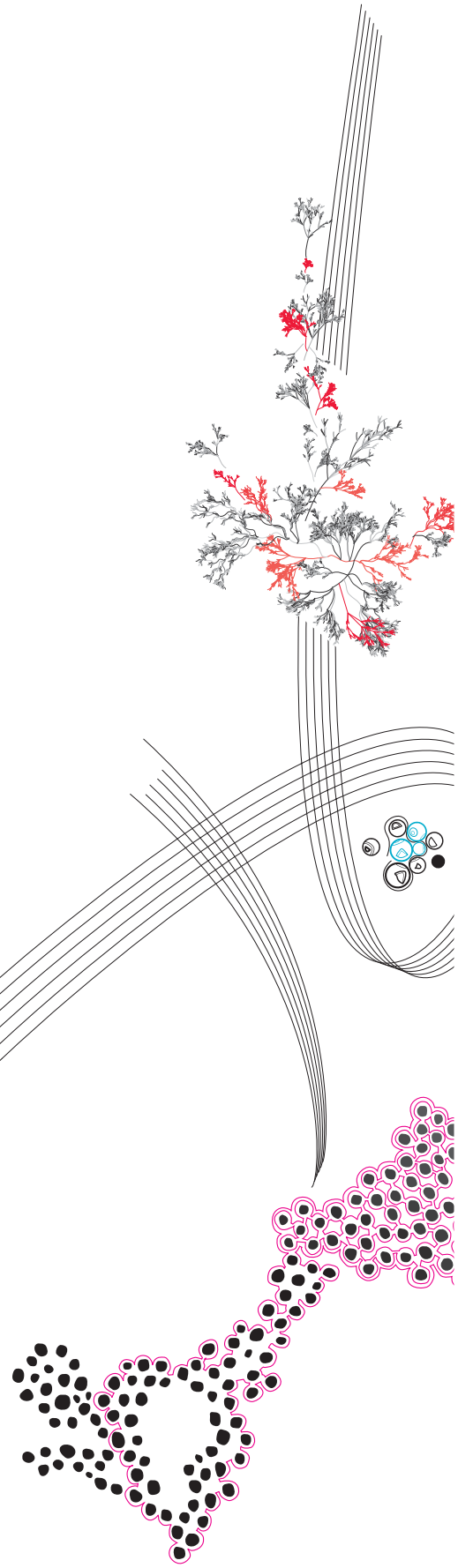

Master Thesis Applied Mathematics

Examining the potential of Graph Neural Networks on road network data for traffic crash prediction.

Wouter Jorick Doedens

Supervisors: dr. C. Stegehuis and dr.ir. M.B. Ulak

February, 2025

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Contents

Abstract

As the number of cyclists in cities increases, so does the number of road accidents. This paper aims to identify crash risk factors in road features contributing to traffic accidents in the four largest Dutch cities. The study proposes training Graph Neural Networks on road network data to identify these factors for traffic crash prediction. Two machine learning (ML) models were trained to predict traffic accidents: a Multi-Layer Perceptron (MLP) and a Graph Convolution Network (GCN). Additionally, to address the class imbalance in traffic accident data, Class Balanced loss is introduced to give greater weight to minority classes. GNNExplainer, an Explainable Artificial Intelligence (XAI) method combined with partial dependence plots, are used in this study to determine which road factors are important for the ML models. The results show that it is harder for Graph Convolution Networks to classify minority classes correctly. Moreover, the results confirm that the chances of crashes increase as the number of cyclists and motorised vehicles on the road increases. The centrality of the road and its proximity to commercial facilities also increase the likelihood of crashes. Furthermore, increasing the amount of traffic lights decreases crash probability. Lastly, the findings show that separating bicycle lanes on 50 km/h roads and including bicycle lanes on 30 km/h roads increase traffic safety.

1 Introduction

Due to its cycling infrastructure, the Netherlands is widely regarded as one of the safest countries in the world for bicycle users. However, due to the sheer number of cyclists per day [1], the Netherlands has one of the highest cycling fatality rates per capita in Europe [2]. As a result, researchers and local governments are continuously working to improve traffic safety in the Netherlands. The field of enhancing traffic safety by examining road factors has been studied for a long time. In statistical road safety modelling (SRSM), statistical models are fitted to data about past accidents and road factors [3]. Various factors related to road segments, such as speed limits and infrastructure, have been shown by statistical models to influence the likelihood of traffic crashes happening [4][5]. Others examined the applicability of Machine Learning models in crash prediction [6].
However, these analyses often overlook the impact of how different roads are interconnected in a network, where road segments with a high crash likelihood might increase the possibility of crashes in nearby road segments. Since road networks act like graphs, where each road segment (node) has its own road information factors (features) and connections (edges) between the roads, the question arises if graph-based models, such as Graph Neural Networks (GNNs), can identify crash risk factors related to the road network in more detail. GNNs are great for modelling spatial relationships as well as temporal relationships, and some researchers have implemented them into the field of traffic safety [7][8].
Nevertheless, these models still focus on optimising model predictions, not on finding what factors influence the predictions the most. To this end, Explainable Artificial Intelligence (XAI) methods can help explain why models have made certain decisions. Existing studies in traffic safety predictions already utilise XAI methods to explain models [9]. However, this has not been done for Graph Neural Networks.
Therefore, this paper aims to answer the question: *How can Graph Neural Networks help identify important road features that influence traffic crashes?*

In Section 2, we first examine Graph Neural Networks in more depth and introduce the most common GNNs. Then, we investigate recent advancements in the traffic safety field and GNNs used to predict traffic crashes. Thereafter, we analyse the class imbalance and

Explanation methods and how they can help us identify important crash risk factors. In Section 3, we introduce the data from Uijtdewilligen et al. [10] used in this paper. We examine the different variables in the data and where they come from. Section 4 explains the methodology used in this paper. It explains which methods were used and why. In Section 5, we examine the results of the trained models. First, we take a look at the results plotted in histograms. We try to explain the model using Explainability methods. In Section 6, we discuss the results and what can be improved for further research. Section 7 gives an overall conclusion for this study.

2 Background

In this Section, we will review the existing literature on Graph Neural Networks and the field of Traffic Safety. First, we will go into the mathematical definition of graphs and the basic structure of Graph Neural Networks. This will serve as a basis for discussing the different types of Graph Neural Networks. We will explore recent literature on Traffic safety and Graph Neural Networks used in traffic crash prediction situations. We will also describe techniques that handle the class imbalance of graphs during model training. Lastly, we introduce the different Explainability methods used in Graph Neural Networks.

