Enhancing Prediction of Disruptions of the Dutch Railway Network Using Environmental and Operational Data

submitted in partial fulfillment for the degree of master of science

Brent Brakenhoff
13068989

Master Information Studies
data science
Faculty of Science
University of Amsterdam

Submitted on 27.06.2025

	UvA Supervisor
Title, Name	Dr. A.M.M. Alsahag, PhD
Affiliation	UvA Supervisor
Email	a.m.m.alsahag@uva.nl

ABSTRACT

Cancellations on the Dutch Railway network are a common and unpredictable occurrence, however little research has been put into predicting these cancellations. Past research on the Dutch railway system has mainly focused on forecasting delays. For this regression task, models such as XGBoost, Random Forest, LSTM, and Gradient Boosting Decision tree have 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 based model combined with an LSTM (DGNN) for binary classification of cancelled trajectories. We compare the model with baseline models on a seasonal split of to compare the feature importance across different seasons. Model performance is gauged using paired t-tests on bootstrapped F1 score. 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 on the influence of various weather and operational features on cancellations on the Dutch railway network, with especially the operational features proving insightful.

KEYWORDS

railway network, comparative study, binary classification, time series, graph neural network, lstm, xgboost

GITHUB REPOSITORY

https://github.com/Brebber/Thesis/

1 INTRODUCTION

In 2024, nearly 6000 disruptions have been reported on the Dutch railway system [3]. Disruptions are defined by an unplanned occurrence that temporarily hinders a train moving from one station to the next, interrupting the trajectory the train operates within. 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 temporary form of edge removal, as the connection between two nodes is temporarily interrupted.

Train disruptions are most often caused by broken down trains, followed by infrastructure issues, accidents, external influences and engineering work [3]. Due to the complexity the unpredictable nature of some of these causes, predicting the occurrence of disruptions based on topological features alone becomes challenging [12]. However as the definition of disruption provided by Dutch Railways (NS) does not include all interruptions of 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 [13]. Lei et al. (2022) provide 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 [12]. Due to this decreased performance, we propose novel external factors in addition to the topological factors that were examined in Kämper (2024). 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 [4]. 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 a hour in past research [10, 14, 20, 22]. 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. 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?

In order 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.

2 RELATED WORK

As mentioned previously, most research on the Dutch railway system concerns real time forecasting or predictions on an at most 90 minute 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 in multiple occasions [10, 14, 20, 22]. This paper is mainly built upon the foundation of Lei et al. (2022), 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 [13]. 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, thus to utilise the proposed methodology in Lei et al. (2022), 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 improve upon the model proposed in Kämper (2024), wherein a topology based solution was proposed to predict significantly delayed trajectories on a monthly time scale in the Dutch railway system [12]. 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 possibly be mitigated by grouping the data into single rides instead of trajectories or viewing the trajectories on a smaller time scale, and the feature limitation can be mitigated by taking external factors into account.

Concerning short-term prediction of disruptions on the Dutch railway system, Dekker, Panja & Dijkstra et al. (2019) propose a data-driven approach that identifies different operational states in the network and provides an early-warning metric based on the probability a section of the network transitions into a disrupted state [4]. The model proposed in this paper is trained and validated on the data of one year, June 2017 to July 2018; uses two principle components for its predictions; and the model does not take memory of the system into account, which potentially would have improved the accuracy of the predictions. These limitations cause inaccurate results, however Dekker et al. (2019) prioritise identifying underlying structures of the social-technical system over providing accurate predictions.

2.2 Graph Neural Networks

RNNs are able to perform machine learning tasks on graph structures, however this 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 [7, 18]. Feng, Wang, Wang et al. (2024) have created an overview of use cases, performance and computational efficiency of over 80 different dynamic graph neural networks, trained on various datasets [5]. This overview shows the versatility of dynamic graph neural networks, and provides metrics for evaluation of the models in different scenarios. The discussion of DGNNs in transportation settings in this overview is limited, however.

A Graph Attention Network (GAT) differs from a regular GNN by taking the state of neighbouring nodes into account while performing node classification [19]. Originally, GATConv was not designed specifically for edge classification, but due to the flexibility of the model and its properties it is suitable for the task.

Graph based models have been used to forecast event times in the Dutch railway system in the past. In Kecman & Goverde (2015), which proposed an dynamic graph based model for predicting train event times in real time [10]. 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 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 [6, 17]. In Purno, El Faouzi et al. (2021), 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 betweenness centrality of the nodes in the network, which indicates which nodes are most often traversed. The node centrality metric proves to be a good measurement for traffic flow, and the overall method proposed to construct 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, Du et al. proposes a dynamic graph neural network to forecast long-term traffic flow, tested on city-bike data in New York city [17]. 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 a baseline to compare to the DGNN. These models have all been used in either regression or classification tasks on the Dutch railway network.

Wen, Mou et al. (2020) propose an LSTM model and compare multiple models for short term delay prediction on the Dutch railway system [14]. These benchmark models include and Artificial Neural

Network (ANN) and a Random Forest model. The research suggests that the proposed LSTM model is 20% more accurate than the proposed ANN. The difference in performance between the LSTM and the RF model is not as significant however. Thus based on this paper, random forest seems to be a more suitable baseline model than an ANN, and an LSTM would be a suitable baseline model as it outperforms both. Li, Wen et al. (2020) have found that Random Forest model performs best on delay prediction on a 20 minutes time frame [20]. This paper considered a single feature to summarise the weather conditions of a given day. This weather feature was later disregarded as the weather summary of a given day did not seem to affect the trajectories on a smaller time frame. Although this paper concludes that the proposed Random Forest model performs best on the tasks, it shows that XGBoost and GBDT only have slightly worse results. XGBoost and RF seemed to perform comparably using the MAE and RMSE metrics, with GBDT following, and XGBoost outperformed RF on precision on predicting delays shorter than or equal to 3 minutes. The proposed ANN in this paper is outperformed by all three models, with significantly longer training and loading times. Similarly, Zhang, Liao et al. (2021) performed this task on the UK railway network, proposing a GBDT model [22]. This paper uses Random Forest, a Support Vector Regression (SVR) model and a Multi Layer Perceptron (MLP) as comparisons.

Concerning a classification task performed on graph structures, both Lei et al. (2022) and Kämper (2024) propose the usage of XGBoost for an edge removal task [12, 13]. These papers include a multitude of baseline models to gauge the performance of XGBoost. Amongst these baseline models, Gradient Boosting Decision Tree (GBDT) and Random Forest seem to achieve comparable results to XGBoost in Kämper (2024). Logistic Regression surprisingly seemed to outperform the rest of the models on the topology feature set on the Dutch railway network. In Lei, et al. (2022), XGBoost outperforms the other baseline models. This paper did not consider any models based on neural networks. The main strengths of tree based models, XGBoost, GBDT and Random Forest, are that these can identify non linear connections between features, which is vital for this research. LSTMs are often used for time series forecasting and able to sequential patterns, which can provide insights on the recurring nature of some disruptions. Overall these discussed methods provide a baseline for the evaluation of the proposed dynamic graph based model, although the specific methods are most often applied for regression instead of classification.

2.4 Feature Selection

In order to represent and visualise the feature importance of the models, the SHAP values of these features are calculated [15]. These are values assigned to features based on their influence on a model during a particular prediction task. These values add important insight on the predictive tendencies of the models, as not every model's predictions are equally intuitive, and thus increase 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 [11]. 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 being 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 models 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 [1]. This approach is especially suitable for the LSTM model as it does not require any insight of the model's internal architecture. This method was originally designed for Random Forest, however due to its model-agnostic nature, it is a common method for calculating feature importance for LSTM models and other Neural Networks. Due to 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.

3 METHODOLOGY

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 2.

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 [12, 16, 20]. The main difference between these

Paper	Dataset(s)	Task	Proposed Model	Baseline	Time
Kecman & Goverde (2015) [10]	NL Rail	Event Time Prediction	DFS over Graph	-	RT
Purno, El Faouzi, et al. (2021) [6]	French Highway	BC Computation	Dynamic Graph Based Modelling	Various BC computation algorithms	RT
Peng, Du et al. (2021) [17]	NYC Bikes	Traffic Flow	DGNN (GCN + LSTM)	ARIMA, SVR, LSTM, Various GNNs	D
Wen, Mou et al. (2020) [14]	NL Rail	Regression	LSTM	RF, ANN	RT
Li, Wen et al. (2020) [20]	NL Rail	Regression	RF	XGBoost, GBDT, ANN, Statistical models	RT
Zhang, Liao et al. (2021) [22]	UK Rail	Regression	GBDT	RF, SVR, MLP	RT
Lei, et al. (2022) [13]	US Air, Brazil Bus	Edge Removal	XGBoost	RF, LR, GBDT, Ridge Classifier, etc.	M
Kämper (2024) [12]	NL Rail, US Air	Edge Classification	XGBoost	LR, RF, GBDT	M
This paper	NL Rail (US Air)	Edge Classification	DGNN (GatConv + LSTM)	LR, GBDT, XGBoost, LSTM	D

Table 1: Proposed and Baseline Models in Related Works.
The time column indicates the time scale, which is either RT for real time, D for daily or M for monthly.

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 [13, 14].

3.1.2 LSTM. An LSTM is chosen over a Classic Neural Network (CNN) or Recurring Neural Network (RNNs), for the architecture's ability to store and learn from past information [8]. This enables the ability to interpret patterns across the dataset, and therefore if the same patterns of cancellations occurs over a longer period of time, an LSTM should be able to take that pattern into consideration during future predictions. LSTMs are also robust against irregularities and outliers in the dataset and able to interpret non-linear patterns between variables. An LSTM model has outperformed ANNs and Random Forest 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 [17, 20].

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 on a daily basis, a Graph Neural Network (GNN) is selected, specifically using a Graph Attention Network (GATConv) based architecture. GATConv layers use an attention mechanism, visible in figure 1, to weigh 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

$$\text{GATConv:} \quad \mathbf{h}_i' = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right)$$
$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_k]\right)\right)}$$

Figure 1: GATConv attention mechanism with weight matrix W and attention vector a.

combined in a context-aware manner to predict edge properties.
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 [5]. The graph convolution layer captures dependencies between stations, the temporal layer models the daily evolution of the graph, and finally an the output layer. For this research, GATConv layers are chosen to form the spatial layer, due to their discussed ability to weigh importance of neighbouring node features. An LSTM is chosen for the temporal layer, following similar reasoning as during its selection as a baseline model [17, 20]. 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. 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 rijdendetreinen.nl: 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 [3]. The Services dataset are 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 different lines. Using the start and end time, the number of disruptions per line per day is added, and the statistical causes are stored into a list. The affected lines are split into start and target of the trajectory, in order to merge the dataset with the daily trajectory dataset. After merging these datasets, it became apparent some trajectories have either partial or complete cancellations, but no disruption was reported in the disruptions dataset. These cancellations are added manually in order 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.

