Measures the disparity between in-degrees and out-degrees at each node, helping to identify key hubs or spokes in the network.
- **Google Matrix:** Based on PageRank centrality, the Google matrix provides a probabilistic transition representation for each airport, which reflects both local and global connectivity.

As we can see, these features are not commonly used in the literature. Here is where the weighted network plays a crucial role, edge betweeness centrality (NEWMAN; GIRVAN, 2004) is constructed using shortest paths in the network, thus the weight will be crucial part of it, since without it the graph is almost fully connected, the shortest path will be almost the same for all pairs of nodes. The same happens with flow betweeness centrality (FREEMAN; BORGATTI; WHITE, 1991), that is a measure based on electrial circuits Kirchoff law, more specifically, instead of working with shortest paths, it use the maximum flow that pass through each edge and the weight visualized as capacity will be crucial to calculate it.

The edge connectivity is a measure of the minimum number of edges that must be removed to disconnect the graph, and the weight will be crucial to calculate it. The degree difference we stated here as a measure of the difference between the in-degree and out-degree of a node. The Google matrix is a way we derived to keep using PageRank for edges. In fact, as we detailed in section 2.1, althought the PageRank centrality could be applied in our graph, since it satisfieis the Perron theorem as it is always postivie and strongly connected, it is a node measure, so we have to adapt it to edges, and the Google matrix is a way to do it.

These features enhance the CatBoost model by embedding graph-theoretic insights into its predictive capabilities, ultimately enabling a more nuanced understanding of how network dynamics relate to flight holding patterns.

3.3 Graph Attention Network

As we previously described, the GAT model in section 2.2.2 has a large range of applications, from drug discovery to fake news detection (SOUZA *et al.*, 2024). The GAT model leverages the underlying graph structure but does not rely on explicitly computed graph-derived features like the CatBoost model does. Instead, it learns node representations in an end-to-end manner, enabling the model to capture the relationships between airports and flights directly from the data.

The modelling of a GNN for our problem is a challenging task, as we have to adapt the model to predict edge features, since ‘holding’ is an edge feature in our setting. In section 2.2.1 we detailed why the spectral-based GNNs are not suitable for our setting, as they are not able to handle edge features and direction, due to their ‘node-centric’ approach based on the adjacency matrix. Although spatial-based GNNs can handle direction in their majority, they are not able to handle edge features in general, since they need to create a way to aggregate the edge features with the neighbors’ features.

The GAT model is so used because it is highly adaptable in pratically any graph setting. As we will show, the attention mechanism detailed in section 2.2.2 can be generalized to handle edge features, and the model can be adapted to predict edge features. In fact, a simple concatenation ($||$) in the attention formula already gives us this power ,

$$\alpha_{ij} = \sigma(\phi_1(\mathbf{a}^T[Wh_i||Wh_j||W_2e_{ij}])) \quad ,$$

where e_{ij} are the edge features, h_i and h_j are the node features, and W and W_2 are the weight matrices. This formula allows the model to focus on the relevant neighboring nodes, making it ideal for relational data. In our case, the edge features are the tabular data features with holding being part of them, which is the target we want to predict. This mechanism is demonstrated in Figure 3.

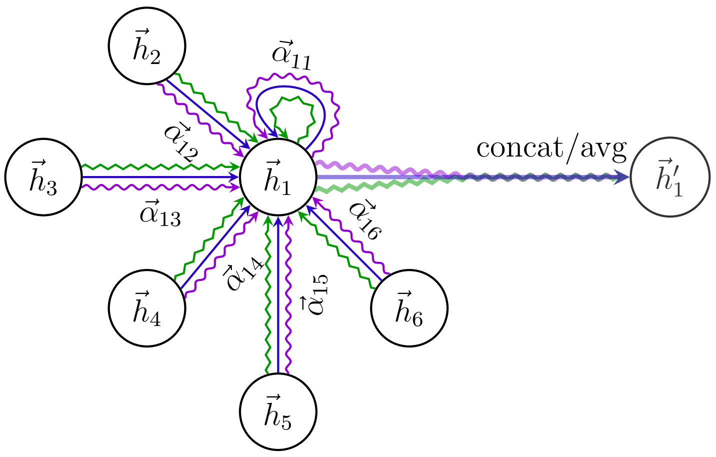


Figure 3 – GAT Layer with multi-head attention. Sourced from (VELIČKOVIĆ *et al.*, 2018).