2.1 Overview of Graph Neural Networks

2.1.1 Graph Definition

A graph G contains a non-empty set of nodes V and a set of edges E. In $G = (V, E)$, any node $i \in V$ connected to a node $j \in V$ is connected by an edge $e_{ij} \in E$. The adjacency matrix $A \in \mathbb{R}^{n \times n}$, where $n = |V|$, stores information about which connections exist in the graph. In Graph Neural Networks, every node i has a feature vector $x_i \in \mathbb{R}^d$, which contains features (variables) about this node. Here, d indicates the number of variables. $X \in \mathbb{R}^{n \times d}$ is the node feature matrix. Edges can have features $x^e_{ij} \in \mathbb{R}^{d'}$ as well. We denote $X^e \in \mathbb{R}^{m \times d'}$ to be the edge feature matrix. The neighbourhood N_i of a node i indicates which nodes surround i and is indicated by $N_i = \{j | e_{ij} \in E\}$.
The edges in a graph can be directed, meaning node i points to node j. Or undirected, which is the same as saying nodes i and j point to one another. Furthermore, graphs can be homogeneous and heterogeneous. Homogeneous means that nodes and edges have the same types, and heterogeneous graphs have nodes and edges of other types. There is also a difference between static and dynamic graphs. In dynamic graphs, the input features, or the graph's topology, can vary over time, whereas in static graphs, these remain the same.

2.1.2 Graph Neural Networks

The input of a Graph Neural Network is always a graph, G, consisting of nodes and edges with node- and/or edge features. Graph Neural Networks have three main applications: node-level, edge-level and graph-level. In node-level applications, the GNN aims to predict information about the nodes in the graph. This can be node classification, in which the goal is to predict the classes of different nodes, or node regression, in which the GNN predicts continuous variables for the nodes in the graph; or node clustering, in which the GNN partitions nodes in different groups based on how similar the nodes are. In edge-level applications, the GNN makes predictions about the graph's edges. This could be edge classification, in which the GNN predicts the class of the edges, or link prediction, in which the GNN predicts if an edge exists between two nodes. In graph-level applications, the

GNN makes predictions about the entire graph. Using a GNN, one might, for example, want to predict the toxicity of a molecule by training the GNN on its graph structure [11].

Graph Neural Networks get trained via a process called Message Passing [12][13]. The neighbours of a node i engage in Message Passing, where information from the feature vectors of the neighbouring nodes j is aggregated using a permutation invariant function like sum, mean or max. This creates the message:

$$m_i^t = Agg(\{C_j \cdot h_j^{(t-1)} : j \in N_i\}). \tag{1}$$

Here $h_j^{(t-1)} \in \mathbb{R}^F$ is the feature vector of node j at the $(t-1)$'th (where $t = 0, ..., T$) layer, or time step. The matrix $C_j \in \mathbb{R}^{F' \times F} \in$ is a trainable weight matrix. Note that in the first layer $h_j^0 = x_j$, the feature vector of node j.
Thereafter, this information is combined with the information of the last iteration of the node itself as

$$h_i^t = \sigma(\text{Combine}(B_i \cdot h_i^{(t-1)}, m_i)). \tag{2}$$

Here $B_i \in \mathbb{R}^{F' \times F}$ is a trainable weight matrix and $h_i^{(t-1)} \in \mathbb{R}^F$ is the hidden state vector of node i at layer $(t-1)$. The combined function can be as simple as a sum but could also concatenate the information. σ is a non-linear activation function. The output of node i equals the hidden state vector at the last layer: $o_i = h_i^T$.

2.1.3 Spatial Graph Neural Networks

Spatial methods are a class of methods that perform convolution operations directly on the graph, leveraging the topology of the graph. A convolution operation defines a method for merging two functions or sets of information to generate a new function that represents their combined effect. Equations (1) and (2) represent the basic convolution operations used in spatial methods that gather and update neighbouring nodes' information onto a new feature space. The challenge for spatial approaches is defining the convolution operation with varying neighbour sizes whilst ensuring that the methods maintain local invariance [12].
In [14], Kipf and Welling introduce the Graph Convolution Network (GCN). GCNs aggregate the neighbouring node features via matrix multiplication. GCN normalises the feature information from neighbouring nodes based on its node degrees. GraphSAGE, introduced by Hamilton, Ying and Leskovec in [15], samples information from a fixed number of neighbours. The authors of this paper suggest three types of aggregation functions for Agg in Equation (1). Mean aggregation, LSTM aggregation and pooling aggregation. GraphSAGE with a mean aggregation can be regarded as an inductive version of GCN [16].
In [17], Veličković et al. introduce the Graph Attention Network (GAT). Each edge in the network gets its own attention coefficient α_{ij}, which determines how much feature information from neighbouring nodes is aggregated. Gao et al. [18] propose a Large-Scale Learnable Graph Convolution Network. First, they introduce the learnable graph convolution layer (LGCL), which enables them to make regular convolutions on a graph. The LGCN uses the LCGL layer in the network for node classification. First, a graph embedding layer maps high-dimensional feature vectors to a low-dimensional representation. After the graph embedding layer, multiple LGCL layers are stacked according to the graph's complexity. Skip connections are used to concatenate the input and output of LGCL layers. Eventually, a fully connected layer and the softmax function are used to create the final prediction and node classification.

2.1.4 Spatio-Temporal Graph Neural Networks

A spatio-temporal graph is a graph where the node attributes change over time [19]. Spatio-temporal models analyse and model data with both spatial and temporal dimensions. In [20], Li at el. Introduce Gated Graph Neural Networks. It uses Gated Recurrent Units (GRU) in the propagation step. A node's features are updated via aggregation of its previous features and the surrounding features. This form of message passing handles the spatial domain. It uses a recurrence method such as the Gated Recurrence Unit (GRU) to handle the temporal dynamics. A Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network (RNN) that uses gates to control the flow of information.
The Spatial-Temporal Graph Convolution Network proposed by Yan et al. [21] first performs a convolution in the spatial domain, similar to the Graph Convolution Network. After that, a 1-dimensional convolution in the temporal domain is performed. These two convolutions, in sequence, form the ST-GCN unit. In [22], Seo et al. explain the Graph Convolutional Recurrent Network (GCRN). GCRN uses either Convolution Neural Networks (CNN's) or Graph Convolution networks to model spatial decencies. Combining these with Recurrent Neural Networks (RNN's) like GRU or LSTM allows it to model both spatial and temporal dependencies. Gated Attention Networks (GaAn) introduced by Zhang et al. [23] use an attention mechanism to handle message passing in the spatial domain. The authors combine GaAN with a GRU model to create the Graph Gated Recurrent Unit (GGRU), which directly applies to spatial-temporal forecasting problems.