3.2.1 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 for the weather in the Netherlands, also a set of 10 features that are suspected to affect the operation schedule of trains the most are 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 degrees Celsius); 'TN', the minimum temperature (in 0.1 degrees Celsius); 'TX', maximum temperature (in 0.1 degrees Celsius); and 'T10N', minimum temperature at 10 cm above surface (in 0.1 degrees Celsius). The features related to precipitation are: 'SQ', the sunshine duration (in 0.1 hour) calculated from global radiation (-1 for <0.05 hour); 'DR', the precipitation duration (in 0.1 hour), '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) [9].
Using the Haversine formula between the coordinates of the train stations and the weather stations to find the 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 3.
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 amount of days since the last disruption, the amount of disruptions previously to that moment and the amount of past disruptions divided by the amount of rides planned on this trajectory in the past. These daily features are stored in a dataset which will be used to train the discussed models.

3.3.1 EDA. The final dataset thus consists of 26 columns, and 612790 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 if that trajectory has been disrupted during that day. Of all daily trajectories in the dataset, only 13.2% have been disrupted or cancelled.
Of all planned trajectories, 6.14% has never been disrupted or cancelled, these trajectories are deleted as the mean and median of planned rides amongst these trajectories was a lot lower that 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 initial stages, using a time-series split, we split the data into 6 partitions of equal size, maintaining chronological order. These partitions are in turn split into a train and test set using a 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, so for topology features only, topology and weather features, topology and operational features, etc. Initially, SMOTE was considered as an option to accommodate for data imbalance, however as SMOTE is not aware of temporal and topological structures of data, which are crucial in the dataset, it was deemed unsuitable for this project [2]. Balancing class weights for the models is a more suitable option.
For the final results, the data is split into 4 different datasets by seasons 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 year. Allowing us to create different models specifically designed for different seasons. Originally, a monthly split was considered, however the overall distribution of cancellations did not seem to differ enough on a monthly basis, and this would have lead to smaller training sets and this would possibly have lead to ambiguity due to the amount of differently configured models, each could have their 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 spans from December 21st 2019 to March 20th 2020. This however 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 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 2, 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.

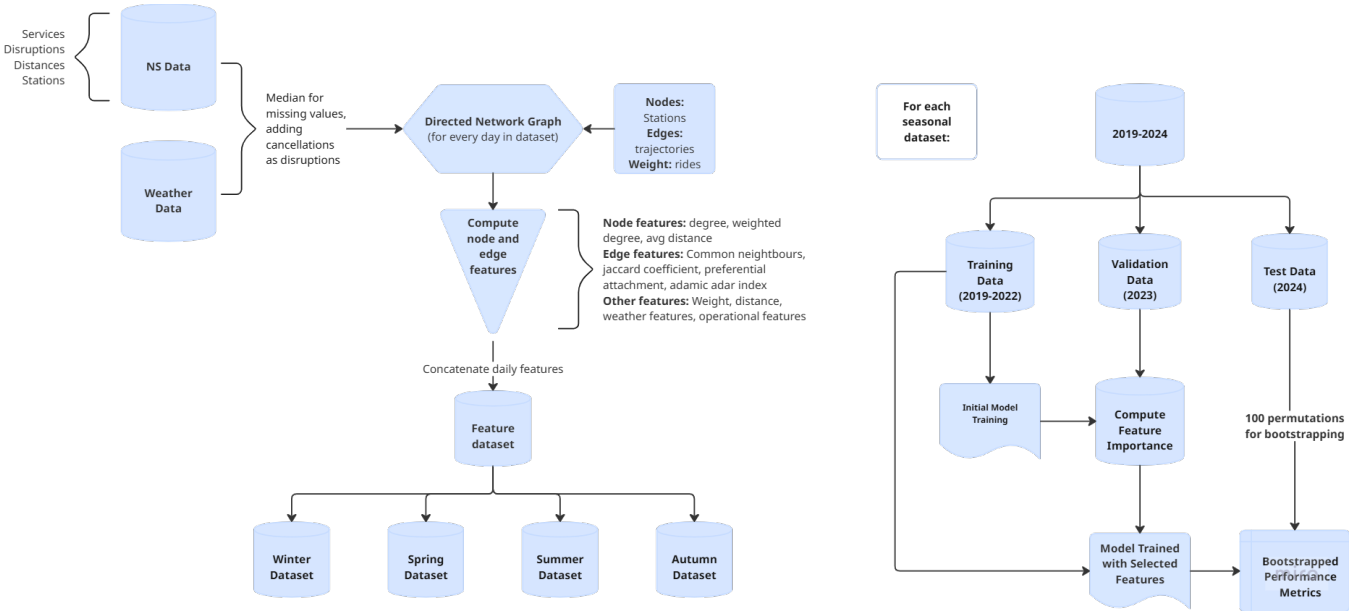


Figure 2: Overview of Proposed Methodology

3.4.2 Model Configuration. The hyperparameters of the models are optimised using random search, as in Kämper (2024) [12]. However Bayesian optimisation is used to tune the LSTM hyperparameters, as it is more computationally efficient [21]. The selection of hyperparameters and their values are visible in table 2. 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.

3.4.3 Logistic Regression. Logistic regression has the least amount of hyperparameters out of the selected models. The ones that are tuned are 'C', 'penalty', '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, 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.

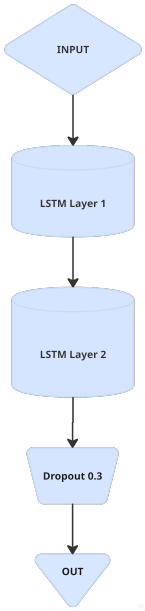


Figure 3: LSTM Model Architecture

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 an densely connected output layer with a Softmax activation function. The variable sizes of the layers are tuned using Bayesian optimisation, alongside the learning rate of the network. These parameters are visible in table

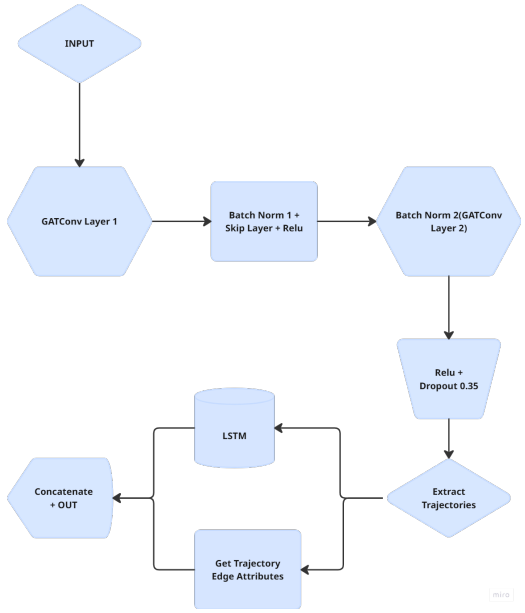


Figure 4: DGNN Model Architecture

2, and the final results for each season are visible in table 4. Early stopping is implemented to prevent over-fitting after 5 epochs.

3.4.6 DGNN. 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 head 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.

3.5 Evaluation

The 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 [13]. 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 [11]. 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 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 [1]. 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 [8].

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 F! scores of the models are compared using an independent t-test.

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 [5]. Additionally, utilizing these metrics enables clear comparison to previous work, as these have been used to evaluate the models in Kämper (2024) and Lei et al. (2022) [12, 13].

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. (2022) [13]. 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 1270480 rows. As a target variable, we take the removal of edges in the dataset, as in Lei et al. (2022). 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.

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

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': {1e-4, 1e-2}, 'lstm1': {32, 256}, 'lstm2': {16, 128}

Table 2: Hyperparameter grids for each model

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.

Model performance wise, these results seem to indicate that either XGBoost or LSTM are likely to be among the best performing models. 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.

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. The exact feature importance metrics are visible in section G of the appendix, alongside the selected features for each model for each season.

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 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 that 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.

4.3 Model Evaluation Results

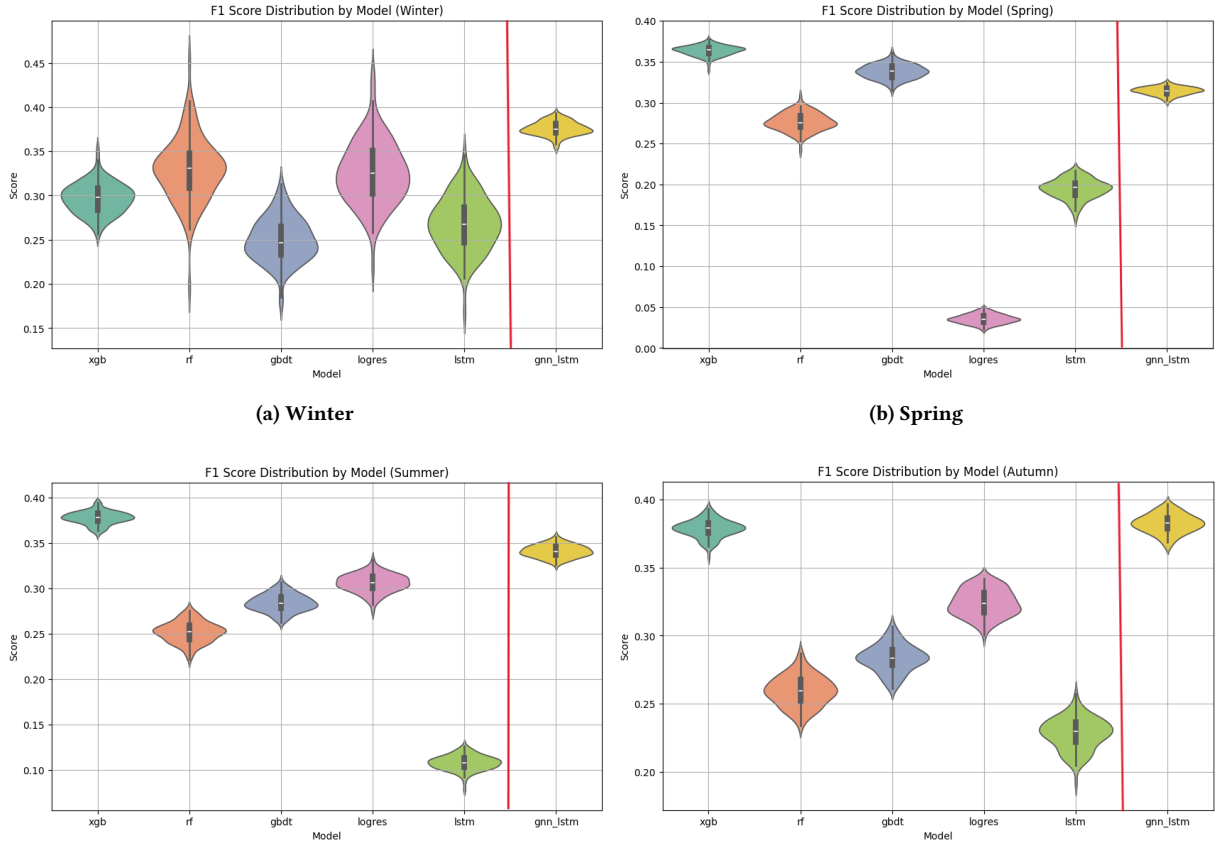
The final two sub-questions are answered by bootstrapping the precision, recall, balanced accuracy, F1 score and AUC metrics of the models over 100 permutations of the training set. The models are compared using an independent t-test on the F1 scores to determine the best performing model for each season.

