Digital Technologies Research and Application
https://ojs.ukscip.com/index.php/dtra

Article

Dynamic GNNs for Predicting Train Cancellations on the Dutch Railway Network: A Multi-Season Study of Environmental and Operational Factors

Brent Brakenhoff , **Ali Mohammed Mansoor Alsahag** * **and Seyed Sahand Mohammadi Ziabari**

Informatics Institute, University of Amsterdam, 1090 GH Amsterdam, The Netherlands

* Correspondence: a.m.m.alsahag@uva.nl

Received: 16 October 2025; **Revised:** 4 November 2025; **Accepted:** 31 December 2025; **Published:** 30 January 2026

Abstract: Cancellations on the Dutch Railway network are a common and unpredictable occurrence; however, little research has focused on predicting these cancellations. Previous studies on the Dutch railway system have primarily concentrated on delay prediction. For this regression task, models such as XGBoost, Random Forest, Long Short-Term Memory (LSTM), and Gradient Boosting Decision Tree have been shown to perform well. Graph neural network-based models have been used for regression tasks on other transportation networks. We propose a Dynamic Graph Neural Network (DGNN) combined with an LSTM network for binary classification of cancelled trajectories. We compare the model with baseline models on a seasonal split to compare the feature importance across different seasons. Model performance is gauged using paired t-tests on bootstrapped F1 scores. Additionally, Precision, Recall, Balanced Accuracy, and AUC are considered metrics for further comparison. The newly proposed features achieve mostly positive feature importance scores across the models. Amongst the evaluated models, the proposed DGNN and XGBoost outperform the baseline models. Overall, the models underperform with F1 scores no higher than 0.4. This paper provides insight into the influence of various weather and operational features on cancellations on the Dutch railway network, with the operational features proving insightful.

Keywords: Railway Network; DGNN; Binary Classification; Time Series; Graph Neural Network; LSTM; XGBoost

1. Introduction

In 2024, nearly 6000 disruptions have been reported on the Dutch railway system [1]. Disruptions are defined as unplanned occurrences that temporarily hinder a train's movement from one station to the next, interrupting the train's normal trajectory. These disruptions cause one or multiple trains to be significantly delayed, or overall, less train traffic on the disrupted trajectory. In the context of a broader network, disruptions can be seen as a temporary form of edge removal, as the connection between two nodes is temporarily interrupted.

In 2024, Rijden de Treinen recorded 5964 disruptions (approximately 16 per day), providing a concrete indication of the scale of the phenomenon [2]. For context, 5321 disruptions were recorded in 2023 [3].

Train disruptions are most often caused by broken down trains, followed by infrastructure issues, accidents, external influences, and engineering work [1]. Due to the complexity and unpredictable nature of some of these causes, predicting the occurrence of disruptions based on topological features alone becomes challenging [4]. However, as the definition of disruption provided by Dutch Railways (NS) does not include all interruptions of the network, for most train cancellations, the causes are not recorded in the Rijdendetreinen.nl dataset. This research does include cancellations to include all occurrences of edge removal within the network.

Previous research shows edge removal is predictable on a monthly time scale, with promising results on the US airways [5]. A framework for a comparative study between models on the prediction of edge removal. The continuation of said research on the Dutch railway system has shown an overall decrease in performance of the same models [4]. Due to this decreased performance, we propose novel external factors in addition to the topological factors. The purpose of the additional features is to gain insight into the effect of external factors on cancellations and to potentially improve model performance. These external factors are divided into two categories: environmental factors and operational factors. Environmental factors include variables related to the temperature, wind speed, precipitation, and visibility at a given train station on a given day. The operational factors are related to the scheduling and previous cancellations on a given trajectory.

Disruptions on the Dutch railway system have been examined and predicted on a similar time frame once before [6]. This was achieved by creating a socio-technical representation of the Dutch railway network. Over a daily timespan, however, predicting disruptions remains largely unexplored for the Dutch railway system.

Each disruption or cancellation causes a delay in the network, which is transmitted across the connected stations in the network. This process is referred to as delay propagation. Predicting this property is essential for accurate train scheduling, and this has been done quite successfully over timespans of an hour in past research [7–10]. Alongside delay propagation, the ability to accurately predict future disruptions based on the operational data would allow backup schedules to be created in advance, decreasing delays across the network.

In real-world scenarios, predicting disruptions on a daily time scale would allow for timely rescheduling of affected trajectories. The prediction of cancellations on a daily time scale has not been researched before, the impact of environmental and operational factors on cancellations on the Dutch Railway Network has not been considered previously and Dynamic Graph Neural networks have not been applied for the purpose of predicting disruptions.

Recent work underscores two trends relevant to our setting. First, dynamic and spatio-temporal GNNs have matured rapidly, with surveys synthesizing architectures that integrate graph convolutions with temporal modules such as RNNs/LSTMs for evolving networks [11–13]. Second, railway-specific prediction has progressed beyond short-term regression: heterogeneous GNNs for delay evolution on rail networks [8] and fresh data-driven models for near-term arrival prediction [14] indicate growing transferable methodology. While the JoVE study [15] focuses on load dispatch in power systems, it exemplifies the broader uptake of neural methods for operational decision making, reinforcing our choice to combine graph-based encoders with temporal learners.

Due to the limited research in daily time scale disruption forecasting, and the effect of environmental and operational factors on disruptions, this project aims to answer the following question:
To what extent can implementing environmental and operational factors into a dynamic graph-based model be used to improve the prediction of disruptions in the Dutch railway system?

To answer this question, the following sub-questions would need to be answered:

- How do environmental factors (precipitation, temperature, etc.) influence disruptions in the Dutch railway system?
- What role do operational factors (derived from previous delays) play in exacerbating or mitigating disruptions?
- How well do various machine learning models incorporating topological, environmental, and operational features perform compared to the best performing model that uses topological features exclusively, measured by balanced accuracy, F1 score, and AUC metrics?
- How well does the proposed dynamic graph-based model perform compared to the baseline models using a combination of topological, environmental, and operational features, measured by balanced accuracy, F1 score, and AUC metrics?

By answering these questions, this project aims to highlight leading factors for disruptions by gauging the effect of environmental and operational factors on disruptions the Dutch railway network. Additionally, we conduct a comparative analysis on multiple machine learning models, which aims to provide an explainable model that predicts disruptions on a daily time scale. This paper makes four contributions: (i) a season-aware, leakage-safe evaluation protocol at daily granularity with matched-bootstrap inference; (ii) a portable set of operational features that consistently dominate attribution across seasons; (iii) a dynamic graph encoder (GAT) coupled with an LSTM temporal module (DGNN) benchmarked against strong tree baselines under severe class imbalance; and (iv) a scope-limited external check (U.S. air) showing that the relative method ordering transfers across domains and temporal resolutions.

2. Related Work

As mentioned previously, most research on the Dutch railway system concerns real-time forecasting or predictions on an at most 90-min time scale, for both disruptions and delay propagation. In the following section, we discuss some foundational works that have inspired this research, and further papers that support the baseline model selection and feature selection tools.

2.1. Foundational Works

Short-term delay prediction on the Dutch railway network has been achieved more successfully on multiple occasions [7–10]. This paper is mainly built upon the foundation of Lei et al.'s study [5], which proposes a model to predict missing edges in rapidly changing transportation networks, the airways of the United States of America, and bus networks in Brazil. Although the Dutch railway system would not be considered a fast-changing network due to the static nature of railways, temporary network changes, in the form of disruptions, still occur quite frequently. These network changes are usually reverted in a span of hours; using a shorter timespan than a month would be beneficial for forecasting on the Dutch railway system. This paper also provides a suitable pipeline for the comparative analysis of multiple models.

Additionally, this research aims to predict significantly delayed trajectories on a monthly time scale in the Dutch railway system. The proposed methodology, while promising, ended with average results. These less-than-desired scores seemed to stem from limited data availability, as the research focuses on trajectories on a monthly basis, and from the fact that the research only takes topological and node centrality-based features into account. The data limitation could be mitigated by grouping the data into single rides instead of trajectories or by viewing the trajectories on a smaller time scale, and the feature limitation can be mitigated by taking external factors into account.

In the short-term setting, Dekker et al. [6] propose a data-driven approach that infers operational states and delivers an early-warning metric for transition risk. The analysis covers June 2017–July 2018 and uses a two-principal-component representation without explicit temporal memory, prioritizing interpretability of the socio-technical system. Building on that perspective, we target daily disruption prediction by incorporating temporal dynamics and a broader feature set. Relative to Lei et al. [5] and Kämper [4], we adopt a daily label (trajectory–day cancellation) rather than monthly edge persistence, and attach operational covariates unavailable or unused in those setups. This shift tightens chronology and base-rate imbalance and lowers attainable precision–recall ceilings, but it reveals season- and operation-specific signal. We therefore position our DGNN as a temporal complement under daily granularity, and use matched-bootstrap tests to compare fairly against strong tree baselines.

2.2. Graph Neural Networks

RNNs are able to perform machine learning tasks on graph structures, which requires the data to be formatted in a way such that temporal relations between nodes are lost. Graph Neural Networks (GNNs) were specifically designed to perform machine learning tasks on graph-structured datasets [16,17]. Feng et al. [18] have created an overview of use cases, performance, and computational efficiency of over 80 different dynamic graph neural networks, trained on various datasets. This overview shows the versatility of dynamic graph neural networks and provides metrics for the evaluation of the models in different scenarios; however, the discussion of DGNNs in transportation settings in this overview is limited.

A Graph Attention Network (GAT) differs from a regular GNN by taking the state of neighbouring nodes into account while performing node classification [19]. Although GATConv was not originally designed for edge classification, its flexibility makes itsuitable for the task.

Graph-based models have been used to forecast event times in the Dutch railway system in the past. In the research by Kecman and Goverde [7], which proposed a dynamic graph-based model for predicting train event times in real time. The graph-based structure of the model allows for faster computations and allows for generalisability for the entire Dutch Railway network, as the proposed method is tested on a subsection of the network. This research does not provide the performance of other models as a baseline, and is out of date with the state-of-the-art, but it does function as a proof of concept.

In other transportation networks, graph-based models have been used to both forecast traffic flow and monitor vulnerabilities in the transportation system on a long-term basis [20,21]. In the research by Furno et al. [20], a dynamic graph-based model is proposed to forecast bottlenecks and vulnerabilities in the road network of Lyon, France. This paper proposes an algorithm that dynamically calculates the betweenness centrality of nodes in the network, indicating which nodes are most frequently traversed. The node centrality metric proves to be a suitable measurement for traffic flow, and the overall method proposed for constructing a dynamic graph will be useful for this project. The network used is structurally more complex than the Dutch railway system, and the model does not take external factors into account.