2.2 Traffic Safety

Over the years, researchers have tried to develop methods that can accurately predict traffic safety factors. One of the first, Hauer [24], uses statistical road safety modelling (SRSM) to fit statistical models on historical data to estimate the road design elements of safety. Mannering and Bhat [25] took a broader approach and reviewed methodological applications in highway-crash data. Their work explored a range of techniques from generalised linear models (GLM), such as linear regression, to machine learning (ML) models, such as neural networks, Bayesian neural networks, and support vector machines (SVM). In [26], Schlögl et al. go into more depth by using various GLM and ML models such as logistic- and binary quantile regression and Random Forests (RF), XGboost and SVM to derive the determining factors associated with road traffic accident occurrence of the whole highway network of Austria over four consecutive years. Subsequently, Morris and Yang [9] examined different methods for analysing freeway crashes. By using Machine Learning models (CatBoost, XGBoost, and Random Forests) and one classic statistical model (Nested Logit) they tried to predict the types of freeway crashes.
To determine what road factors have the highest effect on traffic fatalities amongst cyclists in Delhi, Agrawal et al. [27] use a logistic regression model. Furthermore, in [28], Ahmed et al. combine an ensemble of ML models that predict road accident severity in New Zealand with Shapley value analysis. Shapely value analysis is an Explainable Machine Learning (XML) technique that evaluates the importance of road accident contributing factors in a Machine Learning model used for prediction. Lastly, in [10], Uijtdewillingen et al. examine crash risk factors for cyclists using logistic regression and binomial regression on the four biggest cities in the Netherlands. However, none of these methods considers the graph structure of the road network.

2.2.1 Graph Neural Networks for traffic crash prediction

Building on this gap, researchers use graph neural networks to predict traffic crashes. In [29], Zhao et al. investigate GCNs potential to establish complex spatial relations between crash counts and external variables. The Attention-based Spatio-Temporal Graph Convolutional Network (ASTGCN), developed by Liu et al. [8], uses graph convolutions in the spatio-temporal domain to aggregate feature information. An attention mechanism determines which features contribute the most to predicting crash risk. Zhang et al. [30] propose GraphCast, a multi-modal graph neural network framework to forecast traffic risks. A GNN framework and an attention mechanism are added to make predictions. Furthermore, Yu et al. [7] developed a Deep Spatio-Temporal Graph Convolution Network (DSTGCN) to predict traffic accidents. DSTGCN predicts traffic crashes using blocks of convolutions in the spatial and temporal domain. In [31], Ye et al. use traffic accidents to predict fluctuations in traffic flows. A GAT model is used for the spatial dependencies, and a bidirectional Long Term Short Memory (Bi-LSTM) model is used for the temporal dependencies. The Multi-Attention Dynamic Graph Convolution Network (MADGCN), proposed by Wu et al. [32], uses attention mechanisms in both the spatial and the temporal domains. Whilst most of these works show that GNN's are a promising tool for predicting traffic crashes, none explain which factors were most important in making these predictions.

2.3 Class-Imbalanced Learning on Graphs (CILG)

This Section looks at different methods to handle class imbalance in graph-based data for better predictions. Class imbalance refers to a situation in machine learning where a model is tasked with predicting the class or category of some input data, and the distribution of these classes is not uniform. For example, if 99% of the time, no crashes ever happen, a model that predicts that no crashes will ever happen will have an accuracy of 99%. To address this, Ma et al. [33] introduce Class-Imbalanced Learning on Graphs (CILG), in which different methods to handle the class imbalance in graph-based data are compared. There are two main categories in CILG: data-level methods and algorithmic-level methods.

2.3.1 Data-level Methods

Data-level methods focus on the oversampling of minority classes and undersampling of majority classes. These methods focus mainly on creating synthetic nodes. In [34], Liu et al. develop GATSMOTE, which is a combination of Synthetic Minority Over-sampling Technique (SMOTE) and Graph Attention Networks (GAT) in which SMOTE first develops synthetic nodes and edges and then the attention mechanism decides which new nodes and edges to keep. GraphENS, introduced by Park, Song and Yang [35], adds synthetic nodes by examining the neighbourhood characteristics of existing nodes. Wu et al. [36] also use the interpolation of node features to create new synthetic nodes in their model GraphMixup. In [37], Qu et al. leverage General Adversarial Networks (GAN's) in their model ImGAGN. ImGAGN uses a generator model to create new nodes and a discriminator model to evaluate if nodes are real or synthetic. The goal is to generate synthetic nodes that the discriminator cannot distinguish.

2.3.2 Algorithmic-level methods

Algorithmic-level methods try to adjust learning algorithms to handle the class imbalance. Standard algorithmic level methods alter loss functions such that the model gets penalised more heavily for wrong classifications of minority class nodes. In [38], Lin et al. propose Focal Loss to address the class imbalance in their data. Focal Loss reshapes the standard cross entropy loss by adding a modulating factor that down-weights the loss of well-classified samples. For samples that are not easily classified, the loss remains significant. Class Balanced loss, introduced by Cui et al. [39], handles the class imbalance by creating a weighting factor using the effective number of samples n_y for each class. This makes the loss more sensitive to minority-class data and less sensitive to majority-class data. The benefit of Class Balanced loss is that its weighting factor can be combined with any existing loss. Class Rectification Loss, from Dong et al. [40], widens the decision boundary between minority and majority classes by modifying the loss function, thus effectively reducing misclassification. Other methods, like ReNode [41] and TAM [42], incorporate graph topology in their loss function designs.

2.4 Explainability for Graph Neural Networks

Explainers are techniques designed to interpret and explain a model's decision, making the model more understandable. In Graph Neural Networks, they help identify the nodes, node features, edges, and subgraphs that contribute most to a specific prediction. In [43], Amara et al. develop the GraphFramEx framework for the explainability of graph-based models. Some explainability methods like Saliency [44], Integrated Gradients [45] and Grad-CAM [46] can also be applied on non-graph-based models. These methods are gradient-based; they compute the gradients of the model's output based on the input features. Let ϕ indicate the model used for prediction and let $X \in \mathbb{R}^{n \times d}$ is the input feature matrix of Section 2.1.1, then these methods return a matrix of the same size $X^F \in \mathbb{R}^{n \times d}$ which indicates how much an input feature contributes to the model's output.
Other explanation models are graph-based. Graph-based models take a computation graph $G_C \subseteq G$, where G is the graph definition of Section 2.1.1 and return a subgraph $G_S \subseteq G_C$ containing the most critical nodes and edges to make predictions. GNNExplainer [47], developed by Ying et al., is a graph-based explainer that also returns a subset of the essential features of the subgraph $X_S^F \subseteq X_S$, where $X_S = \{x_j | v_j \in G_S\}$, is the set of all node features of the subgraph G_S. Methods like PGExplainer [48] and SubgraphX [49] focus mainly on explaining which subgraph, G_S, is essential for predictions. These methods focus on removing edges. Similarly, GraphMask [50] identifies which edges are critical for making predictions and removes unnecessary edges. On the other hand, PGM-Explainer [51] perturbs input node features and uses Probabilistic Graphical Models (Bayesian Networks) to find the dependencies between the node features and the predictions.

3 The Data

In this thesis, we will work with the data from Uijtdewilligen et al. [10], which provides the road networks and features of the Netherlands' four biggest cities. We will now describe the data in more detail.