4.3.1 Bootstrapped Performance Metrics. The bootstrapped performance metrics visible in tables 12, 13, 14 and 15 indicate that the overall model performance is lacking. All F1 score 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 due to the fact that 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.

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, the DGNN performs the best on winter and autumn. These results are visible in section H of the appendix. In the seasons the DGNN does not achieve the highest F1 scores, it achieves the second or third highest scores. The distributions of the models scores tends 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



(a) Winter **(b) Spring**

(c) Summer **(d) Autumn**

Figure 5: Bootstrapped F1 Scores across Seasons

by GBDT across spring, summer and autumn, and by Logistic Regression in summer and autumn.

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 7. 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.

5 DISCUSSION

Overall this project proposes models that achieve subpar results. Compared to Kämper (2024), the bootstrapped F1 scores of the proposed models are nearly 50% lower [12]. However the balanced accuracy score of the models in this paper seem to be higher for the better performing models. During the external validation, all proposed models are outperformed by the models in Lei, et al. (2022) [13]. This can be attributed to the limited feature set that was selected for the external validation.

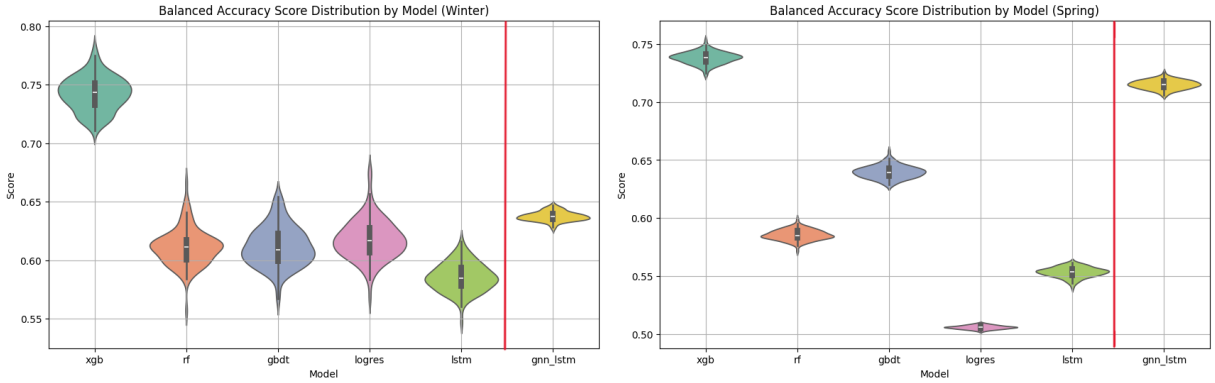
5.1 Data

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 reproducibility of the project. For data concerning railway maintenance I would require access to the NS API, however 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

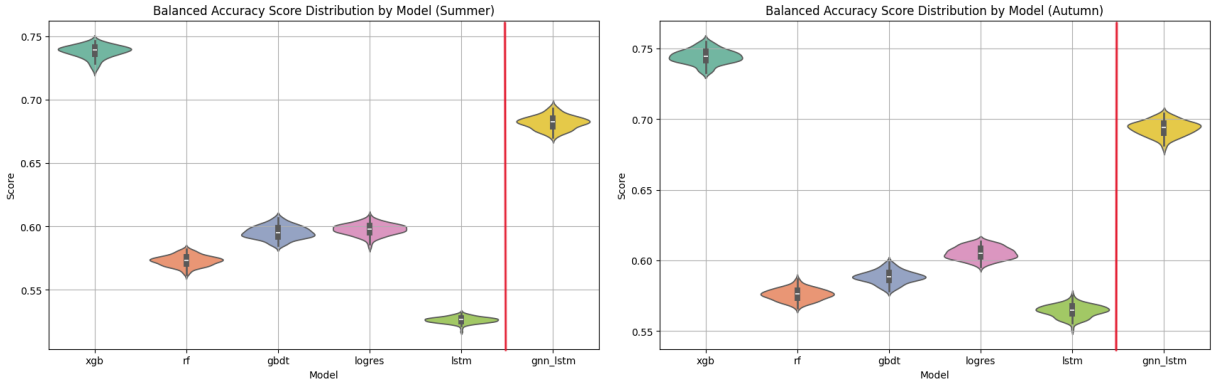
Although the DGNN performs reasonably well compare to the other models, due to the computational power required to train the model, XGBoost would still be superior alternative in real-world scenarios, as it achieves comparable results in a fraction of the computational time.

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, however the research still provides insight on the effect of features that have not been



(a) Winter

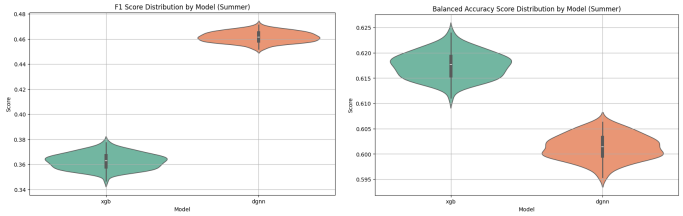
(b) Spring



(c) Summer

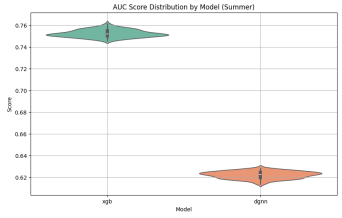
(d) Autumn

Figure 6: Bootstrapped Balanced Accuracy Scores across Seasons



(a) F1 Scores

(b) Balanced Accuracy Scores



(c) AUC Scores

Figure 7: Bootstrapped XGBoost and DGNN Scores on US Airways Data

researched yet previously. This could potentially aid 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

As the external validation was not performed on the entirety of the US Airways dataset due to time constraints, the scalability of the proposed methodology is inconclusive. We also did not consider the newly proposed features during external validation, which resulted in further under-fitting in the baseline models. XGBoost and the DGNN seemed to achieve comparable results with the external dataset. However, as the models still underperform this is a weak argument for generalisability.

6 CONCLUSION

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. However the proposed model falls short compared to XGBoost.
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. Further expanding on these features would potentially lead to more accurate forecasting of disruptions.
The overall performance of baseline machine learning models is increased by adding these environmental and operational features. However the achieved performance is lacking compared to previous work.
The proposed Dynamic Graph Based model outperforms baseline models on the F1 score metric, using the newly proposed features in the autumn and winter seasons, however XGBoost outperforms the model in the remaining seasons. XGBoost outperforms the model on both balanced accuracy and AUC metrics across all seasons.
Compared to the state-of-the art, the results of this paper are underwhelming, however there are some limited contributions to be found. The proposed operational features seem to have an impact on model prediction, and the framework for the DGNN could be utilised in future work after further research. The immediate next steps following this research would be to find more operational features for the Dutch Railway system. Features concerning maintenance for example, would require access to the NS API to be utilised. A next possible step would be finding a suitable feature selection method for the LSTM and DGNN. Lastly the external validation on the US Airways dataset could be carried out in full, instead of on a subset of the data.

REFERENCES

[1] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. 2010. Permutation importance: a corrected feature importance measure. *Bioinformatics* 26, 10 (04 2010), 1340–1347. https://doi.org/10.1093/bioinformatics/btq134 arXiv:https://academic.oup.com/bioinformatics/article-pdf/26/10/1340/48851160/bioinformatics_26_10_1340.pdf

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[3] Rijden de Treinen. 2010-2025. *ww.rijdendetreinen.nl.* https://www.rijdendetreinen.nl/

[4] Mark M. Dekker, Debabrata Panja, Henk A. Dijkstra, and Stefan C. Dekker. 2019. Predicting transitions across macroscopic states for railway systems. *PLOS ONE* 14, 6 (06 2019), 1–26. https://doi.org/10.1371/journal.pone.0217710

[5] ZhengZhao Feng, Rui Wang, TianXing Wang, Mingli Song, Sai Wu, and Shuibing He. 2024. A Comprehensive Survey of Dynamic Graph Neural Networks: Models, Frameworks, Benchmarks, Experiments and Challenges. *arXiv e-prints*, Article arXiv:2405.00476 (May 2024), arXiv:2405.00476 pages. https://doi.org/10.48550/arXiv.2405.00476 arXiv:2405.00476 [cs.LG]

[6] Angelo Furno, Nour-Eddin El Faouzi, Rajesh Sharma, and Eugenio Zimeo. 2021. Graph-based ahead monitoring of vulnerabilities in large dynamic transportation networks. *PLOS ONE* 16, 3 (03 2021), 1–35. https://doi.org/10.1371/journal.pone.0248764

[7] M. Gori, G. Monfardini, and F. Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 2. 729–734 vol. 2. https://doi.org/10.1109/IJCNN.2005.1555942

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[9] Koninklijk Nederlands Meteorologisch Instituut. 2025. *KNMI daggegevens.* https://www.knmi.nl/nederland-nu/klimatologie/daggegevens

[10] Pavle Kecman and Rob M. P. Goverde. 2015. Online Data-Driven Adaptive Prediction of Train Event Times. *IEEE Transactions on Intelligent Transportation Systems* 16, 1 (2015), 465–474. https://doi.org/10.1109/TITS.2014.2347136

[11] Egor Kraev, Baran Koseoglu, Luca Traverso, and Mohammed Topiwalla. 2024. Shap-Select: Lightweight Feature Selection Using SHAP Values and Regression. *arXiv e-prints*, Article arXiv:2410.06815 (Oct. 2024), arXiv:2410.06815 pages. https://doi.org/10.48550/arXiv.2410.06815 arXiv:2410.06815 [cs.LG]

[12] Merel Kämper. 2024. Predicting Delayed Trajectories Using Network Features: A Study on the Dutch Railway Network. (2024). https://github.com/merelkaemper/MSc-Thesis-main

[13] Weihua Lei, Luiz G. A. Alves, and Luís A. Nunes Amaral. 2022. Forecasting the evolution of fast-changing transportation networks using machine learning. *Nature Communications* 13, 1 (2022), 4252. https://doi.org/10.1038/s41467-022-31911-2

[14] Zhongcan Li, C. Wen, Rui Hu, Chuanlin Xu, Ping Huang, and Xi Jiang. 2020. Near-term train delay prediction in the Dutch railways network. *International Journal of Rail Transportation* 9 (11 2020), 1–20. https://doi.org/10.1080/23248378.2020.1843194

[15] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, Vol. 30. https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[16] Luca Oneto, Emanuele Fumeo, Giorgio Clerico, Renzo Canepa, Federico Papa, Carlo Dambra, Nadia Mazzino, and Davide Anguita. 2018. Train Delay Prediction Systems: A Big Data Analytics Perspective. *Big Data Research* 11 (2018), 54–64. https://doi.org/10.1016/j.bdr.2017.05.002 Selected papers from the 2nd INNS Conference on Big Data: Big Data & Neural Networks.