Peng et al. [21] propose a dynamic graph neural network to forecast long-term traffic flow, tested on city-bike data in New York. This paper shows the potential of using a dynamic graph representation of a transportation network as input data for a neural network, and of combining a GNN with an LSTM. Due to the differences in structure between the researched network and the Dutch railway network, the proposed model might lead to different results.

2.3. Baseline Models

The following models are considered as baselines to compare with the DGNN. Each has been used for regression or classification tasks on the Dutch railway network.

For short-term delay prediction on the Dutch railway system, an LSTM model was evaluated against an artificial neural network (ANN) and a Random Forest (RF), with the LSTM reporting roughly 20% higher accuracy than the ANN while RF performed comparably to LSTM [8]. A 20-minute horizon study identified RF as the best performer; a daily weather summary feature was examined but found uninformative at finer temporal scales [10]. In that setting, XGBoost and GBDT trailed RF only slightly on MAE/RMSE, with XGBoost outperforming RF on precision for delays $\leq$3 min, and ANN underperforming all three with notably longer training and loading times [10]. On the UK railway network, a GBDT approach was proposed with RF, SVR, and a multilayer perceptron (MLP) as comparisons [9].

For edge-level classification on graph structures, prior work used XGBoost for predicting edge removal [4,5]. GBDT and RF achieved results comparable to XGBoost in one study, while logistic regression unexpectedly led on topology-only features for the Dutch network [4]; another study found XGBoost to be the top baseline and did not assess neural networks [5].

Tree-based models (XGBoost, GBDT, RF) are effective at capturing nonlinear feature interactions, which is important here. LSTMs are standard for time-series forecasting and can learn sequential patterns relevant to recurring disruption regimes. Collectively, these methods form a solid baseline suite for assessing the proposed dynamic graph-based model, noting that several have more often been applied to regression than to classification in related work.

An overview of proposed models and baselines used in related work is provided in **Table 1**.

Table 1. Proposed and baseline models in related works.

Paper	Dataset(s)	Task	Proposed Model	Baselines	Time
Kecman and Goverde (2015) [7]	NL rail	Event-time pred.	DFS over graph	–	RT
Furno et al. (2021) [20]	French highways	BC computation	Dynamic-graph-based modelling	Various BC algorithms	RT
Peng et al. (2021) [21]	NYC bikes	Traffic flow	DGNN (GCN+LSTM)	ARIMA, SVR, LSTM, various GNNs	D
Li et al. (2020) [8]	NL rail	Regression	LSTM	RF, ANN	RT
Wen et al. (2020) [10]	NL rail	Regression	RF	XGBoost, GBDT, ANN, statistical models	RT
Zhang et al. (2021) [9]	UK rail	Regression	GBDT	RF, SVR, MLP	RT
Lei et al. (2022) [5]	US air; BR bus	Edge removal	XGBoost	RF, LR, GBDT, ridge classifier, etc.	M
Kämper (2025) [4]	NL rail; US air	Edge classification	XGBoost	LR, RF, GBDT	M
This paper	NL rail	Edge classification	DGNN (GATConv+LSTM)	LR, GBDT,XGBoost, LSTM	D

Note: Time scale abbreviations—RT: real-time/near-term; D: daily; M: monthly.

2.4. Feature Selection

In order to represent and visualise the feature importance of the models, the SHAP values of these features are calculated [22]. These are values assigned to features based on their influence on a model during a particular prediction task. These values add important insight into the predictive tendencies of the models, as not every model's predictions are equally intuitive, and thus increase the explainability of the models. Based on these SHAP values, feature selection is also possible to reduce noise in the dataset and further fine-tune the overall models on specific datasets. A recently proposed lightweight method for this task is the 'shap_select' method [22]. This method fits a linear or logistic regression model on the original Shapley values of a fitted model. This method requires one fitted model on the entire feature set, making it inherently less computationally intensive than comparable methods. Besides taking the absolute mean SHAP values into account, it also highlights their statistical importance by fitting either a linear or logistic regression method with the SHAP values of the original model. This method has two major drawbacks for this research; however, the first is that it is not supported for neural networks, so it can not be used for feature selection of the proposed LSTM model. The second drawback of the method is that, in order to use it on a Logistic Regression model, the SHAP values of that model will be used to fit another Logistic Regression model, which is more computationally expensive than looking at the original model's coefficients directly. As one of the models for which 'shap_select' is not supported is the LSTM, for which the permutation importance methods will be used [23]. This approach is especially suitable for the LSTM model as it does not require any insight into the model's internal architecture. This method was originally designed for Random Forest. Due to its model-agnostic nature, it is commonly used to calculate feature importance for LSTM models and other Neural Networks. However, because of its computational complexity, it is less suitable than 'shap_select' for most models in this research.

Overall this paper aims to combine the approaches of delay forecasting on the Dutch railway network and edge removal forecasting on other transportation networks. In further sections, we provide a comparative analysis of models with promising results in past work, and propose a novel Dynamic Graph Neural Network. We use feature selection methods that are suitable for the analysed models, with the goal of providing insight in the decision making process. Finally we evaluate the models on the selected feature subset with the goal of improving performance. All feature selection is performed on the validation year only and frozen prior to testing to prevent leakage.

3. Materials and Methods

The following section outlines the methodology for the discussed comparative study between the baseline models and the proposed DGNN. This section starts by discussing the model selection, after which the data collection and preparation are outlined, followed by the validation and the evaluation. The final section concerns a brief external validation method. A comprehensive overview of the complete pipeline is visible in **Figure 1**.

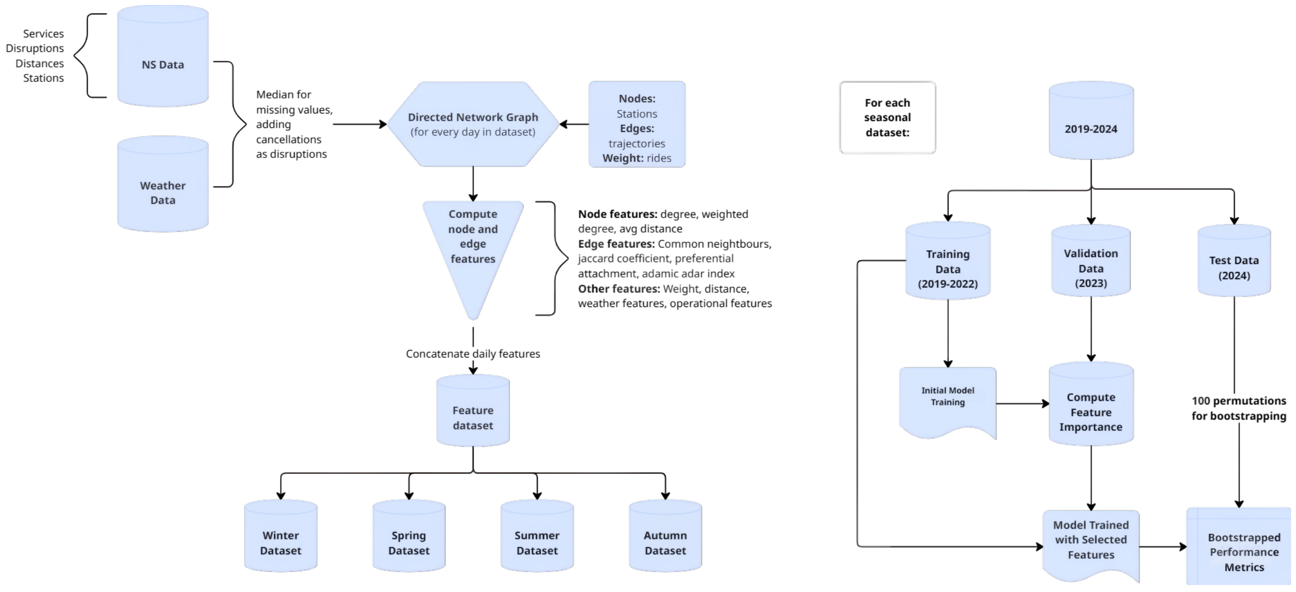

Services
Disruptions
Distances
Stations
NS Data
Weather Data
Median for missing values, adding cancellations as disruptions
Directed Network Graph (for every day in dataset)
Nodes: Stations
Edges: trajectories
Weight: rides
Compute node and edge features
Node features: degree, weighted degree, avg distance
Edge features: Common neightbours, jaccard coefficient, preferential attachment, adamic adar index
Other features: Weight, distance, weather features, operational features
Concatenate daily features
Feature dataset
Winter Dataset
Spring Dataset
Summer Dataset
Autumn Dataset
For each seasonal dataset:
2019-2024
Training Data (2019-2022)
Validation Data (2023)
Test Data (2024)
Initial Model Training
Compute Feature Importance
100 permutations for bootstrapping
Model Trained with Selected Features
Bootstrapped Performance Metrics

Figure 1. Overview of Proposed Methodology.

3.1. Model Selection

The following models from related literature are selected for this task, these models being: Logistic Regression, Gradient Boosting Decision Tree (GBDT), Random Forest (RF), XGBoost, LSTM (Long Short-Term Memory), and a Graph Neural Network.

Logistic Regression is selected based on its interpretability, simplicity, and effectiveness in binary classification tasks, mainly with limited data. This method is used to benchmark the other methods mostly, as it is unlikely to capture the complex dynamics between variables well enough to perform well on this task.

3.1.1. Tree Based Models

Random Forest and Gradient Boosting have shown to outperform simpler approaches in delay prediction on a railway track, as well as during the prediction of significantly delayed trajectories [4,10,24]. The main difference between these two models is that Random Forest builds trees independently and GBDT builds trees sequentially, which causes GBDT to be more sensitive to patterns in the data. However, this approach could lead to over-fitting, hence why the more intuitive and less tunable Random Forest approach is not disregarded.

Out of the non neural network approaches, XGBoost is the closest to the state of the art. The method is an optimised version of the GBDT classifier, therefore has the same benefits as the GBDT. The method is considered as, like the other two tree based methods, the model is capable of capturing feature interactions. Due to XGBoost's similarity to GBDT, it is able to interpret patterns in the dataset and it has the ability to handle data imbalances. XGBoost has performed quite well during the research on the Dutch railway system and other transportation networks [5,8].

3.1.2. LSTM

An LSTM is chosen over a Convolutional Neural Network (CNN) or Recurrent Neural Network (RNNs), for the architecture's ability to store and learn from past information [25]. This enables the ability to interpret patterns across the dataset, and therefore, if the same patterns of cancellations occur over a longer period of time, an LSTM should be able to consider that pattern during future predictions. LSTMs are also robust against irregularities and outliers in the dataset and are able to interpret non-linear patterns between variables. An LSTM model has outperformed ANNs and Random Forest in predicting train delays on single trajectories within the Dutch railway network and has performed well on predicting traffic flow of rental bikes in New York City [10,21].