3.1 The Study Area

The data was collected from the four largest cities in the Netherlands between 2015 and 2019. These cities are Amsterdam, Rotterdam, Utrecht and The Hague. During this period, Amsterdam had 883,000 inhabitants, Utrecht had 362,000 inhabitants, Rotterdam had 655,000 inhabitants, and The Hague had 553,000 inhabitants (CBS)[10]. In this study, data is limited and does not cover all roads in each city. The number of roads in each city might differ from reality. This study has data on 2182 roads in Amsterdam, 4269 roads in Rotterdam, 1928 roads in Utrecht and 9858 roads in The Hague.

3.2 Features

The road features are divided into four categories: crash data, bicycle and motorised vehicle exposure, cycling infrastructure, network structure, popular destinations and the municipality indication.

3.2.1 Crash data and temporal features

As in [10], the crash in this study comes from the Database of Registered Crashes in the Netherlands (BRON). Crashes are divided into two categories: fatal crashes (*Crash_fatal*) and injury (light or severe) crashes (*Crash_injury*) that involve at least one cyclist. The temporal resolution of the data is hourly for seven weekdays. So, in total, there are 168 temporal slots per road segment. The data was collected from 2015 until 2019. The data has no information about the specific date a crash happened. So, the data does not show a distinction if two crashes occurred on the same road at 12:00 on a Wednesday but on different dates. However, the crash frequency on the road (*Crash_freq*) will increase to 2. Since crashes are rare, a variable *Crash_binary* was added, which is 1 if a crash occurred on the road at all and 0 otherwise. Table 1 shows the distribution of crashes based on *Crash_freq*. The time features in the data are indicated by *Day*, which goes from 1 to 7. *Hour* indicates the hour at which the crash occurred, and it goes from 0 to 23. Two binary features *Daytime* and *Nighttime* indicate if it is daytime from 6:00 to 18:00 or nighttime from 18:00 to 6:00. A textual feature *Day_night_fct_* either describes day- or nighttime.

Crash Frequency	**Amsterdam**	**Utrecht**	**Rotterdam**	**The Hague**
1 crash on road	1258	511	1148	1,681
2 crashes on road	9	5	8	10
3 crashes on road	0	0	1	0
Crashes Total	1267 (0.35%)	516 (0.16%)	1157 (0.16%)	1691 (0.1%)
0 crashes	365,309 (99.65%)	323,388 (99.84%)	716,035 (99.4%)	1,654,453 (99.9%)
Total	366,576 (100%)	323,904 (100%)	717,192 (100%)	1,656,144 (100%)

Table 1: Crash Frequency in Different Cities

3.2.2 Bicycle and motorised vehicle exposure

To determine the number of cyclists that traverse a road segment at any given hour, Uijtdewilligen et al. [10] use GPS tracks from the Dutch Bicycle Counting Week, together with local hourly bicycle count data from count stations. To get the hourly motorised vehicle volumes motorised vehicle count data and estimations of the weekly average volumes from local transport models are used. The resulting variables, the bicycle- and motorised vehicle exposure (*Bic_exp* & *MV_exp*), have a temporal resolution of 168 for each road segment. Exposure indicates how many bicycles or motorised vehicles travelled on the road on average for a specific hour over the volume of the road.

3.2.3 Cycling infrastructure

The Netherlands has three main cycling infrastructures: separated bicycle tracks, bicycle lanes (indicated on the roadway), and mixed traffic conditions. (cyclists sharing the road with motorised vehicles). Uijtdewilligen et al. [10] combined bicycle lanes and mixed traffic conditions in the on-road category. This leaves four road types: A 50 km/h speed limit road with a separated cycling lane (*Speed_50_separated*), a 50 km/h speed limit road where cyclists are on the roadway (*Speed_50_on_road*), a 30 km/h speed limit road with a separated cycling lane (*Speed_30_separated*) and a 30 km/h speed limit road where cyclist are on the roadway (*Speed_30_on_road*). In addition, there is a 5'th feature (*Speed_infra_fct__*) giving a textual description of the road type a road segment belongs, for example, "Speed_30_on_road". Table 2 shows how the different roads are distributed in the 4 cities. Information about the speed limits comes from Rijkswaterstaat, and information about the road segments comes from the Dutch Cyclists' Union.

Speed Variable	**Amsterdam**	**Utrecht**	**Rotterdam**	**The Hague**
Speed 50 separated	1622 (74.34%)	1152 (59.75%)	2080 (48.72%)	3388 (34.37%)
Speed 50 on road	289 (13.24%)	246 (12.76%)	590 (13.82%)	1422 (14.42%)
Speed 30 separated	41 (1.88%)	145 (7.52%)	366 (8.57%)	393 (3.99%)
Speed 30 on road	230 (10.54%)	385 (19.97%)	1233 (28.88%)	4655 (47.22%)
Total	2182 (100%)	1928 (100%)	4269 (100%)	9858 (100%)

TABLE 2: Amount of roads based on Speed

3.2.4 Network structure

Important network-related bicycle safety characteristics are centrality, intersection density(traffic lights, roundabouts, and unsignalised), and grade separation. Centrality (*Betweenness_norm*) indicates how central a road is in the network. The higher the score, the easier it is to access the road from other roads. The formula for betweenness centrality comes from [10]. For node v the betweenness centrality is given by :

$$C_B(v) = \sum_{u \neq v \neq w} \frac{\sigma_{uw}(v)}{\sigma_{uw}}. \tag{3}$$

Here, $C_B(v)$ is the betweenness centrality of node v. σ_{uw} is the total number of shortest paths between nodes u and w. $\sigma_{uw}(v)$ is the number of shortest paths from node u to node w that pass through node v.
The intersection densities measure how close intersections are in the network. The intersections are divided into three categories: traffic lights (*Traf_ light_ dens_ scaled*), roundabouts (*Roundabout_ dens_ scaled*) and unsignalised (*Unsignalised_ dens_ scaled*). Some road sections are grade-separated (*Grade_ separated*), like tunnels, bridges and viaducts. These road segments acts as bottlenecks in the network as they force the traffic to one location. The data about the cycling network structure is taken from the Dutch Cyclists' Union.

3.2.5 Popular destinations

Popular destinations might have an effect on bicycle safety [10]. Therefore, if a popular destination is within 150 meters of a road section, it is indicated by a 1 in the data and a 0 otherwise. Popular destinations are commercial facilities (*Commercial_ 150*), offices (*Offices_ 150*), railway stations (*Railway_ 150*) and educational facilities (*Educ_ 150*). Table 3 shows how many road segments per city lie in a 150 meter radius of these popular destinations. "Other" in the table indicates the amount of roads that do not lie within 150 metres of a popular destination. Data about commercial facilities and offices is taken from 'het Kadaster' and the provinces of North Holland, South Holland, and Utrecht. The Dutch Cyclists' Union provided the data about the railway station entries. Information about educational facilities is taken from DUO.