[17] Hao Peng, Bowen Du, Mingsheng Liu, Mingzhe Liu, Shumei Ji, Senzhang Wang, Xu Zhang, and Lifang He. 2021. Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning. *Information Sciences* 578 (2021), 401–416. https://doi.org/10.1016/j.ins.2021.07.007

[18] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[19] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations.* https://openreview.net/pdf?id=rJXMpikCZ

[20] Chao Wen, Weiwei Mou, Ping Huang, and Zhongcan Li. 2020. A predictive model of train delays on a railway line. *Journal of*

Forecasting 39, 3 (2020), 470–488. https://doi.org/10.1002/for.2639 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/for.2639

[21] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. 2019. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. *Journal of Electronic Science and Technology* 17, 1 (March 2019), 26–40. https://doi.org/10.11989/JEST.1674-862X.80904120

[22] Y.D. Zhang, L. Liao, Q. Yu, W.G. Ma, and K.H. Li. 2021. Using the gradient boosting decision tree (GBDT) algorithm for a train delay prediction model considering the delay propagation feature. *Advances in Production Engineering & Management* 16 (09 2021), 285–296. https://doi.org/10.14743/apem2021.3.400

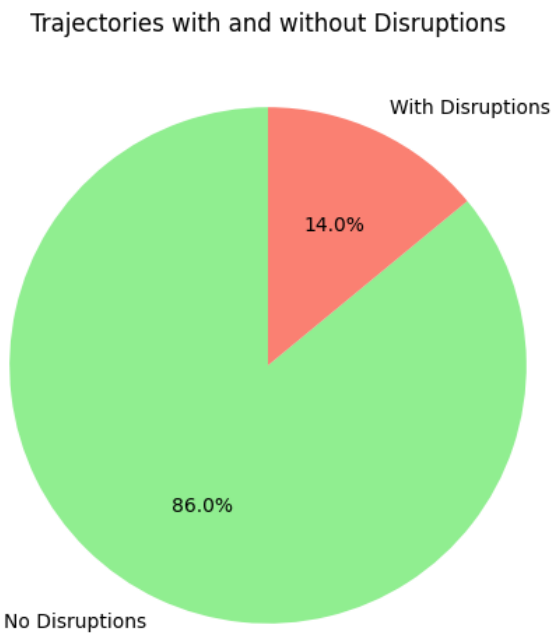


Figure 8: Ratio Not Disrupted vs Disrupted

Appendix A ACKNOWLEDGMENT CONCERNING USAGE OF GENERATIVE AI

In this paper, generative AI was utilised as a tool to format code and quickly resolve small coding errors. In the actual text I only used AI to provide suggestions concerning the grammar and structure in a few sections. None if the ideas presented and communicated in this thesis were formulated by AI.

Appendix B EDA

Appendix C FEATURES

Table 3: Topological Feature Definitions and Formulas

Feature	Type	Formula / Description
Node Degree	Node-level	$deg(v) = \lvert\{u \in V : (v,u) \in E\}\rvert$
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	$\frac{1}{\lvert N(v)\rvert} \sum_{u \in N(v)} d(v,u)$
Common Neighbours	Trajectory-level	$\lvert\Gamma(u) \cap \Gamma(v)\rvert$
Jaccard Coefficient	Trajectory-level	$\frac{\lvert\Gamma(u) \cap \Gamma(v)\rvert}{\lvert\Gamma(u) \cup \Gamma(v)\rvert}$
Preferential Attachment	Trajectory-level	$\lvert\Gamma(u)\rvert \cdot \lvert\Gamma(v)\rvert$
Adamic-Adar Index	Trajectory-level	$\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log \lvert\Gamma(w)\rvert}$

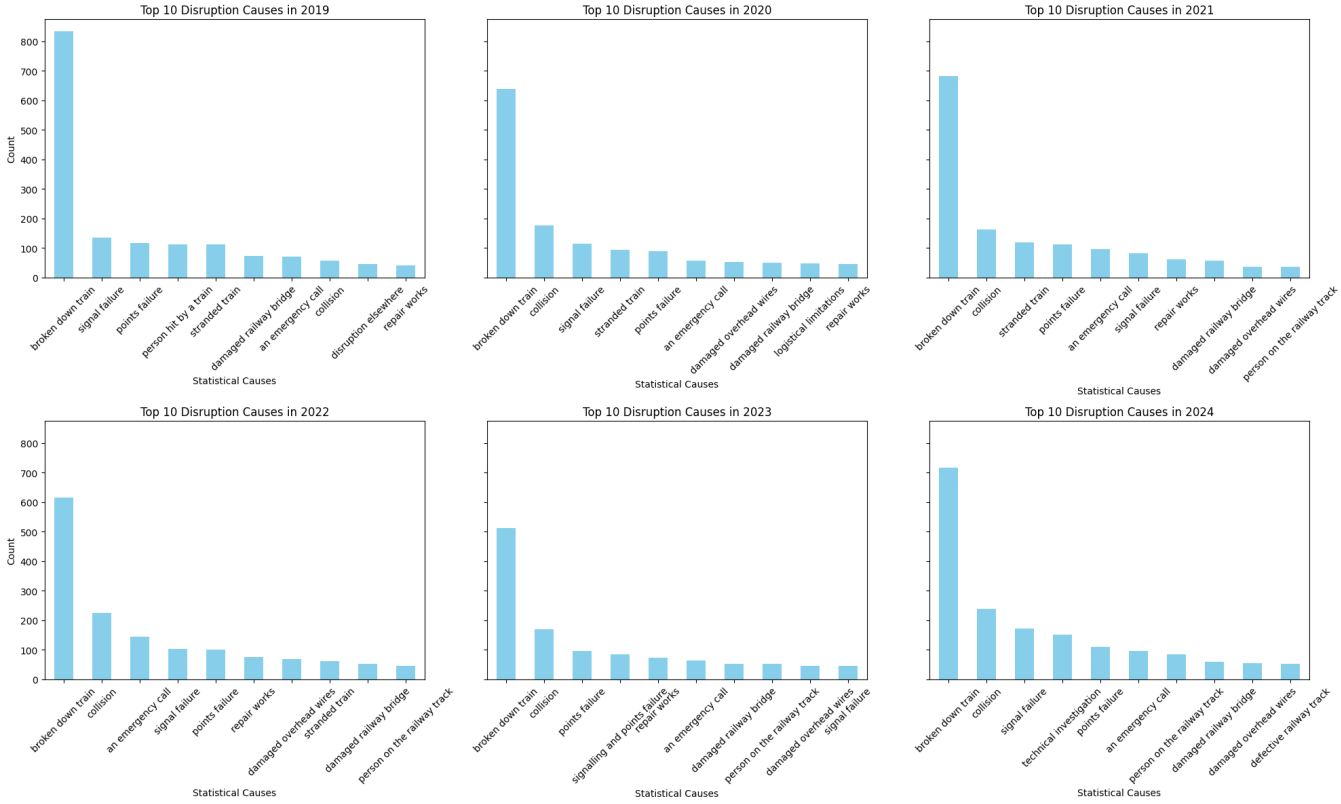


Figure 9: Most Common Causes for Disruptions by Year (Excluding 'Unspecified' causes)

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

Table 4: Results after hyperparameter tuning for each season

Table 5: Cross-Validation Performance Metrics (Mean ± Standard Deviation)