3.1.3. DGNN

As the Dutch railway network can be represented as a graph, with intricate sets of variables for each node and edge that vary daily, a Graph Neural Network (GNN) is selected, specifically using a Graph Attention Network (GATConv) architecture. GATConv layers use an attention mechanism, shown in Equation (1), to weight the importance of neighbouring nodes when aggregating their features. This is particularly useful for edge classification, as the features of the nodes connected by an edge can be combined in a context-aware manner to predict edge properties.

$$\begin{aligned} h_i' &= \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}\, W h_j\right), \\ \alpha_{ij} &= \frac{\exp\big(\text{LeakyReLU}\big(a^\top [\, W h_i \,\|\, W h_j \,]\big)\big)}{\sum_{k\in\mathcal{N}(i)} \exp\big(\text{LeakyReLU}\big(a^\top [\, W h_i \,\|\, W h_k \,]\big)\big)}, \\ \mathcal{L}(\theta) &= -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i) \end{aligned} \tag{1}$$

where i and j index nodes; $\mathcal{N}(i)$ is the neighborhood of i; $h_i \in \mathbb{R}^F$ is the input node feature and $h_i' \in \mathbb{R}^{F'}$ the updated feature; $W \in \mathbb{R}^{F' \times F}$ is a learnable weight matrix; $a \in \mathbb{R}^{2F'}$ is the attention vector; α_{ij} are the normalized attention coefficients with $\sum_{k\in\mathcal{N}(i)} \alpha_{ik} = 1$; $\sigma(\cdot)$ is a pointwise nonlinearity; $[\,\cdot\|\cdot\,]$ denotes concatenation; and LeakyReLU$(\cdot)$ is the leaky rectified-linear activation.

The DGNN is constructed in the following manner. As these are updated daily, the network graph is dynamic. The model architecture consists of a spatial layer, a temporal layer and an output layer [18]. The graph convolution layer captures dependencies between stations, the temporal layer models the daily evolution of the graph, and finally and the output layer. For this research, GATConv layers are chosen to form the spatial layer, due to their discussed ability to weigh the importance of neighbouring node features. An LSTM is chosen for the temporal layer, following similar reasoning as during its selection as a baseline model [10, 21]. The output layer consists of a sigmoid function that returns the probability of a disruption for each edge. The model is trained using a class-weighted binary cross-entropy loss function, 'BCEWithLogitsLoss', with weights that are the inverse of the class frequency.

We minimize the class-weighted logistic loss

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\Big[w_1\, y_i \log \sigma(z_i) + w_0\,(1-y_i)\log\big(1-\sigma(z_i)\big)\Big] \tag{2}$$

with $z_i = f_\theta(x_i)$, σ the sigmoid, and $w_1 = \frac{N}{2N_1}$, $w_0 = \frac{N}{2N_0}$ computed on the training fold. The decision threshold is tuned on the validation year to maximize macro-F1 and then fixed for the test year.

This function handles the imbalanced nature of the dataset, as disruptions are rare events, without altering the dataset itself.

3.2. Data Collection

This project uses four datasets from Rijden de Treinen: the NS Services datasets from 2019 until 2024, the NS Disruption datasets from 2019 until 2024, the Station Distances dataset, and the 'Stations-2023-09-nl' dataset [1]. The Services dataset is filtered for services provided by NS, concatenated, and combined into daily trajectories. This dataset contains data on delays and cancellations for each NS trajectory for every day.

The NS Disruptions dataset is filtered by lines, statistical cause, start, and end time. The lines are split into individually affected lines, as some disruptions affect multiple lines. Using the start and end time, the number of disruptions per line per day is added, and the statistical causes are stored in a list. The affected lines are split into the start and target of the trajectory, in order to merge the dataset with the daily trajectory dataset. After merging these datasets, it became apparent that some trajectories have either partial or complete cancellations, but no disruption was reported in the disruptions dataset. These cancellations are added manually to include all cases of temporary edge deletion in the system. By adding these cancellations, we expand the definition of disruption in this project to include all cases of network interruption.

The binary target is defined at the trajectory–day level: $y_{u \to v,t} = 1$ ("disrupted") if the NS Services data indicate a partial or complete cancellation on (u, v) on day t, and $y_{u \to v,t} = 0$ otherwise. Because some cancellations are not listed in the Disruptions feed, we retain $y_{u \to v,t} = 1$ in such cases to reflect temporary edge removal. Before merging sources, we removed duplicates and normalized station names; we then spot-checked a random sample of 200 trajectory–days for Services/Disruptions consistency. The resulting positive prevalence aligns with the EDA (≈14%).

Weather Dataset

The weather data is sourced from the Royal Netherlands Meteorological Institute. The data is sourced from a selection of 9 weather stations to provide a complete overview of the weather in the Netherlands. A set of 10 features that are suspected to affect the operation schedule of trains the most is selected. Features related to daily temperature, the daily average wind speed in 0.1 m/s (FHVEC), precipitation, and the maximum visibility. The measurements related to temperature are: 'TG', the daily mean temperature in ($0.1\ ^\circ$C); 'TN', the minimum temperature (in $0.1\ ^\circ$C); 'TX', maximum temperature (in $0.1\ ^\circ$C); and 'T10N', minimum temperature at 10 cm above the surface (in $0.1\ ^\circ$C). The features related to precipitation are: 'SQ', the sunshine duration (in 0.1 h) calculated from global radiation (-1 for <0.05 h); 'DR', the precipitation duration (in 0.1 h), 'RH', the daily precipitation amount (in 0.1 mm) (-1 for <0.05 mm); and 'RHX', the maximum hourly precipitation amount (in 0.1 mm) (-1 for <0.05 mm) [26].

Using the Haversine formula between the coordinates of the train stations and the weather stations to find the closest weather station for each train station, the weather data is added to the dataset based on the start stations of the trajectories. The missing measurements in the weather dataset are replaced by the median of that measurement.

3.3. Data Preparation

For each day in the dataset, a directed network graph is created using the stations as nodes, the trajectories as edges, the rides planned as weight and the distance between stations as distance. For both the start and target of the trajectory, the node degree, weighted node degree, and average distance (between connections) are calculated. For the trajectories, the Common neighbours, Jaccard coefficient, preferential attachment, and adamic adar index are calculated, alongside the weight, distance, and weather features. The definitions of the topological features can be found in **Table 2**.

Table 2. Topological Feature Definitions and Formulas.

Feature	Type	Formula/Description
Node Degree	Node-level	$deg(v) = \|\{u \in V : (v, u) \in E\}\|$
Weighted Degree	Node-level	$wdeg(v) = \sum_{u \in N(v)} w(v, u)$, where $w(v, u)$ is the edge weight
Average Distance	Node-level	$\frac{1}{\|N(v)\|} \sum_{u \in N(v)} d(v, u)$
Common Neighbours	Trajectory-level	$\|\Gamma(u) \cap \Gamma(v)\|$
Jaccard Coefficient	Trajectory-level	$\frac{\|\Gamma(u) \cap \Gamma(v)\|}{\|\Gamma(u) \cup \Gamma(v)\|}$
Preferential Attachment	Trajectory-level	$\|\Gamma(u)\| \cdot \|\Gamma(v)\|$
Adamic-Adar Index	Trajectory-level	$\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log \|\Gamma(w)\|}$

We use $G_t = (V_t, E_t)$ for the daily railway graph; nodes V_t are stations; directed edges E_t are trajectories. For $v \in V_t$, deg(v) and $wdeg(v)$ denote (out-)degree and weighted degree; $N(v)$ is the neighbor set; $d(u, v)$ is the geodesic station distance; and $w(u, v)$ is the number of planned rides on (u, v) that day. Unless stated otherwise, weather and operational covariates are attached as edge features.

In addition to the topological features and weather features, operational features based on previously planned trajectories are added to the dataset. These features are the number of days since the last disruption, the number of disruptions previously to that moment, and the amount of past disruptions divided by the number of rides planned on this trajectory in the past. These daily features are stored in a dataset that will be used to train the discussed models.

EDA

The final dataset thus consists of 26 columns and 612,790 rows. The columns consist of the source and destination of the trajectory, the date, the discussed topological features and weather features, and a binary column noting whether that trajectory has been disrupted during that day. Of all daily trajectories in the dataset, only 13.2% have been disrupted or canceled.

Of all planned trajectories, 6.14% have never been disrupted or canceled; these trajectories are deleted as the mean and median of planned rides amongst these trajectories were much lower than the mean and median of the rest of the dataset. This means that either these trajectories do not have a lot of traffic, or disruptions and cancellations have not been recorded in the dataset. We made the decision to drop these outliers from the dataset. This decision increased the percentage of disrupted trajectories in the dataset to 14%.

3.4. Validation

3.4.1. Data Split

To approximate model performance in the initial stages, we used a time-series split to divide the data into 6 partitions of equal size, maintaining chronological order. These partitions are, in turn, split into a train and test set using an 80/20 split for training and validation. The aggregated metrics are calculated by taking the mean of each metric for each model and adding or subtracting the standard deviation. This process is repeated for every subset of features, including topology features only, topology and weather features, topology and operational features, etc. Initially, SMOTE was considered as an option to accommodate data imbalance; however, as SMOTE is not aware of the temporal and topological structures of data, which are crucial in the dataset, it was deemed unsuitable for this project [27]. Balancing class weights for the models is a more suitable option. Nearest neighbours in feature space need not respect graph connectivity or temporal order; oversampling risks topology distortion and time leakage. We therefore rely on class-weighting and threshold tuning.

To support transparency and reproducibility, all temporal splits preserve chronology (train: 2019–2022, validate: 2023, test: 2024 within each season). Randomized procedures (bootstrapping, model initialization, random search) use fixed seeds (42) with deterministic backends where available. Feature selection is performed on the validation year only and then frozen before testing to avoid leakage. Class weights and decision thresholds are tuned on validation and kept fixed for the test year. For all reported metrics, we draw 100 stratified bootstrap resamples from the held-out test year within each season, preserving per-day edge counts to maintain temporal and graph structure. Confidence intervals are percentile-based; random seeds are fixed (42).