Furthermore, the directed multigraph setting we described in section 3.2 is not a problem for the GAT model, since it can handle multiple edges between the same pair of nodes, as we will show in the following sections. We show how we model the GAT to be a directed multigraph representing the flights and their features in Figure 4.

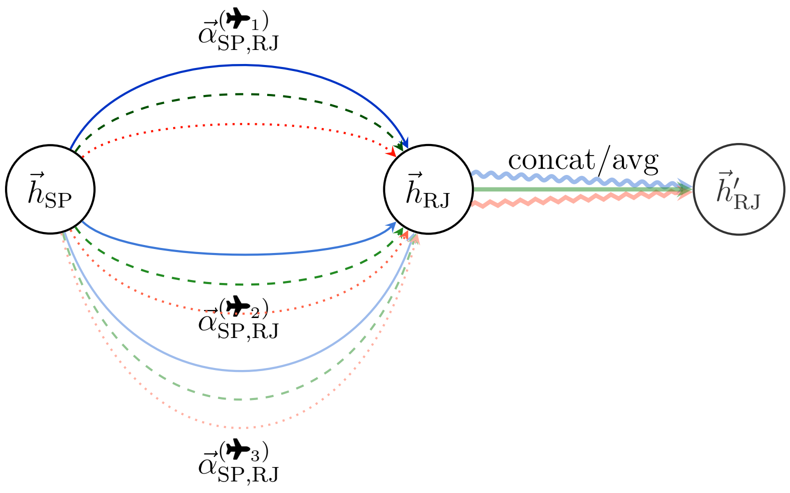

$\vec{\alpha}^{(✈_1)}_{\mathrm{SP,RJ}}$
$\vec{h}_{\mathrm{SP}}$
$\vec{h}_{\mathrm{RJ}}$
concat/avg
$\vec{h}'_{\mathrm{RJ}}$
$\vec{\alpha}^{(✈_2)}_{\mathrm{SP,RJ}}$
$\vec{\alpha}^{(✈_3)}_{\mathrm{SP,RJ}}$

Figure 4 – Airport multigraph GAT Layer with multi-head attention for three different flights between nodes (SP,RJ), with alternating colors opacity for each flight.

Finally, the last layer of our predictor would be to pass the learned node embeddings h_i and h_j with the edge feature $e_{ij}^{(k)}$ of the flight k to a fully connected layer (MLP) to predict the holding of the flight k. That is, we simply concatenate them, and after the MLP layer, we have a sigmoid σ activation function that outputs the prediction $\hat{y}_k$ of holding,

$$\hat{y}_k = \sigma(\mathrm{MLP}(h_i || h_j || e_{ij}^{(k)})) \quad .$$

4 RESULTS

This chapter presents a comprehensive evaluation of the classification performance for the models applied in this study, with a focus on comparing the CatBoost and Graph Attention Network (GAT) models under varying layer configurations. Given the dataset's imbalance, multiple performance metrics are reported to provide a more nuanced assessment than accuracy alone.