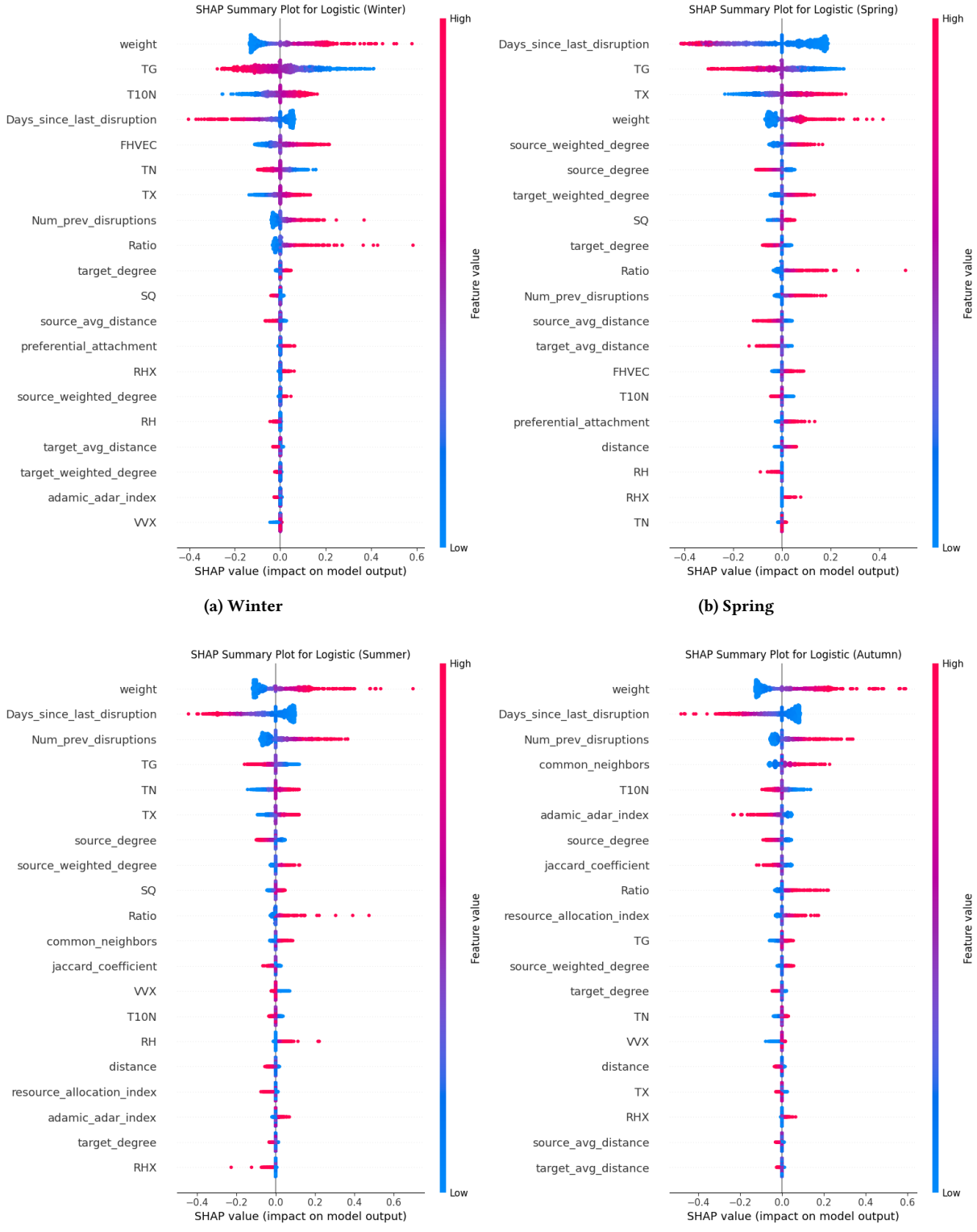
Model	Balanced Accuracy	F1 Score	AUC
Topology features	-	-	-
LogisticRegression	0.676 ± 0.028	0.382 ± 0.037	0.739 ± 0.038
RandomForestClassifier	0.585 ± 0.021	0.286 ± 0.037	0.752 ± 0.035
XGBClassifier	**0.703 ± 0.035**	0.403 ± 0.029	0.772 ± 0.043
GradientBoostingClassifier	0.577 ± 0.017	0.266 ± 0.042	0.781 ± 0.041
LSTM	0.554 ± 0.008	0.201 ± 0.025	0.783 ± 0.018
Mean of the above scores ([sum(scores) ± sum(std)]) / n	[0.597, 0.641]	[0.266, 0.334]	[0.730, 0.800]
Weather features	-	-	-
LogisticRegression	**0.511 ± 0.007**	0.222 ± 0.033	0.512 ± 0.014
RandomForestClassifier	0.501 ± 0.004	0.059 ± 0.017	0.499 ± 0.007
XGBClassifier	0.497 ± 0.007	0.177 ± 0.018	0.495 ± 0.012
GradientBoostingClassifier	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	-	-	-
LogisticRegression	**0.674 ± 0.012**	0.345 ± 0.043	0.741 ± 0.014
RandomForestClassifier	0.551 ± 0.011	0.236 ± 0.031	0.595 ± 0.018
XGBClassifier	0.667 ± 0.013	0.349 ± 0.030	0.725 ± 0.021
GradientBoostingClassifier	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	-	-	-
LogisticRegression	0.673 ± 0.027	0.380 ± 0.035	0.733 ± 0.035
RandomForestClassifier	0.560 ± 0.019	0.220 ± 0.054	0.770 ± 0.035
XGBClassifier	**0.687 ± 0.041**	0.394 ± 0.035	0.759 ± 0.047
GradientBoostingClassifier	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	-	-	-
LogisticRegression	**0.666 ± 0.013**	0.342 ± 0.041	0.739 ± 0.015
RandomForestClassifier	0.546 ± 0.014	0.183 ± 0.038	0.725 ± 0.010
XGBClassifier	0.664 ± 0.017	0.353 ± 0.026	0.725 ± 0.022
GradientBoostingClassifier	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	-	-	-
LogisticRegression	0.708 ± 0.019	0.392 ± 0.032	0.781 ± 0.017
RandomForestClassifier	0.562 ± 0.022	0.223 ± 0.064	0.761 ± 0.059
XGBClassifier	**0.712 ± 0.041**	0.419 ± 0.030	0.789 ± 0.050
GradientBoostingClassifier	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	-	-	-
LogisticRegression	0.707 ± 0.016	0.393 ± 0.031	0.780 ± 0.015
RandomForestClassifier	0.553 ± 0.019	0.194 ± 0.062	0.789 ± 0.040
XGBClassifier	0.703 ± 0.043	0.416 ± 0.036	0.783 ± 0.049
GradientBoostingClassifier	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]

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
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

Season	Selected Features for Logistic Regression
Winter	weight, TG, T10N, Days_since_last_disruption, FHVEC, TN, TX, Num_prev_disruptions, Ratio, target_degree
Spring	Days_since_last_disruption, TG, TX, weight, source_weighted_degree, source_degree, target_weighted_degree, SQ, target_degree, Ratio
Summer	weight, Days_since_last_disruption, Num_prev_disruptions, TG, TN, TX, source_degree, source_weighted_degree, SQ, Ratio
Autumn	weight, Days_since_last_disruption, Num_prev_disruptions, common_neighbors, T10N, adamic_adar_index, source_degree, jaccard_coefficient, Ratio, resource_allocation_index

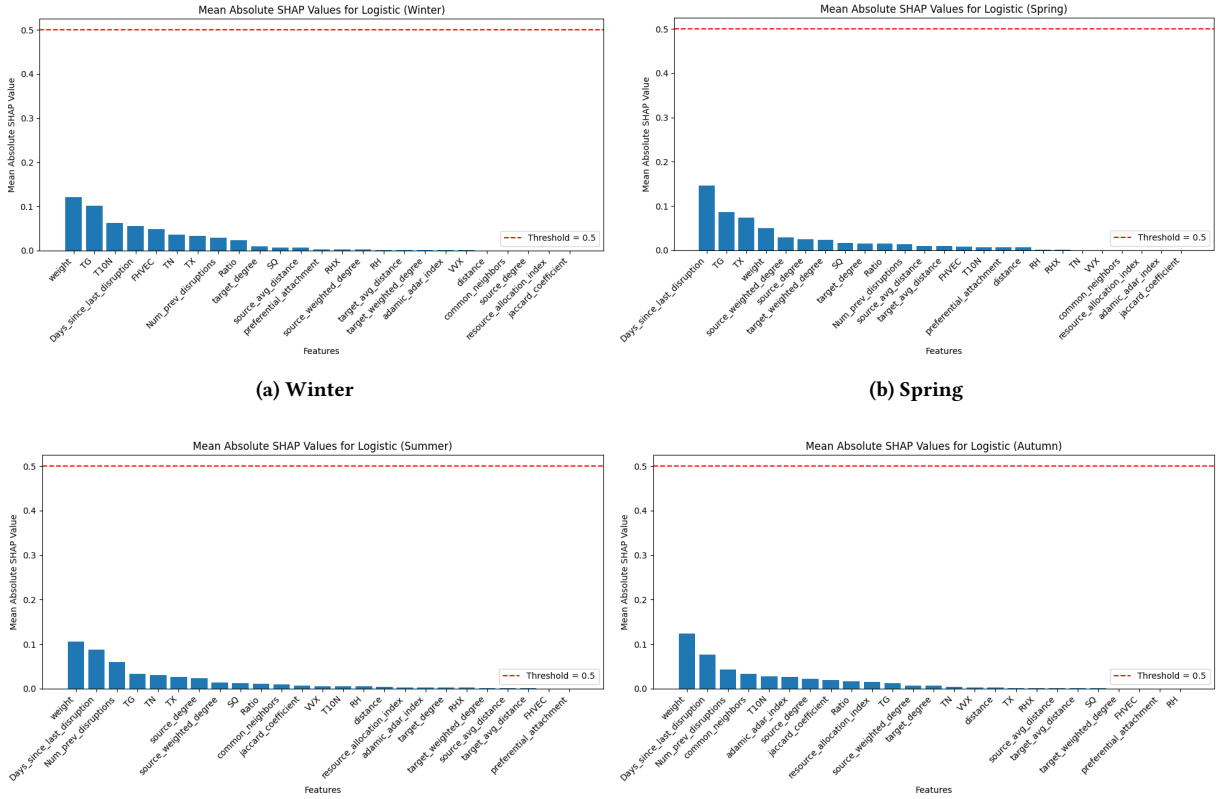
Table 7: Selected features for Logistic Regression by season