For the final results, the data is split into 4 different datasets by season based on the date of each row. The models are validated and evaluated on these seasons separately to further highlight the impact of different features on the predictions during different times of the year. Allowing us to create different models specifically designed for different seasons. Originally, a monthly split was considered, but the overall distribution of cancellations did not seem to differ enough every month, and this would have led to smaller training sets, which would have possibly led to ambiguity due to the amount of differently configured models, each could have its own unique set of hyperparameters and selected features. The seasonal datasets are then split by year for cross-validation, which is straightforward for every season apart from winter, as winter spans two calendar years. Thus, the decision was made to span the winter season of 2020, which spans from December 21st, 2019, to March 20th, 2020. However, this leads to a small issue for the years 2019 and 2024, as we do not have any data from the year 2018. The winter of 2019 starts on January first and is thus 10 days shorter than it would be regularly. As the disruption data for 2025 is yet unavailable, the winter of 2024 currently only spans from December 21st to December 31st. All other seasons behave regularly. As visible in **Figure 1**, for each season, the years from 2019 until 2022 are used as training data, the data in 2023 is used as validation data, and to perform feature analysis and selection. The final year of the dataset is used as testing data exclusively.

3.4.2. Model Configuration

The hyperparameters of the models are optimised using random search, as in the study by Kämper [4]. However, Bayesian optimisation is used to tune the LSTM hyperparameters, as it is more computationally efficient [28]. The selection of hyperparameters and their values are visible in **Table 3**. For Logistic Regression and the tree-based models, Random Forest, Gradient Boosting, and XGBoost, we use random search to tune a model for each season. Using the seasonally split dataset, splitting the data for the season by year, and then cross-validation for 5 splits. The maximum iterations is set to 10, to decrease computational cost. This process is repeated for each of the four seasons.

Table 3. Hyperparameter grids for each model.

Model	**Parameter Grid**
Logistic Regression	C: {0.001, 0.01, 0.1, 1, 10, 100}, penalty: {l1, l2, elasticnet}, solver: {liblinear, saga, newton-cg}
Random Forest	n_estimators: {100, 200, 500, 1000}, max_features: {auto, sqrt, log2}, max_depth: {10, 20, 30, None}, min_samples_split: {2, 5, 10}, min_samples_leaf: {1, 2, 4}, bootstrap: {True, False}
GBDT	n_estimators: {100, 200, 300}, learning_rate: {0.01, 0.1, 0.3}, max_depth: {3, 5, 7}, subsample: {0.6, 0.8, 1.0}, min_samples_split: {2, 5, 10}
XGBoost	max_depth: {3, 5, 7}, learning_rate: {0.01, 0.1, 0.3}, n_estimators: {100, 200, 300}, subsample: {0.6, 0.8, 1.0}, colsample_bytree: {0.6, 0.8, 1.0}, scale_pos_weight: {1, 6.1335921414928185}
LSTM	'dense': {16, 128}, 'learning_rate': {1×10^{-4}, 1×10^{-2} }, 'lstm1': {32, 256}, 'lstm2': {16, 128}

3.4.3. Logistic Regression

Logistic regression has the fewest number of hyperparameters out of the selected models. The ones that are tuned are 'C', 'penalty', and 'solver'. Where 'C' represents the strength of regularisation, 'penalty' represents the function used to add a penalty to the model when too many variables are used, and finally, the solver represents the algorithm used in the optimisation problem.

3.4.4. Tree Based Models

For the Random Forest classifier, tuning focuses on the number of trees (n_estimators), how many features to consider when looking for the best split (max_features), and tree-specific controls such as maximum depth, the minimum number of samples needed to split a node, and the minimum required at a leaf. Bootstrapping is also toggled as part of the optimization.

For Gradient Boosted Decision Trees (GBDT), the key parameters include the number of boosting rounds, learning rate, maximum depth of trees, the subsample ratio of the training data, and the minimum samples needed to split an internal node. For XGBoost, similar parameters are tuned: tree depth, learning rate, and number of estimators, along with subsampling and feature sampling ratios. Additionally, scale_pos_weight is adjusted to account for class imbalance, serving a role similar to class weighting.

3.4.5. LSTM

The LSTM architecture consists of two Keras LSTM layers, each with a variable size, a dropout layer of 0.3 to prevent over-fitting, followed by a densely connected neural network layer which is also of variable size, and finally a densely connected output layer with a Softmax activation function. The LSTM model architecture used in this work is shown in **Figure 2**. The variable sizes of the layers are tuned using Bayesian optimisation, alongside the learning rate of the network. These parameters are visible in **Table 3**, and the final results for each season are visible in **Table 4**. Early stopping is implemented to prevent over-fitting after 5 epochs.

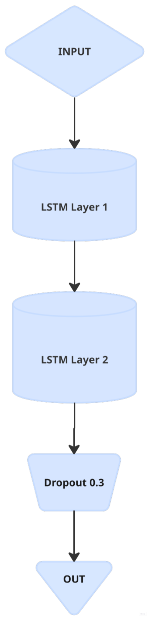


Figure 2. LSTM Model Architecture.

Table 4. Results after hyperparameter tuning for each season.

Model	Best Parameters Winter	Spring	Summer	Autumn
Logistic Regression	‘solver‘: ‘newton-cg‘, ‘penalty‘: ‘l2‘, ‘C‘: 0.001	‘solver‘: ‘saga‘, ‘penalty‘: ‘l1‘, ‘C‘: 0.1	‘solver‘: ‘liblinear‘, ‘penalty‘: ‘l2‘, ‘C‘: 100	‘solver‘: ‘liblinear‘, ‘penalty‘: ‘l2‘, ‘C‘: 100
Random Forest	‘n_estimators‘: 200, ‘min_samples_split‘: 5, ‘min_samples_leaf‘: 2, ‘max_features‘: ‘sqrt‘, ‘max_depth‘: 10, ‘bootstrap‘: False	‘n_estimators‘: 200, ‘min_samples_split‘: 5, ‘min_samples_leaf‘: 2, ‘max_features‘: ‘sqrt‘, ‘max_depth‘: 10, ‘bootstrap‘: False	‘n_estimators‘: 200, ‘min_samples_split‘: 5, ‘min_samples_leaf‘: 2, ‘max_features‘: ‘sqrt‘, ‘max_depth‘: 10, ‘bootstrap‘: False	‘n_estimators‘: 200, ‘min_samples_split‘: 5, ‘min_samples_leaf‘: 2, ‘max_features‘: ‘sqrt‘, ‘max_depth‘: 10, ‘bootstrap‘: False
GBDT	‘subsample‘: 0.8, ‘n_estimators‘: 300, ‘min_samples_split‘: 5, ‘max_depth‘: 7, ‘learning_rate‘: 0.3	‘subsample‘: 0.6, ‘n_estimators‘: 300, ‘min_samples_split‘: 5, ‘max_depth‘: 7, ‘learning_rate‘: 0.1	‘subsample‘: 1.0, ‘n_estimators‘: 300, ‘min_samples_split‘: 10, ‘max_depth‘: 5, ‘learning_rate‘: 0.3	‘subsample‘: 1.0, ‘n_estimators‘: 300, ‘min_samples_split‘: 10, ‘max_depth‘: 5, ‘learning_rate‘: 0.3
XGBoost	‘subsample‘: 0.6, ‘scale_pos_weight‘: 6.134, ‘n_estimators‘: 200, ‘max_depth‘: 3, ‘learning_rate‘: 0.1, ‘colsample_bytree‘: 0.8	‘subsample‘: 0.6, ‘scale_pos_weight‘: 6.134, ‘n_estimators‘: 200, ‘max_depth‘: 3, ‘learning_rate‘: 0.1, ‘colsample_bytree‘: 0.8	‘subsample‘: 0.8, ‘scale_pos_weight‘: 6.134, ‘n_estimators‘: 100, ‘max_depth‘: 7, ‘learning_rate‘: 0.01, ‘colsample_bytree‘: 1.0	‘subsample‘: 0.8, ‘scale_pos_weight‘: 6.134, ‘n_estimators‘: 100, ‘max_depth‘: 7, ‘learning_rate‘: 0.01, ‘colsample_bytree‘: 1.0
LSTM	‘dense‘: 53.627, ‘learning_rate‘: 0.0025, ‘lstm1‘: 130.545, ‘lstm2‘: 50.906	‘dense‘: 50.122, ‘learning_rate‘: 0.00070, ‘lstm1‘: 127.434, ‘lstm2‘: 49.681	‘dense‘: 68.809, ‘learning_rate‘: 0.0017, ‘lstm1‘: 211.639, ‘lstm2‘: 26.808	‘dense‘: 99.171, ‘learning_rate‘: 0.0053, ‘lstm1‘: 99.276, ‘lstm2‘: 18.204

3.4.6. DGNN

A schematic of the proposed DGNN (spatial GAT layers + temporal LSTM) is shown in **Figure 3**. As GNNs take graph structures as input, the final feature dataset was converted back into daily graphs, using the stations as nodes, trajectories as edges, but with added lists of node and edge features, the node features are the degree, weighted degree and average distance, the rest of the features are all set as edge features, as the model mostly predicts based on these edge features. The node and edge features are normalised before the first layer of the model. The DGNN takes these daily graphs of the network as an input, which are first fed into a GATConv layer with 4 heads and a hidden dimension of 32, the output of which is normalised using batch normalisation, after which ReLU is applied. This is repeated once more with a second GATConv layer, with one head and a hidden dimension of 32. The input size for this layer is four times the hidden layer due to the size of the output of the first layer. After this layer, a dropout of 0.35 is applied to reduce over-fitting. Then, the trajectory edge attributes of the result are fed into an LSTM, the output of which is converted to categorical and returned. We do not study the analytic existence/uniqueness of minimizers for this non-convex objective. Our focus is well-posed prediction, supported by chronological validation, regularization (dropout, batch normalization), and bootstrap-based robustness checks.