4.1 GAT vs. CatBoost Classification Performance

The performance of each model in predicting flight delays due to holding maneuvers was assessed using the following metrics:

- **Accuracy**: Measures the overall proportion of correct predictions across all instances. While accuracy is a useful baseline metric, it can be misleading with imbalanced datasets, as it may remain high even if the model fails to correctly predict instances of the minority class, such as delayed flights.
- **Precision**: Represents the ratio of true positive predictions (correctly predicted delays) to all positive predictions made by the model. Precision provides insight into the model's ability to accurately identify actual delays, where higher precision indicates fewer false positives.
- **Recall**: The ratio of true positive predictions to the total number of actual positive instances (delays) in the dataset. A high recall indicates the model's effectiveness in capturing most delayed flights, crucial in applications where missing delay predictions could impact operations.
- **F1-Score**: The harmonic mean of precision and recall, which balances these two metrics into a single score. The F1-score is particularly useful for imbalanced datasets, as it considers both false positives and false negatives, providing a balanced measure of performance.

The performance of each model configuration across these metrics is shown in Table 1, highlighting the trade-offs between model complexity (number of GAT layers) and classification effectiveness. Notably, the CatBoost model achieves balanced performance, which is advantageous given the imbalanced dataset.

As we can see, Catboost outperforms GAT in terms of precision and F1-Score, while GAT models with more layers tend to have higher recall and with a single layer the higher accuracy. However, the 30-layer GAT model exhibits a significant drop in accuracy,

Table 1 – Performance metrics for various GAT layer configurations and CatBoost with graph features.

Model	Test Accuracy	Precision	Recall	F1-Score
CatBoost	0.91	**0.10**	0.56	**0.17**
1 GAT Layer	**0.9483**	0.0288	0.0625	0.0395
3 GAT Layers	0.5168	0.0143	0.3975	0.0277
5 GAT Layers	0.5736	0.0123	0.2992	0.0237
10 GAT Layers	0.9067	0.0167	0.0787	0.0276
30 GAT Layers	0.0181	0.0177	**0.9973**	0.0347

indicating overfitting. The CatBoost model, with a balanced performance across all metrics, is better suited for the imbalanced dataset, as it captures both delayed and non-delayed flights effectively.

Since we are dealing with an unbalanced setting, it was already expected that the CatBoost model would perform better, as it is a decision tree ensemble model that is known for its robustness in handling class imbalance. That is why it had the higher F1-score that is in our problem the most reliable metric, since we want to predict holding and it's better to be wrong some times than always predict 'no holding'. The GAT model, on the other hand, as most of GNNs have trouble in dealing with class imbalance, as they tend more to overfitting.

4.2 Predictive Regression Analysis and Interpretability

Beyond classification, we extended our investigation to evaluate the CatBoost model's capacity for regression on continuous delay values and its interpretability. The objective was to assess how effectively CatBoost could predict the extent of flight delays while providing insights into which features were most influential in the model's decision-making. This section explores both aspects—interpretability and regression accuracy—with a focus on the model's performance and its potential operational impact.

4.2.1 Interpretability of the CatBoost Model

One of the major advantages of the CatBoost model is its transparency, particularly when compared to more complex neural network architectures. By utilizing graph-based features, CatBoost not only achieved favorable classification performance but also provided a clearer view of feature importance. Unlike black-box models, CatBoost can be examined through Explainable AI (XAI) techniques, which help interpret how specific features contribute to predictions.

Figure 5 presents the feature importance for the CatBoost model, identifying which graph-based features most significantly influence the prediction of flight delays due to holding maneuvers. This insight confirms the utility of graph-based features in providing a robust predictive foundation and highlights their relative impact on the delay predictions.