Popular Destination	**Amsterdam**	**Utrecht**	**Rotterdam**	**The Hague**
Offices 150	335 (15.35%)	526 (27.28%)	629 (14.73%)	1154 (11.71%)
Other	1847 (84.65%)	1402 (72.72%)	3640 (85.27%)	8704 (88.29%)
Commercial facilities 150	1042 (47.75%)	600 (31.12%)	1664 (38.98%)	3132 (31.77%)
Other	1140 (52.25%)	1328 (68.88%)	2605 (61.02%)	6726 (68.23%)
Railway stations 150	24 (1.10%)	64 (3.32%)	78 (1.83%)	71 (0.72%)
Other	2158 (98.90%)	1864 (96.68%)	4191 (98.17%)	9787 (99.28%)
Educational facilities 150	459 (21.04%)	432 (22.41%)	1235 (28.93%)	1633 (16.57%)
Other	1723 (78.96%)	1496 (77.59%)	3034 (71.07%)	8225 (83.43%)
Total	2182 (100%)	1928 (100%)	4269 (100%)	9858 (100%)

Table 3: Popular Destinations

3.2.6 Municipality indication

Because there might be underlying city-specific factors, the data for each of the four cities include a binary variable indicating to which municipality a road segment belongs. Utrecht (*UTR_ mun*), Amsterdam (*AMS_ mun*), Rotterdam (*ROT_ mun*) and The Hague (*TH_ mun*). In addition, there is a 5'th feature (*mun_ fct_*) giving a textual description of to which city a road segment belongs, for example, "Amsterdam".

3.3 The data tables

The data consists of two tables: the crash data table and the adjacency data table. The crash data table contains columns with all the feature variables. Table 6 gives an example of the crash data table. Note that not all the features are shown here.
The adjacency table contains two columns. One with the road index *NWB_ID* and one with a string containing all the road indices to which this road segment is connected. An example of the adjacency data table is given in Table 5. The graph is created by combining these two tables. Each node gets its features from the crash table, and the adjacency table creates the edges.
There is some mismatched data since the adjacency table has more road indices than the crash data table, which has information on roads. For example, some road segments in Amsterdam connect to a road segment with $NWB\_ID = 2105$. However, this road is not in the crash data table. Road segments that did not contain any information were removed from the data. Another error that might occur is that the string variable in the second column in the adjacency table does not have enough bytes to contain the road indices of all the connected roads. An example is given in Table 5 where 16 (a road segment in Utrecht) is given a red colour because it should not have a connection to road index 16298 (a road segment in Rotterdam). These non-existing road connections were taken out of the data.

Index	**NWB_ID**	**Day**	**Hour**	**Daytime**	**Night_time**	**Day_night_fct**	**...**
0	1	1	0	0	1	Night_time	...
1	1	1	1	0	1	Night_time	...
2	1	1	2	0	1	Night_time	...
3	1	1	3	0	1	Night_time	...
4	1	1	4	0	1	Night_time	...

TABLE 4: Example of the crash data table for the first 5 rows and 6 columns (features)

NWB_ID	**adj_ID**
16298	16293,16297,16300,16396,16397,16398,16399,16403,16

TABLE 5: Adjacency table row with wrong connection

4 Methodology

This Section will describe the methodology used to address the research problem and how the data was made binary to handle class imbalance. Thereafter, an explanation will be given of which models were chosen and how they work. In addition, we will describe the choices for the loss function. Lastly, we will define the different explanation methods.

4.1 Data Preparation

Based on the feature *mun_fct_*, the data is divided according to city. Some roads had two traffic speed indications in their binary variables for speed, *Speed_50_separated* and *Speed_50_on_road*, for example. Based on the feature *Speed_infra_fct__*, one binary speed limit per road segment was assigned. After that, all the non-numerical (textual) features are removed from the data since all the textual features have an equivalent numerical feature that can be used for training.

To decrease the class imbalance, we get rid of the temporal dimension, which leaves multi-class spatial data. To make the data spatial all continuous variables that change over time are summed. For example, for *Bic_exp* and *MV_exp*, all 168 time variables per road segment were summed. Continuous variables that stay constant over time, like the length of a road segment (*length_km*), which is the same every day at every hour, remain the same. All binary features, like the road types, remain the same. This creates a multi-class spatial problem, where some roads have had multiple accidents, and other roads have had no accidents. Table 6 shows how the road segments are distributed according to the number of crashes. Here, Amsterdam has 1550 roads where no accident happened at all, 340 roads where 1 accident occurred and even 1 road with 13 crash accidents between 2015 and 2019.

Number of crashes	**Amsterdam**	**Utrecht**	**Rotterdam**	**The Hague**
0	1550 (71,1%)	1585 (82,2%)	3503 (82,1%)	8606 (87,3%)
1	340 (15,58%)	242 (12,55%)	537 (12,58%)	964 (9,78%)
2	143 (6,55%)	65 (3,37%)	137(3,21%)	191 (1,94%)
3	57 (2,61%)	19 (0,99%)	54 (1,26%)	60 (0,61%)
4	43 (1,97%)	6 (0,31%)	17 (0,40%)	22 (0,22%)
5	24 (1,01%)	5 (0,26%)	10 (0,23%)	8 (0,08%)
6	11 (0,50%)	2 (0,10%)	7 (0,16%)	4 (0,04%)
7	3 (0,14%)	2 (0,10%)	0 (0,00%)	1 (0,01%)
8	6 (0,27%)	1 (0,05%)	3 (0,07%)	2 (0,02%)
9	3 (0,14%)	1 (0,05%)	0 (0,00%)	0 (0,00%)
10	0 (0,00%)	0 (0,00%)	1 (0,02%)	0 (0,00%)
12	1 (0,05%)	0 (0,00%)	0 (0,00%)	0 (0,00%)
13	1 (0,05%)	0 (0,00%)	0 (0,00%)	0 (0,00%)
Total	2182 (100%)	1928 (100%)	4269 (100%)	9858 (100%)

Table 6: Distribution of the number of crashes for each city.

Any model trained on the data in Table 6 would experience class imbalance in predicting which nodes would have no crashes and which nodes would, for example, experience five or more crashes. To further decrease any class imbalance in the data, the data is transformed from a multi-class problem to a binary-class problem. To adapt the graph for binary classification, the variables *Crash_freq*, *Crash_fatal*, *Crash_injury*, and *Crash_binary* are converted into binary variables, where zeros remain unchanged, and any value greater than zero is assigned a value of one. Now, all the non-zero classes in the spatial problem of Table 6 belong to the same class 1, indicating if an accident has ever happened on that road. Making the problem spatial first and then binary makes the class imbalance smaller. This can be seen by comparing Table 1 to Table 7.

All feature columns in the data are normalised using Min-Max Scaling to avoid exploding/vanishing gradients and make training more stable. The formula for Min-Max Scaling is:

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}, \tag{4}$$

$x_i' \in [0, 1]$ is the normalized feature vector. The graph is made undirected by enforcing that all nodes with one directed edge have an additional edge in the reverse direction, ensuring mutual connectivity between the nodes.