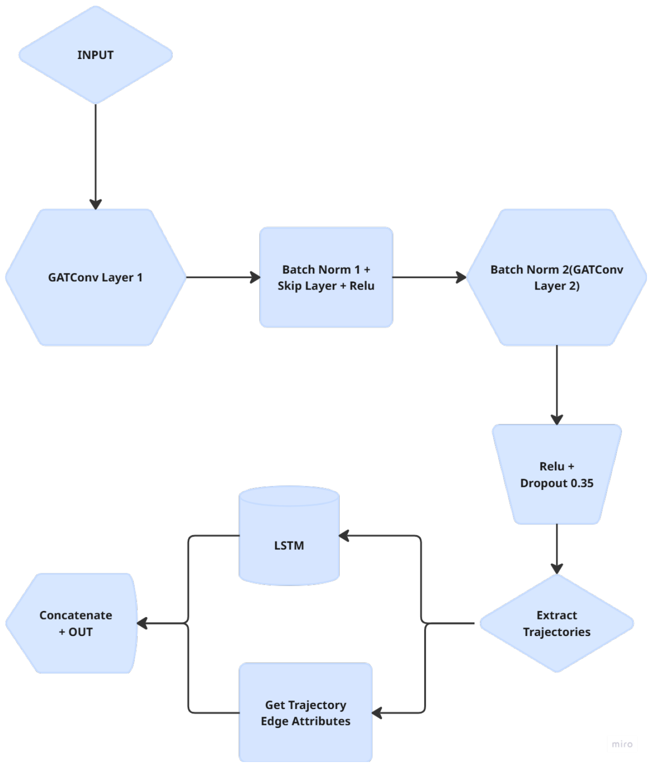

INPUT
GATConv Layer 1
Batch Norm 1 + Skip Layer + Relu
Batch Norm 2(GATConv Layer 2)
Relu + Dropout 0.35
Extract Trajectories
LSTM
Get Trajectory Edge Attributes
Concatenate + OUT
miro

Figure 3. DGNN Model Architecture.

3.5. Evaluation

The first two sub questions are answered by calculating the SHAP values of the newly proposed features, as these values quantify their importance and overall influence on the models [5]. After model validation, the best performing model of each type for each season are used to calculate the SHAP values on the data its respective season in 2024. Using these values, for the supported models, the aforementioned 'shap-select' method is used [22]. As this method malfunctions for Random Forest, and does not support Logistic Regression and LSTM, the feature importance for Logistic Regression and Logistic Regression are based on the absolute mean SHAP values. The top 10 highest scoring values with an absolute value over 0.01 are selected. This method ensures the models are trained on enough features to make an accurate prediction, while minimising the amount of noise added to the models by training on insignificant features. The feature importance for the LSTM model is calculated using the permutation method [23]. The top 10 features with a positive permutation importance score are selected for each season, as

LSTM models tend to under-fit easily on lower-dimensional data [25].

The models are retrained using their respective subset of selected features, using the configuration of hyperparameters selected during validation. These newly trained models are compared using a paired t-test on the macro F1 score, which is calculated over folds of the test set. This results in a statistically valid comparison of the models. In addition to comparing the models with each other, their accuracy and recall on disrupted cases, their balanced accuracies and AUC score are evaluated. Accuracy and recall are important during this research especially, as ideally the model misses as few disrupted cases as possible. Falsely predicting a disruption is less critical, however with too many of such cases a model is still not suitable. Therefore the F1 score is selected for the model comparison, as it provides a compromise between precision and recall, as ideally most predicted disruptions are accurate and most of the occurring disruptions are predicted. The F1 scores of the models are compared using a paired t-test over matched bootstrap resamples, which controls for fold-to-fold variability.

Furthermore, balanced accuracy has been selected to mitigate the imbalanced nature of the dataset, as standard overall accuracy of the model would be biased towards the majority class of no disruption. The AUC has been selected as a tool to assess the models true positive rate and false positive rate, and has proven useful in binary classification problems [18]. Additionally, utilizing these metrics enables clear comparison to previous work, as these have been used to evaluate the models [4,5].

3.6. External Validation

As an argument for generalisability of the methodology, the validation is performed on the US Airways data provided by Lei et al. [5]. Due to computational limitations however, this external validation is only performed on a subset of the network, containing the top 10% most popular airports during the summer season. The dataset consists of a monthly structure instead of a daily structure, so monthly graphs are created instead of daily ones. As the dataset spans from 2004 to 2021, the filtered dataset consists of 1,270,480 rows. As a target variable, we take the removal of edges in the dataset [5]. Approximately 1.9% of the rows in the dataset have been removed, which makes the dataset more imbalanced than the NS dataset. The rest of the external validation pipeline follows the methodology for the NS dataset as closely as possible. However only the scores for the best performing baseline model and the DGNN are reported, for brevity.

This external check is intentionally scope-limited (top 10% busiest airports; summer; monthly aggregation) to ensure a dense graph and computational tractability under the original labeling protocol. Within this conservative setup, the relative ordering of methods matches the Dutch results, indicating that the seasonal evaluation protocol and modelling strategy transfer without bespoke tuning.

4. Results

4.1. Initial Results on Yearly Split

The results in **Table 5** are inconclusive as these were gathered prior to feature selection and the first yearly data split was used instead of the later seasonal split. However, the following hypotheses can be formed:

- The models tend to perform better on a combination of topological, weather and operational features than on the topological features alone.
- The weather features are not substantially important to model performance, as the subset containing only the weather feature seems to have the lowest performance scores overall.
- This means the best performing feature subsets for each model are likely to consist of only topological and operational features.

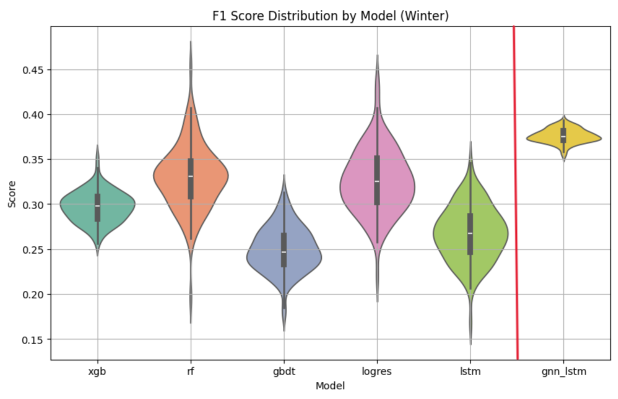
Table 5. Cross-Validation Performance Metrics (Mean $\pm$ Standard Deviation).

Model	Balanced Accuracy	F1 Score	AUC
Topology features	-	-	-
Logistic Regression	0.676 ± 0.028	0.382 ± 0.037	0.739 ± 0.038
Random Forest Classifier	0.585 ± 0.021	0.286 ± 0.037	0.752 ± 0.035
XGB Classifier	0.703 ± 0.035	0.403 ± 0.029	0.772 ± 0.043
Gradient Boosting Classifier	0.577 ± 0.017	0.266 ± 0.042	0.781 ± 0.041
LSTM	0.554 ± 0.008	0.201 ± 0.025	0.783 ± 0.018
Mean of the above scores ([sum(scores) $\pm$ sum(std)])/n	[0.597, 0.641]	[0.266, 0.334]	[0.730, 0.800]

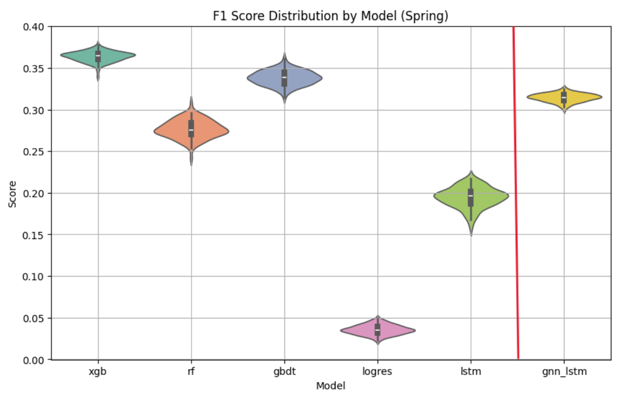
Table 5. *Cont.*

Model	Balanced Accuracy	F1 Score	AUC
Weather features	-	-	-
Logistic Regression	0.511 ± 0.007	0.222 ± 0.033	0.512 ± 0.014
Random Forest Classifier	0.501 ± 0.004	0.059 ± 0.017	0.499 ± 0.007
XGB Classifier	0.497 ± 0.007	0.177 ± 0.018	0.495 ± 0.012
Gradient Boosting Classifier	0.500 ± 0.000	0.005 ± 0.003	0.498 ± 0.008
LSTM	0.501 ± 0.009	0.186 ± 0.027	0.499 ± 0.015
Mean of the above scores	[0.497, 0.507]	[0.110, 0.149]	[0.489, 0.512]
Operational features	-	-	-
Logistic Regression	0.674 ± 0.012	0.345 ± 0.043	0.741 ± 0.014
Random Forest Classifier	0.551 ± 0.011	0.236 ± 0.031	0.595 ± 0.018
XGB Classifier	0.667 ± 0.013	0.349 ± 0.030	0.725 ± 0.021
Gradient Boosting Classifier	0.567 ± 0.016	0.259 ± 0.029	0.700 ± 0.030
LSTM	0.594 ± 0.021	0.307 ± 0.056	0.773 ± 0.025
Mean of the above scores	[0.596, 0.625]	[0.261, 0.337]	[0.685, 0.728]
Topology and weather features	-	-	-
Logistic Regression	0.673 ± 0.027	0.380 ± 0.035	0.733 ± 0.035
Random Forest Classifier	0.560 ± 0.019	0.220 ± 0.054	0.770 ± 0.035
XGB Classifier	0.687 ± 0.041	0.394 ± 0.035	0.759 ± 0.047
Gradient Boosting Classifier	0.576 ± 0.018	0.263 ± 0.045	0.776 ± 0.042
LSTM	0.543 ± 0.014	0.164 ± 0.044	0.780 ± 0.018
Mean of the above scores	[0.584, 0.632]	[0.242, 0.327]	[0.728, 0.799]
Weather and operational features	-	-	-
Logistic Regression	0.666 ± 0.013	0.342 ± 0.041	0.739 ± 0.015
Random Forest Classifier	0.546 ± 0.014	0.183 ± 0.038	0.725 ± 0.010
XGB Classifier	0.664 ± 0.017	0.353 ± 0.026	0.725 ± 0.022
Gradient Boosting Classifier	0.577 ± 0.014	0.273 ± 0.023	0.718 ± 0.031
LSTM	0.630 ± 0.053	0.363 ± 0.096	0.776 ± 0.030
Mean of the above scores	[0.594, 0.639]	[0.258, 0.348]	[0.715, 0.758]
Topology and operational features	-	-	-
Logistic Regression	0.708 ± 0.019	0.392 ± 0.032	0.781 ± 0.017
Random Forest Classifier	0.562 ± 0.022	0.223 ± 0.064	0.761 ± 0.059
XGB Classifier	0.712 ± 0.041	0.419 ± 0.030	0.789 ± 0.050
Gradient Boosting Classifier	0.604 ± 0.029	0.331 ± 0.039	0.775 ± 0.064
LSTM	0.571 ± 0.015	0.249 ± 0.042	0.804 ± 0.012
Mean of the above scores	[0.606, 0.657]	[0.281, 0.364]	[0.742, 0.822]
Topology, weather and operational features	-	-	-
Logistic Regression	0.707 ± 0.016	0.393 ± 0.031	0.780 ± 0.015
Random Forest Classifier	0.553 ± 0.019	0.194 ± 0.062	0.789 ± 0.040
XGB Classifier	0.703 ± 0.043	0.416 ± 0.036	0.783 ± 0.049
Gradient Boosting Classifier	0.605 ± 0.025	0.333 ± 0.040	0.783 ± 0.053
LSTM	0.717 ± 0.013	0.406 ± 0.043	0.799 ± 0.016
Mean of above scores	[0.634, 0.680]	[0.306, 0.391]	[0.752, 0.821]

Model performance wise, these results seem to indicate that either XGBoost or LSTM are likely to be among the best performing models. Bootstrapped F1 scores across seasons are visualized in **Figures 4** and **5**. These results do not include the DGNN. As the LSTM by itself seems to perform fairly well, using it in the DGNN's architecture may lead to an increased performance.