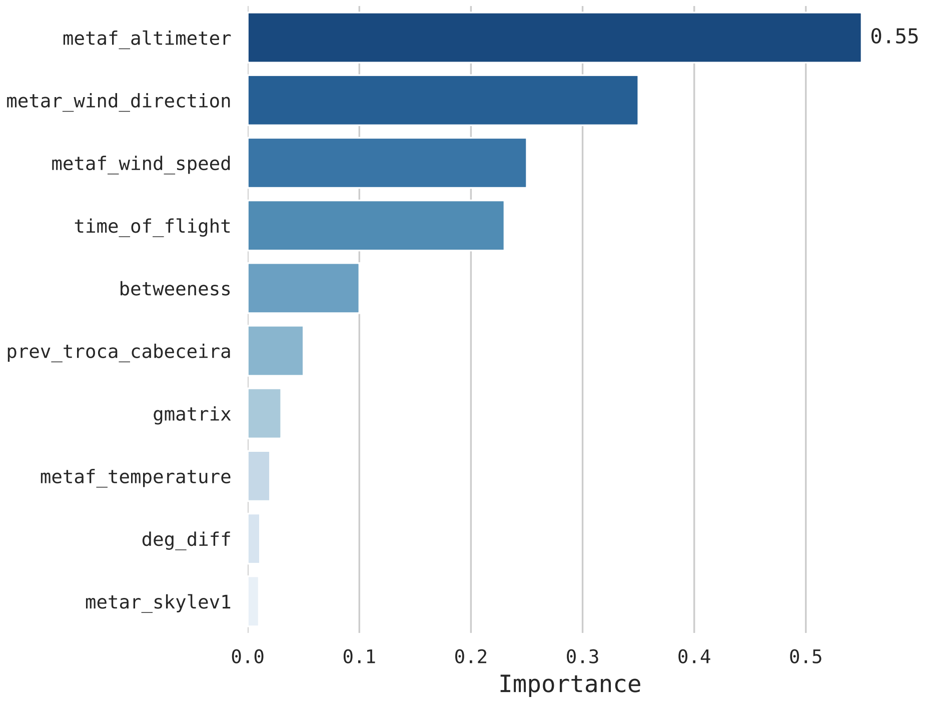


Figure 5 – Feature importance for the CatBoost model on the airport network dataset, indicating the relevance of graph-based features.

4.2.2 Regression Performance

To further assess the model's robustness, we applied CatBoost to a regression task, predicting continuous delay values rather than binary classifications. Figures 6 and 7 display the distribution of predicted (y_{pred}) and actual (y_{test}) delay values, respectively. By comparing these distributions, we gain insights into CatBoost's ability to capture trends and variability within the dataset.

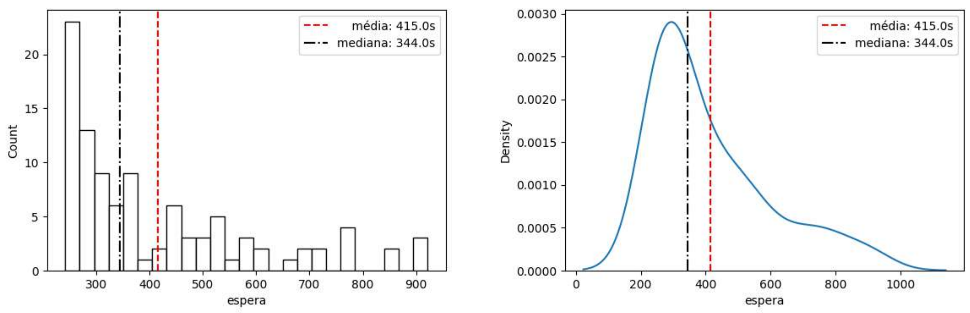


Figure 6 – Predicted delay values distribution (y_{pred}) for the regression task using CatBoost.

The comparison between predicted and actual distributions reveals that CatBoost captures core trends in the data, though some deviation at extreme delay values suggests