(a) Winter **(b) Spring**

(c) Summer **(d) Autumn**

Figure 10: Logistic Regression SHAP-values for All Seasons



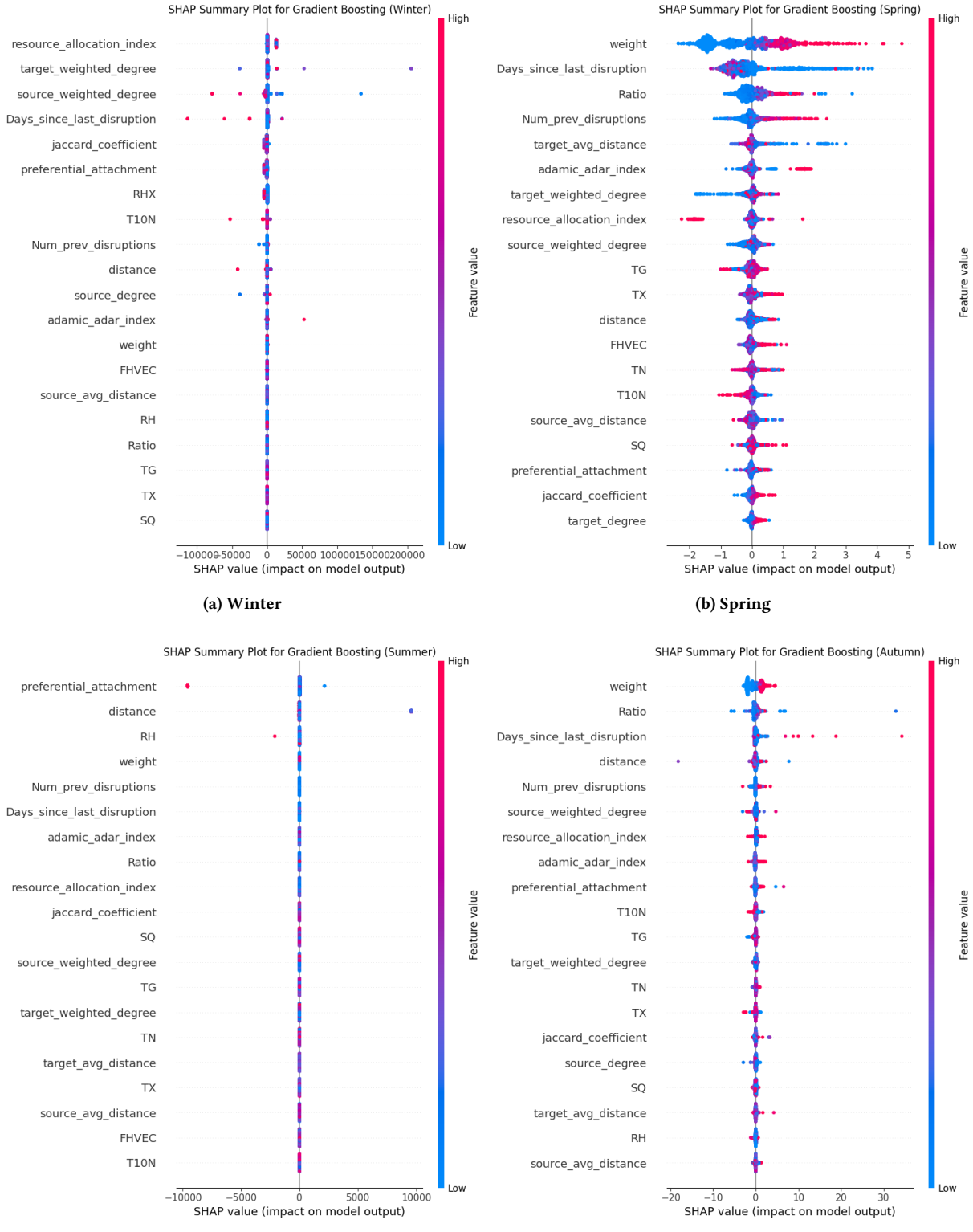
(a) Winter **(b) Spring**

(c) Summer **(d) Autumn**

Figure 11: Logistic Regression Absolute Mean SHAP-values for All Seasons

Season	Selected Features for GBDT
Winter	preferential_attachment, common_neighbors, weight, Ratio, RH, Days_since_last_disruption, RHX, source_degree, source_avg_distance, target_degree, TN, Num_prev_disruptions, FHVEC
Spring	weight, Days_since_last_disruption, Ratio, Num_prev_disruptions, TX, target_degree, source_weighted_degree, adamic_adar_index, distance, TG
Summer	weight, Num_prev_disruptions, adamic_adar_index, Ratio, target_degree, RH, Days_since_last_disruption, TX, preferential_attachment
Autumn	weight, distance, Ratio, common_neighbors, target_weighted_degree, preferential_attachment, adamic_adar_index, VVX, source_avg_distance, source_degree

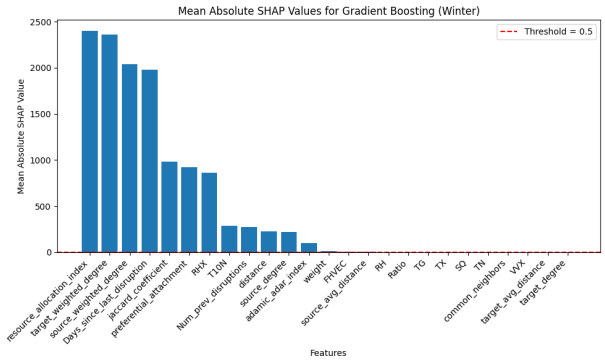
Table 8: Selected features for GBDT by season



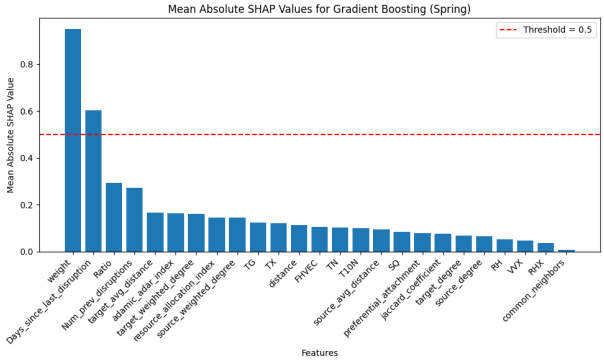
(a) Winter **(b) Spring**

(c) Summer **(d) Autumn**

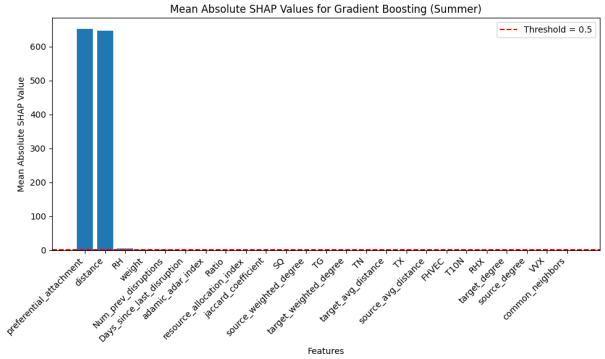
Figure 12: Gradient Boosting SHAP-values for All Seasons



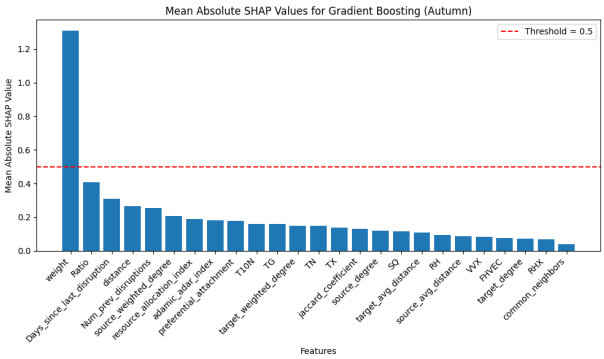
(a) Winter



(b) Spring

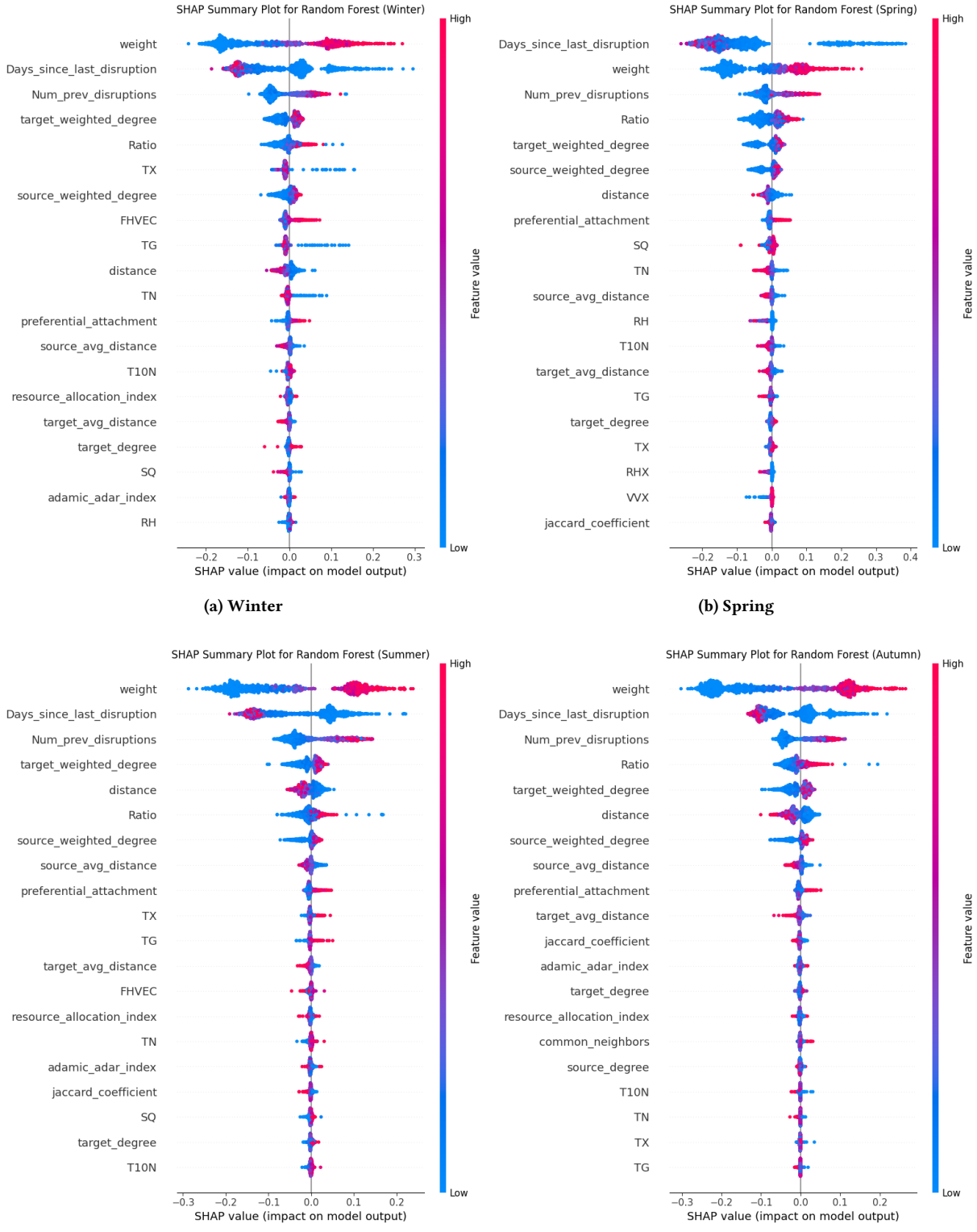


(c) Summer



(d) Autumn

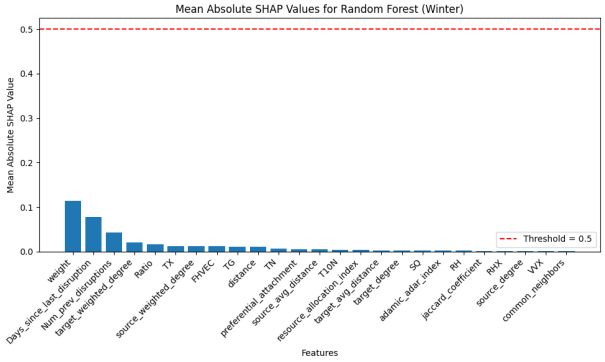
Figure 13: Gradient Boosting Absolute Mean SHAP-values for All Seasons