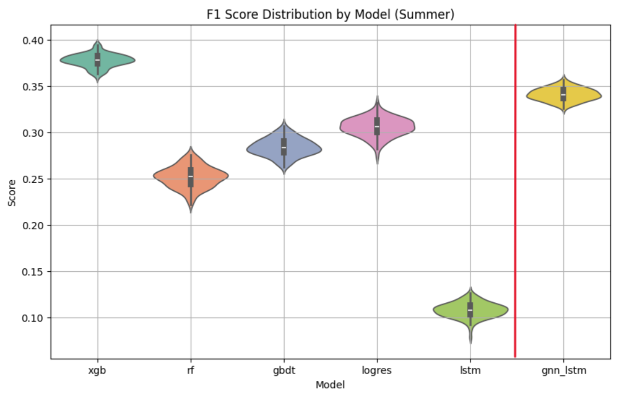


(**a**) Winter.

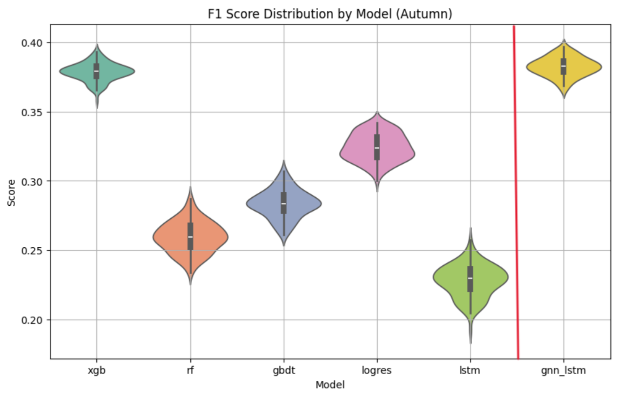


(**b**) Spring.

Figure 4. Bootstrapped F1 scores across seasons (winter and spring).



(**a**) Summer.



(**b**) Autumn.

Figure 5. Bootstrapped F1 scores across seasons (summer and autumn).

4.2. Feature Importance

The first two sub-questions are answered by inspecting the SHAP-values and mean absolute SHAP-values for each model and the permutation importance values of the LSTM models. These SHAP values have been used for feature selection in the GBDT and XGBoost models, using 'shap-select'. The mean absolute SHAP-values have been used for the feature selection for Logistic Regression and Random Forest.

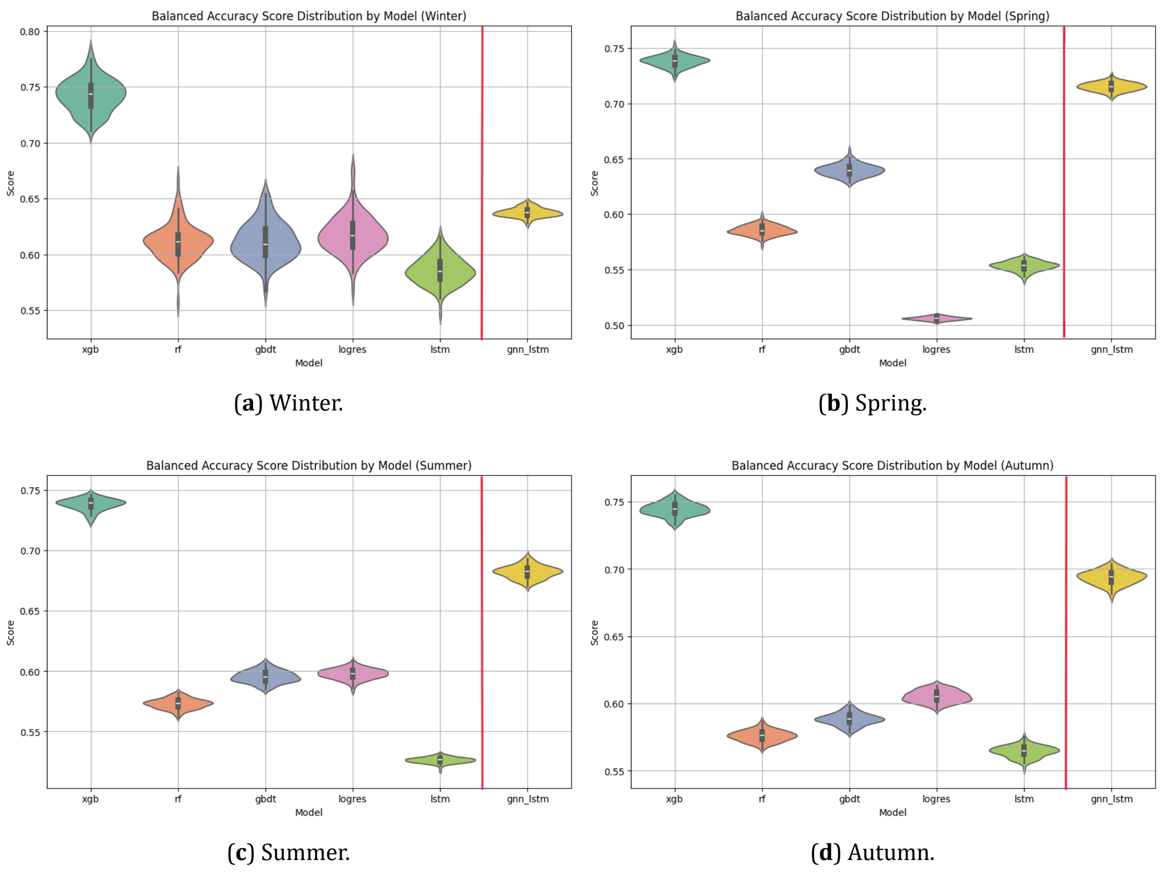
From **Table 6**, it is evident that the weight was selected for every model for every season, and the following most selected features are the three proposed operational features and the distance. Generally, the operational features are most often selected, followed by the topology-based features and then by the weather features. Thus, we can conclude that the features related to the structure of the overall network and those related to previous disruptions are most impactful. The Jaccard Coefficient and VVX features have both been selected only once: the Jaccard Coefficient by Logistic Regression in Autumn and VVX by GBDT in Autumn. The selection of these unpopular features does not seem to impact model performance during these seasons. Overall, the weather features seem to be slightly more impactful than hypothesized after the initial results. None of the temperature features have been selected once during autumn, but TX and TG are quite prevalent across other seasons. Temperature fluctuations are more frequent in autumn than in other seasons, making them less correlated with train cancellations. This pattern likely extends to other weather features: in seasons with stable weather, such as spring or summer, weather conditions appear to have a greater impact on cancellations. Unexpected weather events, like summer storms, are more disruptive because they contrast with the typically consistent conditions of those seasons, unlike in colder or rainier seasons, where worse weather is more expected. The corresponding bootstrapped balanced accuracy scores are shown in **Figure 6**.

Table 6. Feature Occurrences Across Models for Every Season.

Feature	Winter	Spring	Summer	Autumn	Total Occurrences
weight	5	5	5	5	20
Days_since_last_disruption	5	5	5	4	19
Num_prev_disruptions	5	4	5	4	18
Ratio	4	4	4	5	17
distance	3	4	3	4	14
source_weighted_degree	2	5	2	2	11
TX	3	4	4	0	11
target_degree	4	3	2	1	10
preferential_attachment	2	1	3	3	9
target_weighted_degree	1	3	2	2	8
TG	4	3	1	0	8
source_degree	2	2	2	2	8
source_avg_distance	1	1	1	4	7
adamic_adar_index	1	1	2	3	7
common_neighbors	2	1	0	3	6

Table 6. *Cont.*

Feature	Winter	Spring	Summer	Autumn	Total Occurrences
TN	3	1	1	0	5
T10N	1	1	1	1	4
target_avg_distance	1	0	1	2	4
SQ	0	2	2	0	4
RH	1	0	1	1	3
FHVEC	3	0	0	0	3
resource_allocation_index	0	0	1	1	2
RHX	1	0	1	0	2
jaccard_coefficient	0	0	0	1	1
VVX	0	0	0	1	1



(**a**) Winter.

(**b**) Spring.

(**c**) Summer.

(**d**) Autumn.

Figure 6. Bootstrapped Balanced Accuracy Scores across Seasons.

4.3. Model Evaluation Results

The final two sub-questions are answered by bootstrapping the precision, recall, balanced accuracy, F1 score, and AUC metrics of the models over 100 permutations of the training set. The models are compared using an independent t-test on the F1 scores to determine the best-performing model for each season.

4.3.1. Bootstrapped Performance Metrics

The bootstrapped performance metrics visible in **Tables 7–10** indicate that the overall model performance is lacking. All F1 scores are below 0.4, and the balanced accuracy scores are below 0.75 across all seasons. The models seem to be biased towards predicting no disruptions due to the imbalanced nature of the dataset, despite the measures taken to account for data imbalance. Most models tend to predict with a higher recall than precision, which is a more important metric in this task, as discussed earlier, but this could be attributed to chance. XGBoost seems to perform best across all seasons, considering its balanced accuracy scores. This is because the model predicts with a high recall, ranging from 0.75 to 0.80. However, the model's lacking precision decreases its F1 scores, hence why the DGNN outperforms it on the winter and Autumn season.

Table 7. Bootstrapped model performance metrics—Winter (mean with 95% CI).