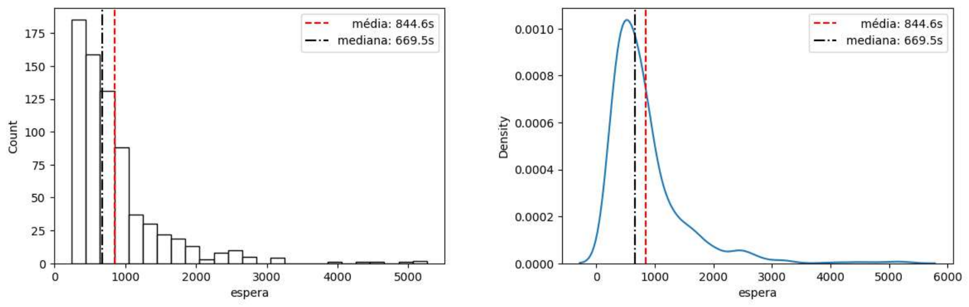


Figure 7 – Actual delay values distribution (y_{test}) for the regression task in the test set.

potential areas for improvement. Overall, CatBoost's performance in regression further validates its flexibility and predictive power, offering a promising model for continuous delay predictions in airport network analysis.

4.3 Deployment and Implementation

The source code for this project is available on GitHub at https://github.com/graph-learning-ita/airnet-holding-ml/.

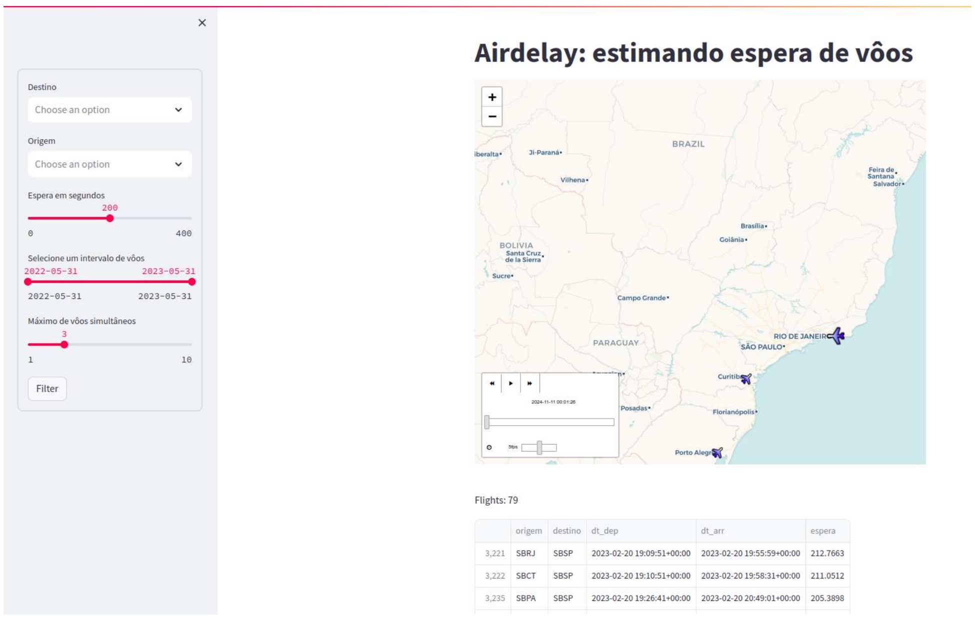
We have also developed a web-based simulation tool using Folium and Streamlit, which visualizes flight delays as predicted by the CatBoost model. Users can specify a simulation period, during which the model's predictions guide the flight paths and indicate holding maneuvers in real time, enabling stakeholders to observe potential delay scenarios. This interactive platform enhances the practical application of the model by providing a visual, user-friendly interface to interpret predictions dynamically.

We called the application 'Airdelay' and it is available at Airdelay.com. Figure 8 shows a screenshot of the Airdelay tool, illustrating the predicted flight delays due to holding maneuvers. Users can interact with the map, explore different scenarios, and observe the model's predictions in real time, enhancing their understanding of the model's performance and potential operational impact.

4.4 Discussion, Limitations and Future Works

The results presented in this chapter provide insights into the trade-offs between the CatBoost model and various GAT configurations. CatBoost, with its interpretable, tabular-focused approach, offers a more reliable model for imbalanced data, outperforming GAT in both precision and recall. GAT models with additional layers did not significantly improve performance and, in some cases, led to overfitting or instability due to the network's complexity and the dataset's limitations.