Some of the variables initially in the data are not included in the node features at the start of training the models since they were deemed redundant. *'Bic_exp_log'* and *'MV_exp_log'* were taken out since they are similar to *Bic_exp* and *MV_exp*. *length_km* is taken out because it is correlated with the intersection density variables, and both *Bic_exp* and *MV_exp*. All the municipality indications are taken out *'UTR_mun', 'AMS_mun', 'ROT_mun', 'TH_mun'*. Since the feature *Crash_binary* is the target variable, the features *Crash_freq, Crash_fatal* and *Crash_injury* are taken out of the data.

Crash Binary	**Amsterdam**	**Utrecht**	**Rotterdam**	**The Hague**
No crashes	1550 (71, 1%)	1585 (82, 2%)	3503 (82, 1%)	8606 (87, 3%)
At least 1 crash	632 (28, 9%)	343 (17, 8%)	766 (17, 9%)	1252 (12, 7%)
Total	2182 (100%)	1928 (100%)	4269 (100%)	9858 (100%)

Table 7: Crash binary in different cities

This leaves the following 15 features used for training the models: *Bic_exp*, *MV_exp*, *Betweenness_norm*, *Speed_50_separated*, *Speed_30_separated*, *Speed_50_on_road*, *Speed_30_on_road*, *Grade_separated*, *Traf_light_dens_scaled*, *Roundabout_dens_scaled*, *Unsignalised_dens_scaled*, *Offices_150*, *Commercial_150*, *Railway_150*, and *Educ_150*. The feature *Crash_binary* indicates the 'ground truth' of the nodes that we want to predict.

4.2 Models

In this study, two different models were used. A Multi-Layer Perceptron as a baseline method that does not consider graph structure. And a Graph Convolution Network because it is simple and computationally less expensive compared to Graph Attention Networks. Also, GCNs have a straightforward aggregation mechanism, making it better for explanations. Furthermore, the choice for the loss functions and the explanation methods are addressed.

4.2.1 Multi Layer Perceptron

A Multi-Layer Perceptron (MLP) [52] consists of multiple layers of nodes/neurons: an input layer, a hidden layer and an output layer. The input layer takes the number of features as input, so the number of neurons equals the number of features. The hidden layers can consist of any number of neurons connected to the input and previously hidden layers. Each neuron is connected to the output of the neurons from the previous layer. For a layer t with n_t neurons, the output for the i-th neuron in layer t is computed as:

$$z_i^t = \sum_{j=1}^{n_{t-1}} w_{ij}^t \cdot a_j^{(t-1)} + b_i^t. \tag{5}$$

Here, z_i^t is the output of neuron i in the t'th layer, w_{ij}^t is the weight of the connection from neuron j in layer $(t-1)$ to neuron i in layer t, $a_j^{(t-1)}$ is the output from neuron j in the previous layer and b_i^t is the bias of neuron i in the t'th layer. The output of neuron i in the t'th layer is than calculated by:

$$a_i^t = \sigma(z_i^t). \tag{6}$$

Here, $\sigma(\cdot)$ is an activation function, like the ReLU- or sigmoid activation function. The formula for ReLU (Rectified Linear Unit) is:

$$\sigma(z) = \max(0, z). \tag{7}$$

It maps all values to z if $z \geq 0$ and to 0 if $z < 0$. The formula for the sigmoid activation function is:

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{8}$$

It maps all values z into a range $[0, 1]$. The weights in the network are updated via a process called backpropagation:

$$w_{ij} \leftarrow w_{ij} - \eta \cdot \frac{\partial L}{\partial w_{ij}}. \tag{9}$$

Here, w_{ij} is the weight between neuron i and j, η is the learning rate which controls how much each weight should be updated, and $\frac{\partial L}{\partial w_{ij}}$ is the gradient of the loss L with respect to the weight w_{ij}.

The Multi-Layer Perceptron trained for this study consists of four layers. The input layer is composed of 15 neurons since it has to equal the number of input features. Then, there are two hidden layers, both having 15 neurons. This ensures the MLP model has the same number of weights as the GCN model. Each neuron in these hidden layers uses ReLu (Equation (7)) as its activation function. The output layer consists of one neuron, with a sigmoid (Equation (8)) as its activation function, mapping the final value to $[0, 1]$. This model was trained with an Adam optimiser with a learning rate of 0.01 for 500 epochs.

4.2.2 The Graph Convolution Network

The Graph Convolution Network [14] from Section 2.1.3 updates each layer through the following formula:

$$H^{(t+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{t}W^{t}\right). \tag{10}$$

Here, $H^{(t+1)} \in \mathbb{R}^{n\times F'}$ is the matrix after the convolution at the $(t+1)$'th layer, with n nodes and F' node features. Each row in this matrix represents a single node $h_i^{(t+1)} \in \mathbb{R}^{F'}$, with $i = 1, \ldots, n$. The matrix $\tilde{A} = A + \mathbf{I}$ is the graph Adjacency matrix with self-loops. Here, A is the adjacency matrix, and $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix that ensures self-loops. The matrix $\tilde{D}$ is a diagonal matrix with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. And $W^t \in \mathbb{R}^{F\times F'}$ is the linear transformation matrix which will be trained. The dimensions F and F' represent the number of node features in the t'th and $(t+1)$'th layers. For a single node in the network, the node update is:

$$h_i^{(t+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{1}{\sqrt{\tilde{D}_{ii}\tilde{D}_{jj}}}\tilde{A}_{ij}h_j^t W^t\right). \tag{11}$$

Because of the matrix multiplication in Equation (10), in Equation (11), the aggregation and combine functions of Equations (1) and (2) in Section 2.1.2 are joined in a single summation of the neighbouring nodes and the last node embedding of node i. Instead of using two separate weight matrices C_j and B_i, a single weight matrix W^t is used.

The Graph Convolution Network consists of three layers. This means that the GCN can aggregate feature information from its 3-hop node neighbourhood or three road segments away. The first convolution layer has ReLu (Equation (7)) as its activation function and $W^1 \in \mathbb{R}^{15\times 15}$, so it outputs 15 hidden features. The same holds for the second layer, so $W^2 \in \mathbb{R}^{15\times 15}$. The dimensions of the weight matrices were chosen to contain the same amount of feature representations in the updated layers as the amount of original features. The third layer has a weight matrix of $W^3 \in \mathbb{R}^{15\times 1}$; thus, the output layer consists of a single value. The output layer has the sigmoid (Equation (8)) as its activation function, mapping the final value to $[0, 1]$. The model was trained for 500 epochs, and the Adam optimiser, with a learning rate of 0.01, was chosen.

4.3 Class Imbalance

To handle the class imbalance in the data, two loss functions from Section 2.3 are used. The choice to use loss functions is based on the fact that we wanted to stay as close to reality as possible. Utilising methods like interpolation that create synthetic nodes and thus add nodes and thus add roads to the network is unreasonable.
This study implements two types of losses. The first is binary loss cross-entropy loss, which is used for binary classification problems. The formula for the binary cross-entropy loss is:

$$\ell(x, y) = L = \{l_1, l_2, \ldots, l_N\}^\top. \tag{12}$$

$$l_n = -w_n\Big[y_n \cdot \log(x_n) + (1 - y_n) \cdot \log(1 - x_n)\Big]. \tag{13}$$

Here y_n is the target label or ground truth, and x_n is the predicted probability. w_n is an optional weight vector set to 100 because it showed better results for the Class Balanced loss.