Model	F1	Precision	Recall	Balanced Acc.	AUC
XGBoost	0.297 (0.266–0.327)	0.186 (0.163–0.208)	0.740 (0.691–0.797)	0.742 (0.717–0.771)	0.830 (0.808–0.857)
Random Forest	0.330 (0.268–0.403)	0.536 (0.431–0.616)	0.239 (0.193–0.312)	0.611 (0.587–0.647)	0.819 (0.795–0.850)
GBDT	0.250 (0.212–0.300)	0.204 (0.168–0.240)	0.324 (0.270–0.397)	0.612 (0.586–0.650)	0.748 (0.721–0.772)
Logistic Regression	0.326 (0.269–0.397)	0.431 (0.349–0.506)	0.263 (0.202–0.332)	0.618 (0.589–0.653)	0.777 (0.745–0.811)
LSTM	0.267 (0.215–0.317)	0.467 (0.359–0.571)	0.188 (0.149–0.231)	0.585 (0.565–0.607)	0.585 (0.565–0.607)
DGNN	0.376 (0.360–0.390)	0.420 (0.399–0.436)	0.340 (0.324–0.357)	0.638 (0.630–0.646)	0.781 (0.772–0.788)

Table 8. Bootstrapped model performance metrics—Spring (mean with 95% CI).

Model	F1	Precision	Recall	Balanced Acc.	AUC
XGBoost	0.364 (0.352–0.374)	0.241 (0.231–0.249)	0.741 (0.724–0.760)	0.738 (0.728–0.747)	0.815 (0.805–0.823)
Random Forest	0.277 (0.259–0.295)	0.465 (0.438–0.495)	0.197 (0.181–0.212)	0.586 (0.578–0.593)	0.816 (0.806–0.825)
GBDT	0.338 (0.322–0.353)	0.308 (0.292–0.323)	0.375 (0.356–0.397)	0.640 (0.630–0.651)	0.790 (0.781–0.801)
Logistic Regression	0.036 (0.026–0.046)	0.231 (0.163–0.301)	0.019 (0.014–0.025)	0.506 (0.503–0.509)	0.754 (0.744–0.765)
LSTM	0.195 (0.169–0.214)	0.408 (0.364–0.451)	0.128 (0.108–0.144)	0.553 (0.544–0.561)	0.553 (0.544–0.561)
DGNN	0.314 (0.304–0.323)	0.194 (0.186–0.200)	0.828 (0.816–0.840)	0.715 (0.708–0.722)	0.784 (0.775–0.792)

Table 9. Bootstrapped model performance metrics—Summer (mean with 95% CI).

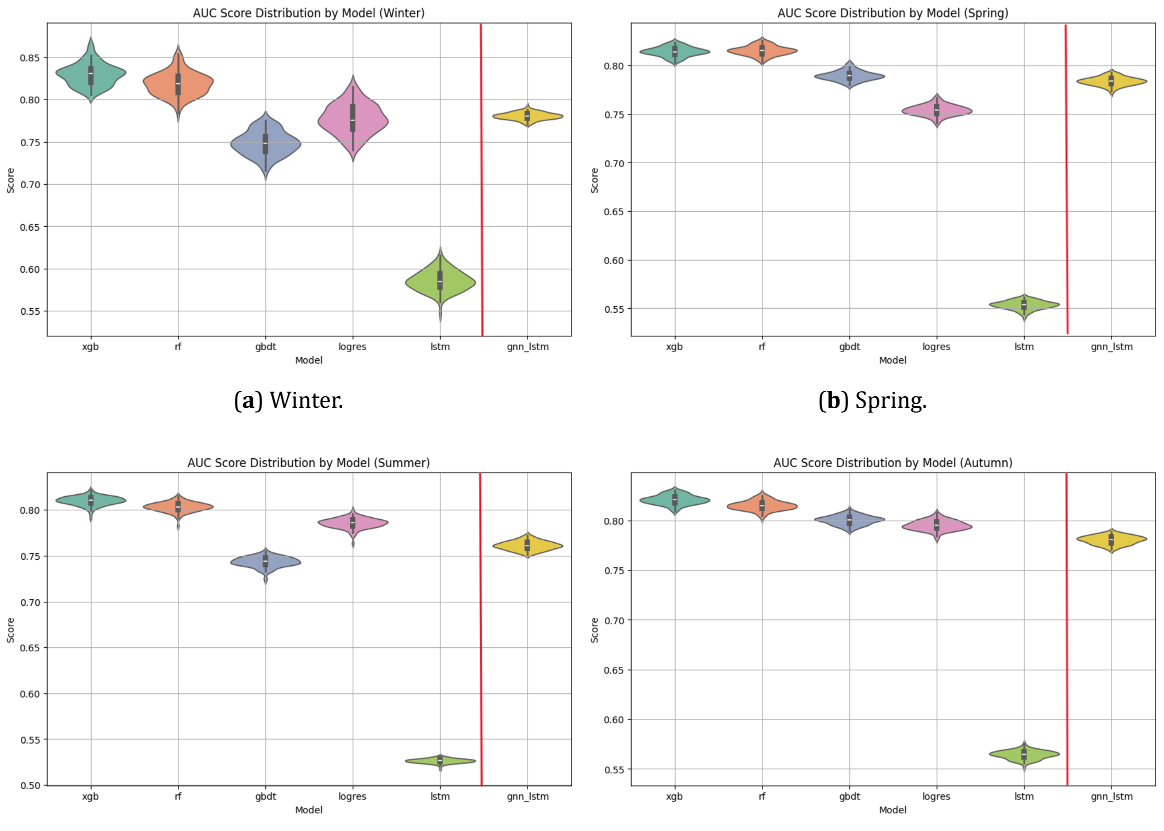
Model	F1	Precision	Recall	Balanced Acc.	AUC
XGBoost	0.378 (0.364–0.393)	0.249 (0.238–0.262)	0.781 (0.759–0.794)	0.738 (0.726–0.746)	0.810 (0.799–0.818)
Random Forest	0.252 (0.228–0.273)	0.539 (0.501–0.579)	0.164 (0.148–0.180)	0.573 (0.564–0.581)	0.803 (0.793–0.813)
GBDT	0.284 (0.266–0.302)	0.286 (0.268–0.306)	0.282 (0.265–0.300)	0.596 (0.586–0.606)	0.743 (0.733–0.753)
Logistic Regression	0.306 (0.286–0.323)	0.441 (0.415–0.470)	0.234 (0.217–0.247)	0.598 (0.589–0.605)	0.786 (0.774–0.795)
LSTM	0.108 (0.093–0.123)	0.493 (0.447–0.535)	0.061 (0.051–0.070)	0.526 (0.522–0.531)	0.526 (0.522–0.531)
DGNN	0.341 (0.329–0.353)	0.236 (0.227–0.246)	0.613 (0.594–0.631)	0.682 (0.672–0.693)	0.761 (0.752–0.771)

Table 10. Bootstrapped model performance metrics—Autumn (mean with 95% CI).

Model	F1	Precision	Recall	Balanced Acc.	AUC
XGBoost	0.379 (0.366–0.390)	0.248 (0.238–0.258)	0.797 (0.778–0.814)	0.744 (0.734–0.754)	0.821 (0.811–0.830)
Random Forest	0.259 (0.240–0.281)	0.558 (0.521–0.599)	0.169 (0.155–0.184)	0.576 (0.569–0.584)	0.815 (0.806–0.825)
GBDT	0.283 (0.264–0.302)	0.413 (0.384–0.440)	0.216 (0.198–0.235)	0.588 (0.580–0.597)	0.800 (0.791–0.809)
Logistic Regression	0.324 (0.307–0.340)	0.482 (0.455–0.510)	0.244 (0.229–0.260)	0.605 (0.598–0.613)	0.795 (0.786–0.805)
LSTM	0.228 (0.207–0.248)	0.487 (0.444–0.522)	0.149 (0.134–0.163)	0.564 (0.557–0.572)	0.564 (0.557–0.572)
DGNN	0.382 (0.369–0.395)	0.285 (0.274–0.296)	0.579 (0.563–0.598)	0.694 (0.684–0.703)	0.780 (0.772–0.788)

4.3.2. *t*-Test Results

Following the t-tests on the bootstrapped F1 scores of the models, XGBoost is the best performing model in the spring and summer datasets, and the DGNN performs the best in winter and autumn. These results are visible in **Figure 7**. In the seasons the DGNN does not achieve the highest F1 scores, it achieves the second or third highest scores. The distributions of the model's scores tend to have less variance than the other models, which, alongside its performance, makes the model robust across seasons. For both summer and autumn, the LSTM is the worst-performing model; this can be attributed to possible under-fitting of the model following feature selection. The bootstrapped F1 scores achieved by Random Forest do not vary strongly across the seasons, unlike the scores achieved by Logistic Regression and GBDT. However, Random Forest is outperformed by GBDT across spring, summer, and autumn, and by Logistic Regression in summer and autumn.

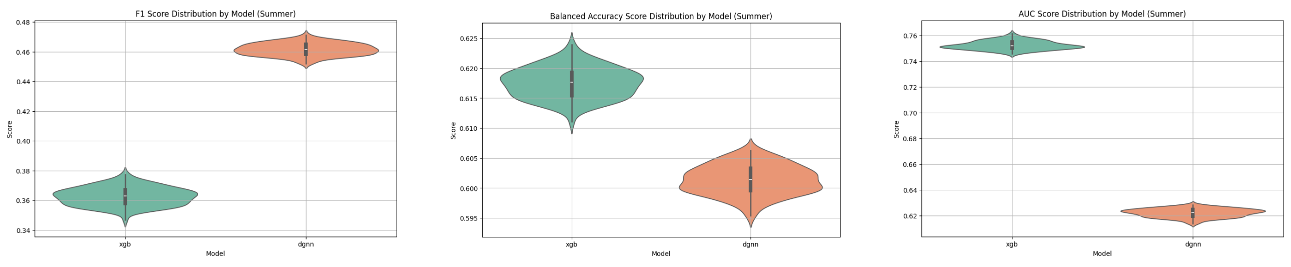


(**a**) Winter. (**b**) Spring. (**c**) Summer. (**d**) Autumn.

Figure 7. Bootstrapped AUC Scores across Seasons.

4.4. External Validation Results

As in the results on the Dutch railway network, both XGBoost and the DGNN outperform the baseline models consistently in the different score metrics. These metrics are displayed in **Figure 8**. The F1 scores achieved by the DGNN are higher than those achieved on the NS dataset, with a more limited feature set. This highlights the potential of the model after further fine tuning in future work.



(**a**) F1 scores. (**b**) Balanced accuracy. (**c**) AUC.

Figure 8. Bootstrapped XGBoost and DGNN scores on the US airways data.

5. Discussion

This work prioritizes robust, season-aware prediction and interpretability. Across seasons, balanced accuracy and AUC are stable, while F1 behaves as expected under severe class imbalance. Compared with Kämper [4], our daily granularity and inclusion of operational features provide complementary insights into disruption drivers. The external experiment following Lei et al. [5], using a reduced and portable feature set, reproduces the relative ordering of methods and indicates cross-network applicability.