By employing graph features in a gradient boosting framework, we have shown that structured data representations, combined with graph-theoretic metrics, can in some cases



Destino: Choose an option

Origem: Choose an option

Espera em segundos: 200 (0–400)

Selecione um intervalo de vôos: 2022-05-31 – 2023-05-31

Máximo de vôos simultâneos: 3 (1–10)

Flights: 79

	origem	destino	dt_dep	dt_arr	espera
3,221	SBRJ	SBSP	2023-02-20 19:09:51+00:00	2023-02-20 19:55:59+00:00	212.7663
3,222	SBCT	SBSP	2023-02-20 19:10:51+00:00	2023-02-20 19:58:31+00:00	211.0512
3,235	SBPA	SBSP	2023-02-20 19:26:41+00:00	2023-02-20 20:49:01+00:00	205.3898

Figure 8 – Airdelay web-based simulation tool, showing predicted flight delays due to holding maneuvers.

outperform GNNs in predictive tasks. Revealing that traditional graph machine learning continues important even in the area of graph deep learning, because of the possibility of integrating graph measures into tree-based models.

However, our work is limited due to the lack of time for more extensive hyperparameter tuning and model optimization. Future work should focus on refining the GAT model architecture, exploring additional graph-based features, and addressing the dataset's class imbalance more effectively with undersampling and/or oversampling techiniques. Additionally, other ML models, such as SVM, could be tested.

Furthermore, there are new GNN architectures that could be explored. A recent work by Egressy *et al.* (2024) introduced a provably powerful graph neural network for directed multigraphs, that could improve minority-class F1 score by up to 30%. In our scenario this could be a game changer, since the problem with the GAT model was the class unbalancing.

5 CONCLUSION

5.1 Project Contributions to the Student

This project was an excellent way to reinforce the knowledge I gained in my undergraduate program, particularly in areas like algorithms and graph applications. Working at ICMC was an invaluable opportunity to engage with experts in graph machine learning and complex networks. Through this project, I gained insight into both fields—a feat that would have been challenging without the foundation provided by courses like Complex Networks and Artificial Intelligence, which taught me the basics of each. Additionally, this experience provided an ideal introduction to studying *graph neural networks*, a field I plan to pursue further.

5.2 Relationship between the Undergraduate Course and the Project

The undergraduate curriculum strongly aligned with this project. Key subjects included Algorithms and Data Structures, Computational Modeling in Graphs, Artificial Intelligence, High-Performance Computing, Linear Algebra, Calculus, Stochastic Processes, Statistics, and Advanced Algorithms and Applications. Together, these courses provided a comprehensive foundation in computer science, with ample focus on graph theory, machine learning, and algorithm design. The curriculum thus offered a robust theoretical framework that enriched my understanding of complex networks and graph-based models.

Another crucial aspect of my program was the opportunity to participate in teaching and research. This experience not only deepened my technical knowledge but also honed my communication skills, enabling me to articulate ideas and solutions more effectively.

5.3 Reflections on the Undergraduate Course

The Bachelor of Computer Science at the Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, has been instrumental in providing me with knowledge, friendships, determination, and joy. I am deeply grateful for the supportive structure at ICMC, which offered everything necessary for personal and academic growth. Being surrounded by dedicated and inquisitive peers and professors has been the highlight of my time here, creating a stimulating environment that constantly pushed me to expand my horizons as a person, a scholar, and a future professional.

That said, I do have one critique regarding the course curriculum. Recently, it feels as though the Bachelor of Computer Science program has shifted its focus more toward the job market than foundational science. While I understand the need to prepare students for industry, I believe that we must preserve the course's scientific roots. For instance, 'Research

Methodology’ is not a required course, yet it is fundamental for academic development, while other subjects more geared toward job readiness are mandatory. The removal of ‘Calculus 4‘ from the curriculum further illustrates this shift away from theoretical depth, despite its relevance in areas like machine learning.