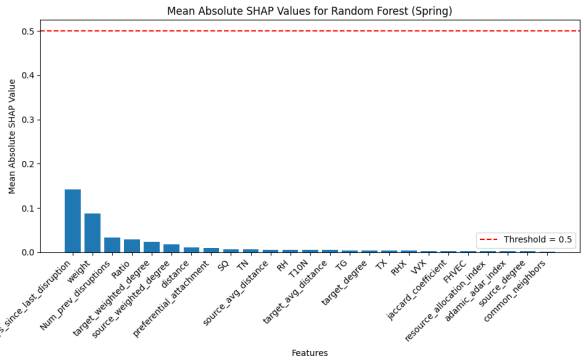
(a) Winter **(b) Spring**

(c) Summer **(d) Autumn**

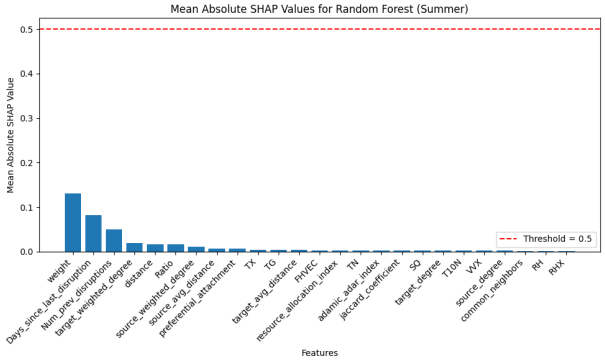
Figure 14: Random Forest SHAP-values for All Seasons



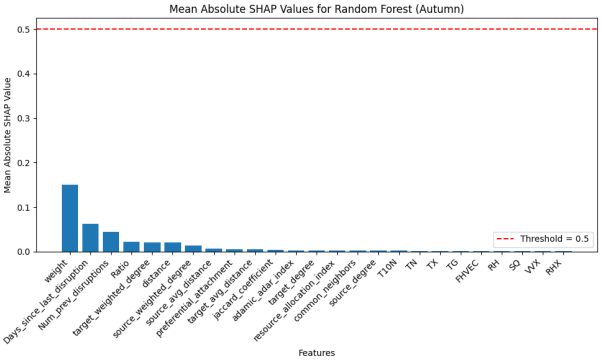
(a) Winter



(b) Spring



(c) Summer



(d) Autumn

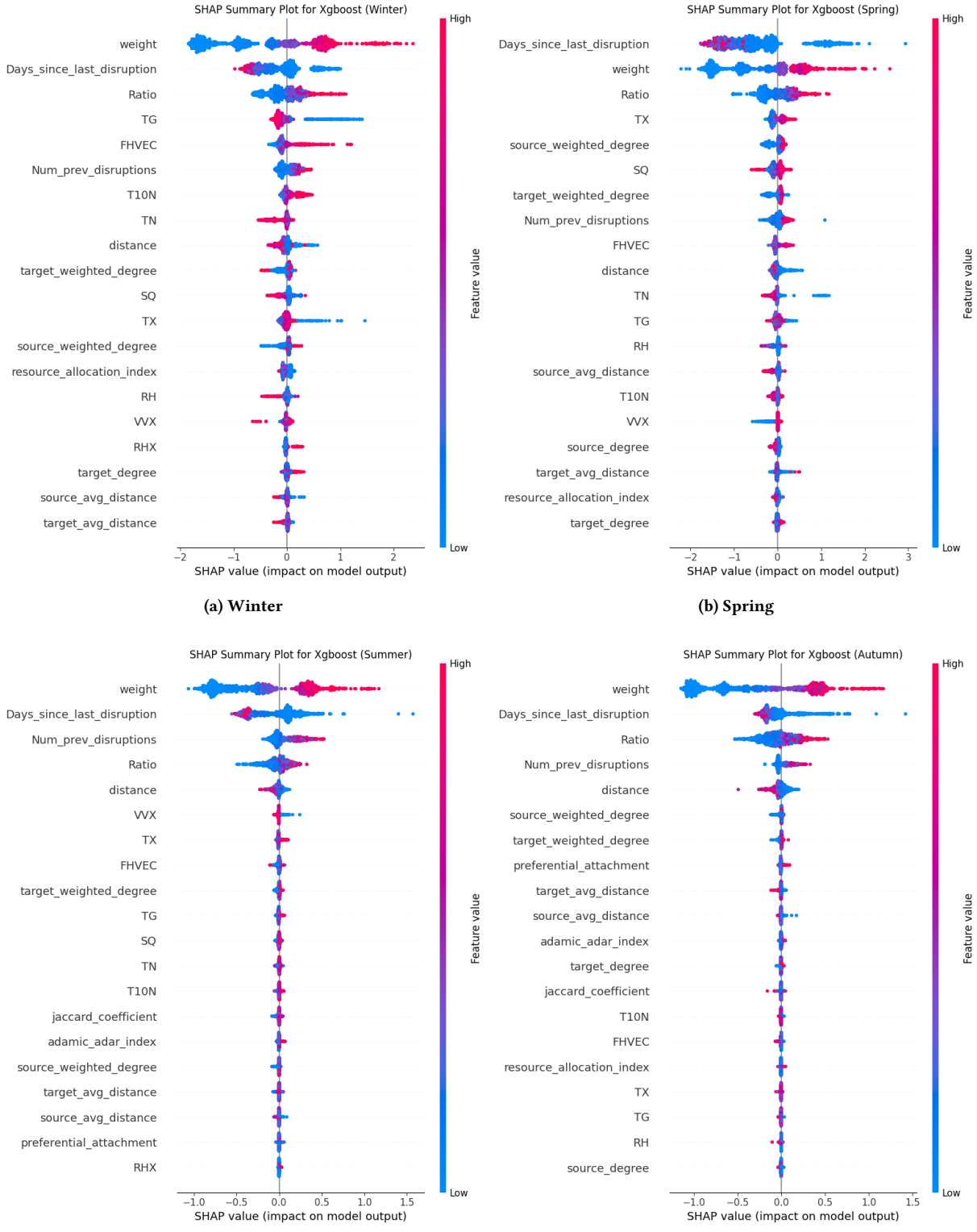
Figure 15: Random Forest Absolute Mean SHAP-values for All Seasons

Season	Selected Features for Random Forest
Winter	weight, Days_since_last_disruption, Num_prev_disruptions, target_weighted_degree, Ratio, TX, source_weighted_degree, FHVEC, TG, distance
Spring	Days_since_last_disruption, weight, Num_prev_disruptions, Ratio, target_weighted_degree, source_weighted_degree, distance, preferential_attachment, SQ, TN
Summer	weight, Days_since_last_disruption, Num_prev_disruptions, target_weighted_degree, distance, Ratio, source_weighted_degree, source_avg_distance, preferential_attachment, TX
Autumn	weight, Days_since_last_disruption, Num_prev_disruptions, Ratio, target_weighted_degree, distance, source_weighted_degree, source_avg_distance, preferential_attachment, target_avg_distance

Table 9: Selected features for Random Forest by season

Season	Selected Features for XGBoost
Winter	weight, Ratio, Days_since_last_disruption, Num_prev_disruptions, TG, distance, source_weighted_degree, source_degree, target_avg_distance, target_degree, TN
Spring	weight, Days_since_last_disruption, Ratio, TX, Num_prev_disruptions, distance, source_weighted_degree, target_weighted_degree, source_avg_distance, TG
Summer	weight, Days_since_last_disruption, Ratio, Num_prev_disruptions, distance, target_weighted_degree, TX, adamic_adar_index, RHX, preferential_attachment
Autumn	weight, Days_since_last_disruption, Ratio, Num_prev_disruptions, distance, target_degree, source_weighted_degree, source_avg_distance, preferential_attachment

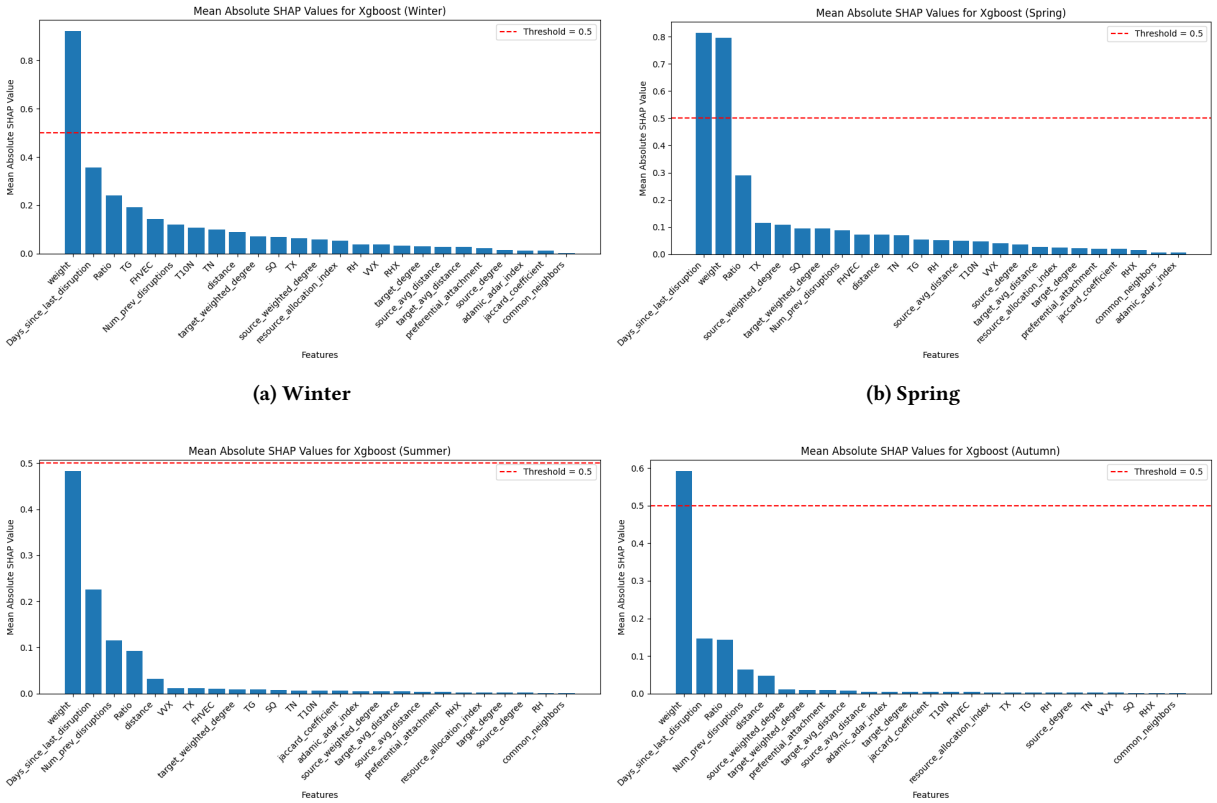
Table 10: Selected features for XGBoost by season