To decrease the class imbalance, a Class Balanced loss function was chosen as the second loss because of its applicability in combination with other losses like the BCE loss above. Class Balanced (CB) loss [39] is defined as:

$$L(p; y) = \frac{1 - \beta}{1 - \beta^{n_y}} L(p; y). \tag{14}$$

Here, $\beta \in [0, 1)$ and n_y is the number of samples in the ground-truth classy. $\beta = \frac{n_y - 1}{n_y}$ ensures majority classes have a smaller effect on the loss function.

4.4 Explainability

4.4.1 GNNExplainer

For the explanation of the GCN Models, GNNExplainer [47] from Section 2.4 was chosen because it provides both an explanatory subgraph and the most important features in this subgraph. Let ϕ be the function that indicates a trained GNN model. It learns a prediction based on the computation graph $G_c(v)$, and the node features $X_c(v)$ from the neighbouring nodes of v. The size of the computation graph is dependent on the k-hop neighbourhood of v. The model prediction is expressed as $\hat{t} = \phi(G_c(v), X_c(v))$. This prediction is based on the conditional probabilities $P_\phi(Y|G_c, X_c)$, where Y denotes a random variable representing labels from the set $\{1, \ldots, C\}$, reflecting the likelihood of nodes belonging to any of the C classes.
The goal of GNNExplainer is to find a small subgraph $G_s \subseteq G_c(v)$ that includes the most critical nodes and edges necessary to make the same predictions. Another objective is to find the subset $X_S^F \subseteq X_S$, , with $X_S = \{x_j | v_j \in G_S\}$, that includes the most important features of this subgraph. By maximising the Mutual Information (MI) between the entropy $H(Y)$ and the conditional entropy $H(Y|G = G_S, X = X_S^F)$, GNNExplainer generates an explanation (G_S, X_S^F) for the prediction $\hat{t}$. The feature subset X_S^F is determined using a binary feature selector $F \in \{0, 1\}^d$, which acts like a mask to only keep important node features. The goal is to maximize the mutual information between the label Y and the pair (G_S, X_S^F):

$$\max_{G_S, F} \mathrm{MI}(Y, (G_S, F)) = H(Y) - H(Y \mid G = G_S, X = X_S^F) \tag{15}$$

Appendix B contains a more detailed description of the way GNNExplainer works. For each node $v \in V$ in the graph G, we train a GNNExplainer with 200 epochs. Each GNNExplainer returns a subgraph $G_S(v) = (V_s(v), E_S(v))$ and a feature importance matrix $X_S^F(v) \in \mathbb{R}^{n \times d}$ for that node v.
New variables were created using $G_S(v)$ and $X_S^F(v)$. The Road Return frequency, $f_{\text{road}} \in \mathbb{R}^n$, for the amount of times nodes occur in all subgraphs, and the feature importance matrix $F_{\text{imp}} \in \mathbb{R}^{n \times d}$, that records the overall feature importances for each node. Let $f_{\text{road},i}$ be the i'th entry of f_{road}, with $i = 1, \ldots, n$, where $n = |V|$, then:

$$f_{\text{road},i} = \sum_{v \in V} 1(i \in V_S(v)). \tag{16}$$

Now let $F_{\text{imp},k}$ be he k'th row of $F_{\text{imp}} \in \mathbb{R}^{n \times d}$, with $k = 1, \ldots, n$, where $n = |V|$. And let $X_{S,k}^F[i, j]$ be the entry at i'th row and j'th column of $X_{S,k}^F \in \mathbb{R}^{n \times d}$, the feature importance matrix of node k, then:

$$F_{\text{imp},k} = \sum_{i=1}^{n} X_{S,k}^F[i, :] \tag{17}$$

For each node in the graph, this gives the importance per feature. To be able to see the overall importance per feature of the graph, $F_{\text{total}} \in \mathbb{R}^d$, is given by:

$$F_{\text{total}} = \sum_{k=1}^{n} F_{\text{imp},k,j}, \quad \forall j = 1, \ldots, d. \tag{18}$$

4.4.2 Partial Dependence Plots

Another method generally used to explain the impact a feature has on the result is Partial Dependence Plots. One target feature is varied in partial dependence plots while keeping all other variables constant. Let ϕ be the function that indicates the model. Let $\phi(G, X)$ be the model prediction, where G is the graph and $X \in \mathbb{R}^{n \times d}$ the feature matrix. First we define a new matrix $\hat{X}$ as:

$$\hat{x}_{i,j} = \begin{cases} \frac{1}{n} \sum_{i=1}^{n} x_{ij} & \text{if } x_{i,j} \text{ is a non-binary feature} \\ x_{i,j} & \text{if } x_{i,j} \text{ is binary a binary feature} \end{cases} \tag{19}$$

For each non-binary feature $x_j \in \mathbb{R}^n$, the 5th and 95th percentiles are calculated as

$$x_{j,\min} = \text{quantile}(x_j, 0.05), \quad x_{j,\max} = \text{quantile}(x_j, 0.95).$$

Then, a range of evenly spaced values for x_j is created:

$$\mathcal{X}_j = \{x_{j,1}, x_{j,2}, \ldots, x_{j,M}\}, \quad x_{j,m} \in [x_{j,\min}, x_{j,\max}].$$

For each value $x_{j,m} \in \mathcal{X}_j$, we create a modified feature matrix $\hat{X}_{-j}$ where the j-th feature (column) is replaced by the constant value $x_{j,m}$ across all nodes, while all other features remain constant at their values in $\hat{X}$. Formally:

$$\hat{x}_{k,j} = \begin{cases} x_{j,m}, & \text{if } k = j, \\ \hat{x}_{k,j}, & \text{otherwise.} \end{cases} \tag{20}$$

Once the modified feature matrix $\hat{X}_{-j}$ is created for each $x_{j,m}$, the Partial Dependence for feature x_j is calculated as the average model prediction over all nodes:

$$\hat{\phi}_j(x_{j,m}) = \frac{1}{n} \sum_{k=1}^{n} \phi(G, \hat{X}_{-j}). \tag{21}$$

This will give M values for each non-binary feature x_j going from the 5th to the 95th percentile.

$$\{(x_{j,m}, \hat{\phi}_j(x_{j,m})) \mid m = 1, 2, \ldots, M\}.$$

This process is repeated to examine the effect of binary values. Each time, one binary feature x_j is set to 1. So for each binary feature l, in Equation (19), if $j = l$, $x_{i,j} = 1$. To make it realistic, all other binary speed variables are set to 0 in the case of the speed binary variables. So, all roads in the network can be turned into one road type only.