To address this, I would suggest offering two distinct tracks within the program: an Academic Track and a Professional Track. This structure would allow students interested in research and theoretical science to focus on these areas, while those aiming for immediate industry applications could focus on practical skills. This change might also encourage more students to pursue advanced degrees at ICMC by fostering a stronger scientific foundation during their undergraduate studies.

Despite these critiques, I am confident that choosing this program was one of the best decisions I have made. The course has opened doors to opportunities I had not even dreamed of achieving.

REFERENCES

AMANCIO, D. R. *et al.* A systematic comparison of supervised classifiers. **PLOS ONE**, Public Library of Science, v. 9, n. 4, p. 1–14, 04 2014. Available at: https://doi.org/10.1371/journal.pone.0094137.

BO, D. *et al.* **A Survey on Spectral Graph Neural Networks**. 2023. Available at: https://arxiv.org/abs/2302.05631.

BOCCALETTI, S. *et al.* Complex networks: Structure and dynamics. **Physics Reports**, v. 4, n. 424, p. 175–308, 2006.

BONACICH, P. Power and centrality: A family of measures. **American journal of sociology**, University of Chicago Press, v. 92, n. 5, p. 1170–1182, 1987.

BRIN, S. The pagerank citation ranking: bringing order to the web. **Proceedings of ASIS, 1998**, v. 98, p. 161–172, 1998.

BRÖHL, T.; LEHNERTZ, K. Centrality-based identification of important edges in complex networks. **Chaos: An Interdisciplinary Journal of Nonlinear Science**, AIP Publishing, v. 29, n. 3, 2019.

BRUNA, J. *et al.* Spectral networks and locally connected networks on graphs. **arXiv preprint arXiv:1312.6203**, 2013.

CHAMI, I. *et al.* Machine learning on graphs: A model and comprehensive taxonomy. **Journal of Machine Learning Research**, v. 23, n. 89, p. 1–64, 2022. Available at: http://jmlr.org/papers/v23/20-852.html.

CHEN, X. Understanding spectral graph neural network. 12 2020. Available at: https://arxiv.org/abs/2012.06660.

COSTA, L. d. F. Characterization of complex networks: A survey of measurements. **Advances in physics**, Taylor & Francis, v. 56, n. 1, p. 167–242, 2007.

DEFFERRARD, M.; BRESSON, X.; VANDERGHEYNST, P. Convolutional neural networks on graphs with fast localized spectral filtering. **Advances in neural information processing systems**, v. 29, 2016.

EGRESSY, B. *et al.* Provably powerful graph neural networks for directed multigraphs. *In*: **Proceedings of the AAAI Conference on Artificial Intelligence**. [*S.l.: s.n.*], 2024. v. 38, n. 10, p. 11838–11846.

FREEMAN, L. C.; BORGATTI, S. P.; WHITE, D. R. Centrality in valued graphs: A measure of betweenness based on network flow. **Social networks**, Elsevier, v. 13, n. 2, p. 141–154, 1991.

GROVER, A.; LESKOVEC, J. node2vec: Scalable feature learning for networks. *In*: **Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining**. [*S.l.: s.n.*], 2016. p. 855–864.

GUI, G. *et al.* Flight delay prediction based on aviation big data and machine learning. **IEEE Transactions on Vehicular Technology**, IEEE, v. 69, n. 1, p. 140–150, 2019.

HAMILTON, W.; YING, Z.; LESKOVEC, J. Inductive representation learning on large graphs. **Advances in neural information processing systems**, v. 30, 2017.

KIPF, T. N.; WELLING, M. Semi-supervised classification with graph convolutional networks. **arXiv preprint arXiv:1609.02907**, 2016.

LAMBELHO, M. *et al.* Assessing strategic flight schedules at an airport using machine learning-based flight delay and cancellation predictions. **Journal of air transport management**, Elsevier, v. 82, p. 101737, 2020.