(a) Winter **(b) Spring**

(c) Summer **(d) Autumn**

Figure 16: XGBoost SHAP-values for All Seasons



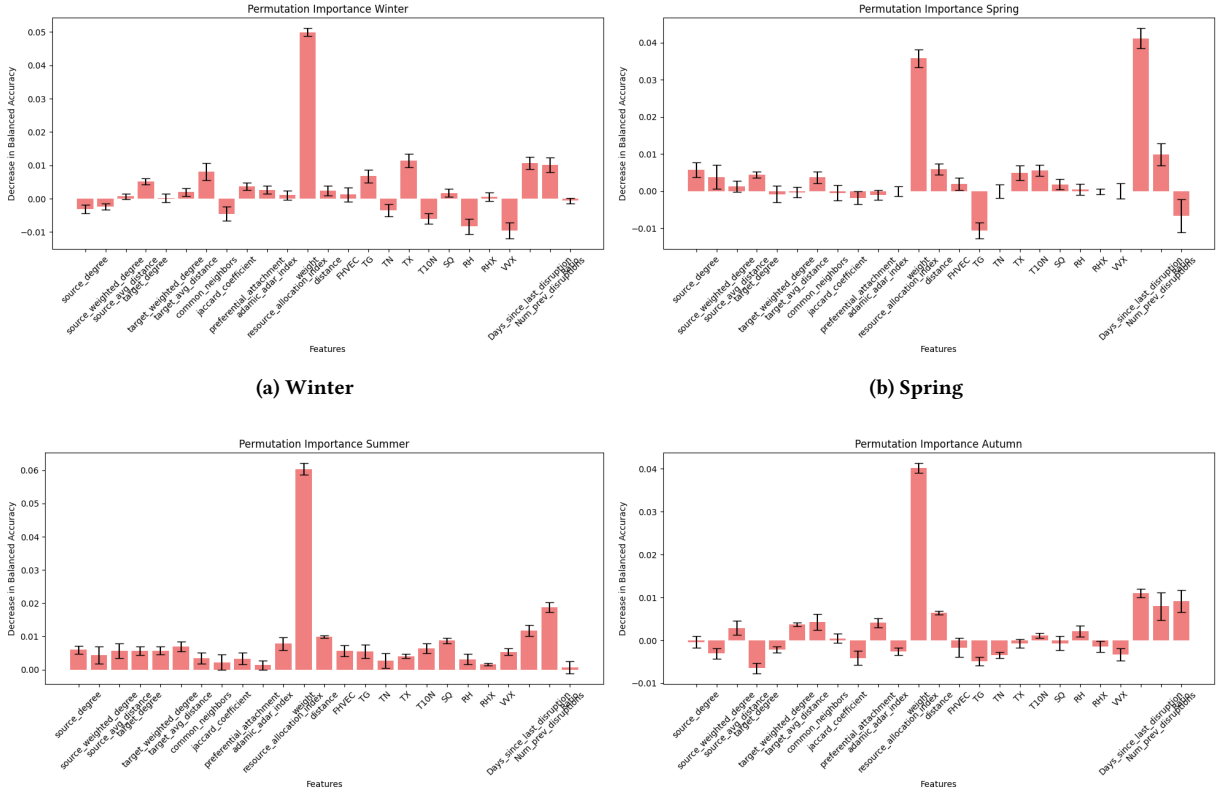
(a) Winter **(b) Spring**

(c) Summer **(d) Autumn**

Figure 17: XGBoost Absolute Mean SHAP-values for All Seasons

Season	Selected Features for LSTM
Winter	weight, TX, Days_since_last_disruption, Num_prev_disruptions, common_neighbors, TG, target_degree, preferential_attachment, adamic_adar_index, distance
Spring	Days_since_last_disruption, weight, Num_prev_disruptions, distance, source_degree, T10N, TX, target_degree, source_weighted_degree, common_neighbors
Summer	weight, Num_prev_disruptions, Days_since_last_disruption, distance, SQ, resource_allocation_index, target_avg_distance, T10N, source_degree, target_degree
Autumn	weight, Days_since_last_disruption, Ratio, Num_prev_disruptions, distance, common_neighbors, adamic_adar_index, target_avg_distance, source_avg_distance, RH

Table 11: Selected features for LSTM by season



(a) Winter **(b) Spring**

(c) Summer **(d) Autumn**

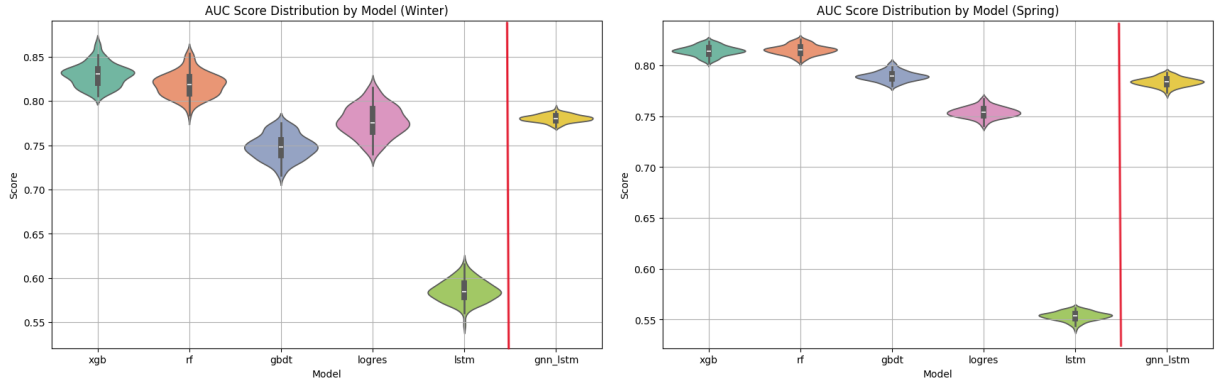
Figure 18: LSTM Permutation Importance Values for All Seasons

Table 12: Bootstrapped Model Performance Metrics Winter Dataset (Mean (Confidence Interval))

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)

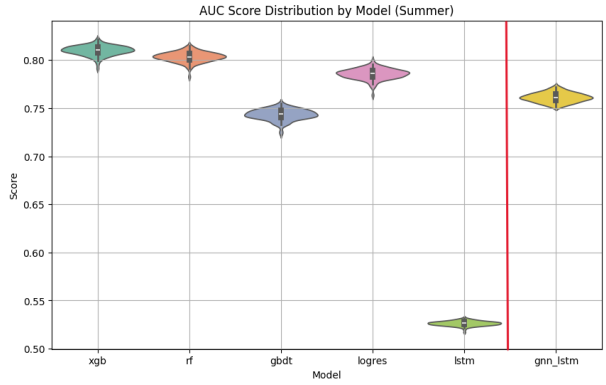
Table 13: Bootstrapped Model Performance Metrics – Spring Dataset (Mean (Confidence Interval))

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)

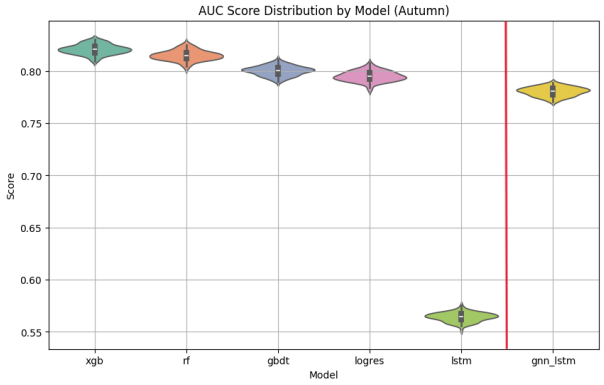


(a) Winter

(b) Spring



(c) Summer



(d) Autumn

Figure 19: Bootstrapped AUC Scores across Seasons

Table 14: Bootstrapped Model Performance Metrics – Summer Dataset (Mean (Confidence Interval))

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 15: Bootstrapped Model Performance Metrics – Autumn Dataset (Mean (Confidence Interval))

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)

Table 16: Independent t-tests on F1 Score Across Seasons

(a) Winter

Model Comparison	t-value	p-value
XGBoost vs RF	-8.21	1.00
XGBoost vs GBDT	15.86	0.00
XGBoost vs LogReg	-7.28	1.00
XGBoost vs LSTM	8.59	0.00
XGBoost vs DGNN	-42.18	1.00
RF vs GBDT	18.40	0.00
RF vs LogReg	0.75	0.23
RF vs LSTM	13.30	0.00
RF vs DGNN	-12.62	1.00
GBDT vs LogReg	-17.56	1.00
GBDT vs LSTM	-4.48	1.00
GBDT vs DGNN	-49.01	1.00
LogReg vs LSTM	12.51	0.00
LogReg vs DGNN	-13.67	1.00
LSTM vs DGNN	-34.70	1.00

(b) Spring

Model Comparison	t-value	p-value
XGBoost vs RF	69.69	0.00
XGBoost vs GBDT	23.99	0.00
XGBoost vs LogReg	392.08	0.00
XGBoost vs LSTM	125.32	0.00
XGBoost vs DGNN	61.91	0.00
RF vs GBDT	-44.25	1.00
RF vs LogReg	196.64	0.00
RF vs LSTM	50.60	0.00
RF vs DGNN	-31.67	1.00
GBDT vs LogReg	291.98	0.00
GBDT vs LSTM	96.90	0.00
GBDT vs DGNN	23.59	0.00
LogReg vs LSTM	-119.94	1.00
LogReg vs DGNN	-369.43	1.00
LSTM vs DGNN	-92.20	1.00

(c) Summer

Model Comparison	t-value	p-value
XGBoost vs RF	95.13	0.00
XGBoost vs GBDT	79.46	0.00
XGBoost vs LogReg	58.56	0.00
XGBoost vs LSTM	251.81	0.00
XGBoost vs DGNN	37.82	0.00
RF vs GBDT	-21.93	1.00
RF vs LogReg	-35.72	1.00
RF vs LSTM	103.93	0.00
RF vs DGNN	-68.19	1.00
GBDT vs LogReg	-15.68	1.00
GBDT vs LSTM	141.25	0.00
GBDT vs DGNN	-49.05	1.00
LogReg vs LSTM	152.82	0.00
LogReg vs DGNN	-29.08	1.00
LSTM vs DGNN	-221.55	1.00

(d) Autumn

Model Comparison	t-value	p-value
XGBoost vs RF	93.64	0.00
XGBoost vs GBDT	80.01	0.00
XGBoost vs LogReg	46.24	0.00
XGBoost vs LSTM	113.63	0.00
XGBoost vs DGNN	-3.90	1.00
RF vs GBDT	-16.29	1.00
RF vs LogReg	-44.06	1.00
RF vs LSTM	19.79	0.00
RF vs DGNN	-94.83	1.00
GBDT vs LogReg	-29.10	1.00
GBDT vs LSTM	36.45	0.00
GBDT vs DGNN	-81.44	1.00
LogReg vs LSTM	63.49	0.00
LogReg vs DGNN	-48.36	1.00
LSTM vs DGNN	-114.51	1.00