5.1. Data

NS does not classify every cancellation as a disruption; this work deviated from that definition, as working with the unaltered dataset proved to be challenging for the models. However, this might raise concerns for the reproducibility of the project. For data concerning railway maintenance, I would require access to the NS API; this might have been useful for a single extra variable in the current feature set. As most of the disruptions in the current dataset are cancellations without a specific cause for those disruptions, the features based on these causes were disregarded entirely.

5.2. Applicability

Although the DGNN performs reasonably well compared to the other models, due to the computational power required to train the model, XGBoost would still be a superior alternative in real-world scenarios, as it achieves comparable results in a fraction of the computational time.

Due to the low F1 scores achieved by the models, stakeholder trust in this project would be quite low. Therefore, the project is unsuitable for immediate real-world application, but the research still provides insight into the effect of features that have not been researched previously. This could potentially aid in constructing a better-performing model to be constructed in the future. The construction of the DGNN could potentially be fine-tuned and applied for different tasks in the future.

The feature selection methods, apart from ‘shap_select’, did not improve model performance, and significantly decreased performance in the case of the LSTM. During the initial results, the LSTM seemed to perform best on the complete feature set, and the permutation importance method significantly decreased its performance.

5.3. External Validity

To align with prior work, the external check follows the monthly aggregation and edge-removal labeling used in related studies, and we report balanced accuracy, AUC, and macro-F1. We restrict to the busiest U.S. airports in summer to keep the graph dense and computations tractable while preserving representative structure. Features are limited to those portable across networks. In this configuration, XGBoost and the DGNN again lead the baseline suite (**Figure 8**). The relative ordering of methods matches the Dutch results, which indicates that the approach transfers across transport domains and temporal resolutions under a common labeling protocol.

Generalisation is shaped by three practical factors. First, operational signals that are highly informative in the railway setting may not be available with the same semantics in aviation; limiting to portable attributes is therefore conservative. Second, daily modeling in rail versus monthly aggregation in aviation changes signal granularity and base rates, which affects recall/precision trade-offs. Third, network topology and reporting practices differ across domains, introducing covariate and label shift. These are expected contextual differences rather than weaknesses of the modeling framework.

Overall, the external check provides evidence that the seasonal evaluation protocol and the DGNN/tree-based modeling strategy are transferable. A natural extension is to broaden coverage to full networks and additional countries using a minimal cross-network feature schema, and to examine domain shift explicitly via rolling-origin experiments, calibration assessment, and simple transfer learning (fine-tuning DGNN/LSTM on target data). This would further quantify how much of the residual gap is due to feature availability versus intrinsic domain differences. Taken together, these results support cautious generalisation: the approach is portable under label and feature schema alignment, while full-network, multi-season aviation tests are a straightforward next step rather than a prerequisite for the present contribution.

6. Conclusions

This comparative study aims to assess model performance and feature importance on a binary classification task. The specific task is predicting whether a given trajectory on the Dutch Railway system will face a cancellation on a given day. This research aimed to find the extent to which implementing environmental and operational factors into a dynamic graph based model, can be used to improve the prediction of disruptions in the Dutch railway system.

We benchmark the proposed dynamic graph-based model (DGNN) against strong tree-based baselines and quantify the added value of environmental and operational signals.

The amount of influence environmental factors such as precipitation and temperature have on disruptions on the Dutch railway system is limited overall. However during seasons with more predictable weather patterns, extreme weather can be more predictive of disruptions. Overall topology features are more influential.

The role of operational factors in exacerbating or mitigating disruptions is more prevalent than topology or weather features, as these features are most often selected to predict disruptions. Expanding these signals is a promising path to higher accuracy.

The overall performance of baseline machine learning models is increased by adding these environmental and operational features. Absolute F1 remains modest, reflecting severe class imbalance and daily granularity.

The DGNN attains the highest F1 in autumn and winter, while XGBoost leads on balanced accuracy and AUC across seasons, indicating complementary strengths.

Beyond seasonal performance, this study contributes a season-aware evaluation protocol and an operational feature set that consistently drives predictive signal, and it shows that a DGNN can complement tree-based methods under class imbalance.

Future Work

Building on these results, promising extensions are: (i) incorporating richer operational signals (e.g., planned maintenance and rolling-stock/crew availability) and higher-resolution meteorology to sharpen season-specific sensitivity; (ii) refining feature selection and attribution tailored to sequence/graph models (LSTM, DGNN) to improve transparency without sacrificing recall; and (iii) stress-testing generalisability via full-scale external validation and decision-focused evaluation (e.g., schedule-recovery benefit).

Author Contributions

Conceptualization, A.M.M.A. and B.B.; methodology, B.B.; software, B.B.; validation, A.M.M.A.; formal analysis, B.B.; investigation, B.B.; resources, A.M.M.A. and B.B.; data curation, B.B.; writing—original draft preparation, B.B.; writing—review and editing, A.M.M.A. and S.S.M.Z.; visualization, B.B.; supervision, A.M.M.A.; project administration, A.M.M.A. All authors have read and agreed to the published version of the manuscript.

Funding

This work received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

Public raw data were obtained from Rijden de Treinen [1–3] and KNMI [26]. The US Airways dataset follows the construction in Lei et al.'s research [5]. The processed datasets and scripts to reproduce the tables and figures will be made available in an online repository upon acceptance; until then, they are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Rijden de Treinen. Available online: https://www.rijdendetreinen.nl/ (accessed on 18 November 2024).
2. Rijden de Treinen. Statistics for 2024. Available online: https://www.rijdendetreinen.nl/en/statistics/2024 (accessed on 13 January 2025).
3. Rijden de Treinen. Statistics for 2023. Available online: https://www.rijdendetreinen.nl/en/statistics/2023 (accessed on 13 January 2025).
4. Kämper, M. Predicting Delayed Trajectories Using Network Features: A Study on the Dutch Railway Network. *arXiv preprint* **2025**, *arXiv:2507.11776*. [CrossRef]
5. Lei, W.; Alves, L.G.A.; Amaral, L.A.N. Forecasting the Evolution of Fast-Changing Transportation Networks Using Machine Learning. *Nat. Commun.* **2022**, *13*, 4252. [CrossRef]
6. Dekker, M.M.; Panja, D.; Dijkstra, H.A.; et al. Predicting Transitions across Macroscopic States for Railway Systems. *PLoS ONE* **2019**, *14*, 1–26. [CrossRef]
7. Kecman, P.; Goverde, R.M.P. Online Data-Driven Adaptive Prediction of Train Event Times. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 465–474. [CrossRef]
8. Li, Z.; Wen, C.; Hu, R.; et al. Near-Term Train Delay Prediction in the Dutch Railways Network. *Int. J. Rail Transp.* **2020**, *9*, 520–539. [CrossRef]
9. Zhang, Y.D.; Liao, L.; Yu, Q.; et al. Using the Gradient Boosting Decision Tree Algorithm for a Train Delay Prediction Model Considering the Delay Propagation Feature. *Adv. Prod. Eng. Manag.* **2021**, *16*, 285–296. [CrossRef]
10. Wen, C.; Mou, W.; Huang, P.; et al. A Predictive Model of Train Delays on a Railway Line. *J. Forecast.* **2020**, *39*, 470–488. [CrossRef]
11. Zheng, Y.; Yi, L.; Wei, Z. A Survey of Dynamic Graph Neural Networks. *arXiv preprint* **2024**, *arXiv:2404.18211*.[CrossRef]
12. Luo, X.; Zhu, C.; Zhang, D.; et al. STG4Traffic: A Survey and Benchmark of Spatial-Temporal Graph Neural Networks for Traffic Prediction. *arXiv preprint* **2023**, *arXiv:2307.00495*. [CrossRef]
13. Xue, J.; Tan, R.; Ma, J.; et al. Data Science in Transportation Networks with Graph Neural Networks: A Review and Outlook. *Data Sci. Transp.* **2025**, *7*, 10. [CrossRef]
14. Fu, Q.; Ding, S.; Zhang, T.; et al. Short-Term Train Arrival Delay Prediction: A Data-Driven Approach. *Railw. Sci.* **2024**, *3*, 514–529.
15. Raghavendran, P.; Gochhait, S.; Gunasekar, T. Optimizing Load Dispatch Using Artificial Neural Networks and Fractional Weighted Models. *J. Vis. Exp.* **2025**, 68811. [CrossRef]
16. Gori, M.; Monfardini, G.; Scarselli, F. A New Model for Learning in Graph Domains. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005. [CrossRef]
17. Scarselli, F.; Gori, M.; Tsoi, A.C.; et al. The Graph Neural Network Model. *IEEE Trans. Neural Netw.* **2009**, *20*, 61–80.
18. Feng, Z.; Wang, R.; Wang, T.; et al. A Comprehensive Survey of Dynamic Graph Neural Networks: Models, Frameworks, Benchmarks, Experiments and Challenges. *arXiv preprint* **2024**, *arXiv:2405.00476*. [CrossRef]
19. Veličković, P.; Cucurull, G.; Casanova, A.; et al. Graph Attention Networks. *arXiv preprint* **2018**, *arXiv:1710.10903*. [CrossRef]
20. Furno, A.; Fanouzi, N.-E.E.; Sharma, R.; et al. Graph-Based Ahead Monitoring of Vulnerabilities in Large Dynamic Transportation Networks. *PLoS ONE* **2021**, *16*, 1–35. [CrossRef]
21. Peng, H.; Du, B.; Liu, M.; et al. Dynamic Graph Convolutional Network for Long-Term Traffic Flow Prediction with Reinforcement Learning. *Inf. Sci.* **2021**, *578*, 401–416. [CrossRef]
22. Kroev, E.; Koseoglu, B.; Traverso, L.; et al. Shap-Select: Lightweight Feature Selection Using SHAP Values and Regression. *arXiv preprint* **2024**, *arXiv:2410.06815*. [CrossRef]
23. Altmann, A.; Toloşi, L.; Sander, O.; et al. Permutation Importance: A Corrected Feature Importance Measure. *Bioinformatics* **2010**, *26*, 1340–1347. [CrossRef]
24. Oneto, L.; Fumeo, E.; Clerico, G.; et al. Train Delay Prediction Systems: A Big Data Analytics Perspective. *Big Data Res.* **2018**, *11*, 54–64. [CrossRef]
25. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
26. Koninklijk Nederlands Meteorologisch Instituut. About KNMI. Available online: https://www.knmi.nl/nederland-nu/klimatologie/daggegevens (accessed on 15 January 2025).
27. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; et al. SMOTE: Synthetic Minority Over-Sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357.

28. Wu, J.; Chen, X.-Y.; Zhang, H.; et al. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. *J. Electron. Sci. Technol.* **2019**, *17*, 26–40.