LEE, D. h. *et al.* Development of real-time maneuver library generation technique for implementing tactical maneuvers of fixed-wing aircraft. **International Journal of Aerospace Engineering**, Wiley Online Library, v. 2020, n. 1, p. 7025374, 2020.

LU, L.; ZHANG, M. Edge betweenness centrality. *In*: _____. **Encyclopedia of Systems Biology**. New York, NY: Springer New York, 2013. p. 647–648. ISBN 978-1-4419-9863-7. Available at: https://doi.org/10.1007/978-1-4419-9863-7_874.

NEWMAN, M. **Networks**. [*S.l.: s.n.*]: Oxford university press, 2018.

NEWMAN, M.; GIRVAN, M. Finding and evaluating community structure in networks. **Physical review E**, APS, v. 69, n. 2, p. 026113, 2004.

PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk: Online learning of social representations. *In*: **Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining**. [*S.l.: s.n.*], 2014. p. 701–710.

PROKHORENKOVA, L. *et al.* Catboost: unbiased boosting with categorical features. **Advances in neural information processing systems**, v. 31, 2018.

RAHMANI, S. *et al.* Graph neural networks for intelligent transportation systems: A survey. **IEEE Transactions on Intelligent Transportation Systems**, v. 24, n. 8, p. 8846–8885, 2023.

SILVA, T.; ZHAO, L. **Machine Learning in Complex Networks**. Springer International Publishing, 2016. ISBN 9783319172897. Available at: https://books.google.com.br/books?id=WdDurQEACAAJ.

SMITH, A. P.; BATEMAN, H. Management of holding patterns: A potential ads-b application. *In*: IEEE. **2008 IEEE/AIAA 27th Digital Avionics Systems Conference**. [*S.l.: s.n.*], 2008. p. 3–D.

SOUZA, M. Caravanti de *et al.* Keywords attention for fake news detection using few positive labels. **Inf. Sci.**, Elsevier Science Inc., USA, v. 663, n. C, jun. 2024. ISSN 0020-0255. Available at: https://doi.org/10.1016/j.ins.2024.120300.

TUDISCO, F.; HIGHAM, D. J. Node and edge nonlinear eigenvector centrality for hypergraphs. **Communications Physics**, Nature Publishing Group UK London, v. 4, n. 1, p. 201, 2021.

VELIČKOVIĆ, P. *et al.* Graph Attention Networks. **International Conference on Learning Representations**, 2018. Available at: https://openreview.net/forum?id=rJXMpikCZ.

VERRI, F. A. N.; URIO, P. R.; ZHAO, L. Advantages of edge-centric collective dynamics in machine learning tasks. **Journal of Applied Nonlinear Dynamics**, L&H Scientific Publishing, v. 7, n. 3, p. 269–285, 2018.

VERRI, F. A. N.; ZHAO, L. High level data classification based on network entropy. *In*: **The 2013 International Joint Conference on Neural Networks (IJCNN)**. [*S.l.: s.n.*], 2013. p. 1–5.

WU, F. *et al.* Simplifying graph convolutional networks. *In*: PMLR. **International conference on machine learning**. [*S.l.: s.n.*], 2019. p. 6861–6871.

YOU, J. *et al.* Handling missing data with graph representation learning. **Advances in Neural Information Processing Systems**, v. 33, p. 19075–19087, 2020.

ZHAO, L. *et al.* T-gcn: A temporal graph convolutional network for traffic prediction. **IEEE Transactions on Intelligent Transportation Systems**, v. 21, n. 9, p. 3848–3858, 2020.

ZHAO, T.; ZHANG, X.; WANG, S. Graphsmote: Imbalanced node classification on graphs with graph neural networks. *In*: **Proceedings of the 14th ACM international conference on web search and data mining**. [*S.l.: s.n.*], 2021. p. 833–841.