5 Results

In this Section, we will discuss the results. We will start by examining the histograms for each model. Then, we will explore which features were most important for creating explanations. Based on the GNNExplainer results, we look at the partial dependence plots to see the impact of the most important features. Lastly, we examine the road return frequency f_{road} of Section 4.4.1.

5.1 Histograms

This Section shows the different histograms created during training. A histogram was chosen to represent the results since it gives more information than accuracy. For accuracy, a threshold would have to be selected, making the accuracy threshold dependent on different models. A good classification of the nodes is indicated by 0-class or non-crash nodes being skewed to the left and 1-class node or crash nodes being skewed to the right of the Figures. Nodes (road segments) that were skewed to the right (towards 1) have a higher probability or likelihood of having a crash happen there, according to the models.

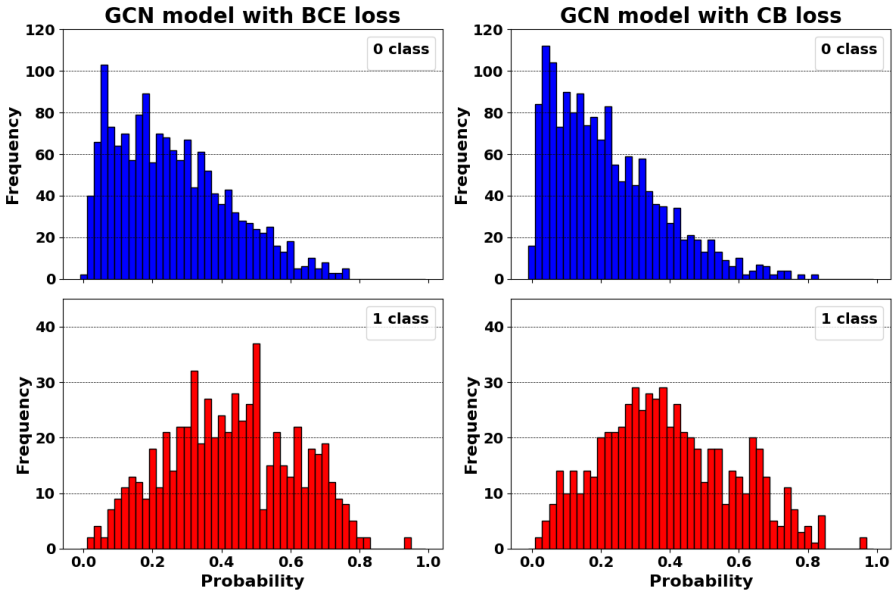

FIGURE 1: Histogram of the Amsterdam GCN model

Comparing the GCN models in Figure 1 for the city of Amsterdam shows that the probability of a 1-class prediction increases slightly when using the Class Balanced (CB) loss instead of the BCE loss. Interestingly, a couple of nodes get a higher probability than the others in both methods.

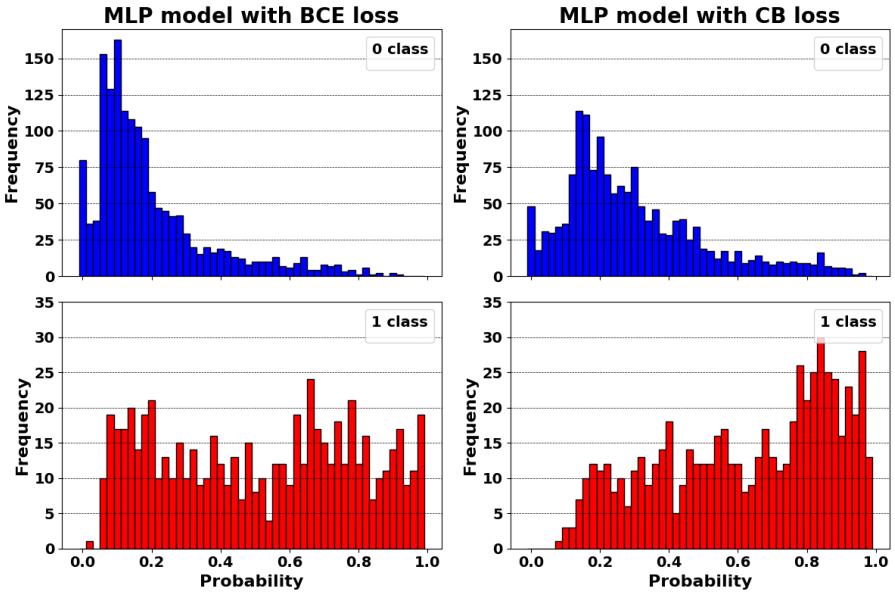

FIGURE 2: Histogram of the Amsterdam MLP model

Compared with the GCN models in Figure 1 , the MLP models in Figure 2 seem to be better at predicting a high crash probability for 1-class nodes. Using Class Balanced (CB) loss gives 1-class node a higher likelihood in the MLP model. However, it is also worse at predicting 0-class nodes, compared to both the MLP model with BCE loss and the GCN model with CB loss of Figure 1

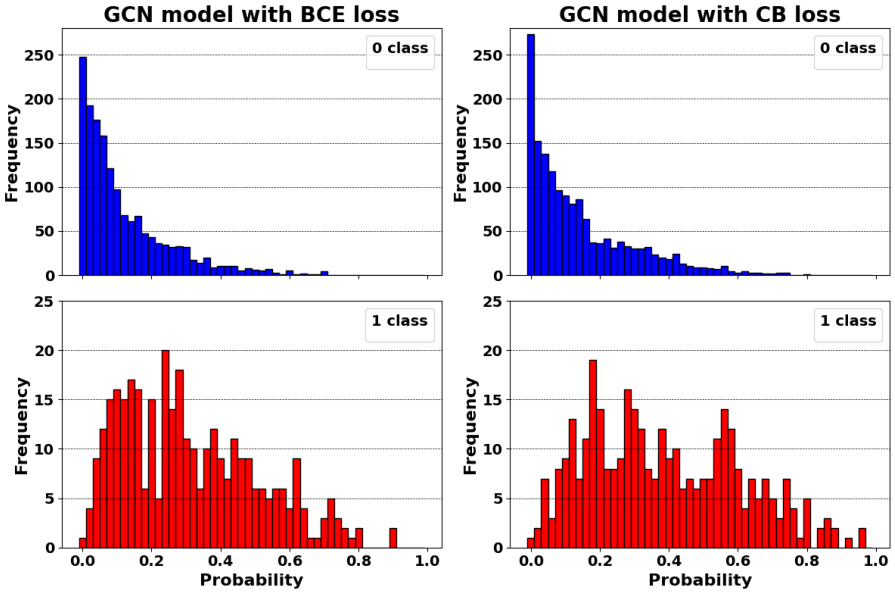

FIGURE 3: Histogram of the Utrecht GCN model

Looking at the results for the GCN models in Figure 3, it appears a couple of 1-class nodes again have a higher crash probability. Using Class Balanced loss increases the probability of 1-class nodes with a slight decrease in the amount of correctly classified 0-class nodes.

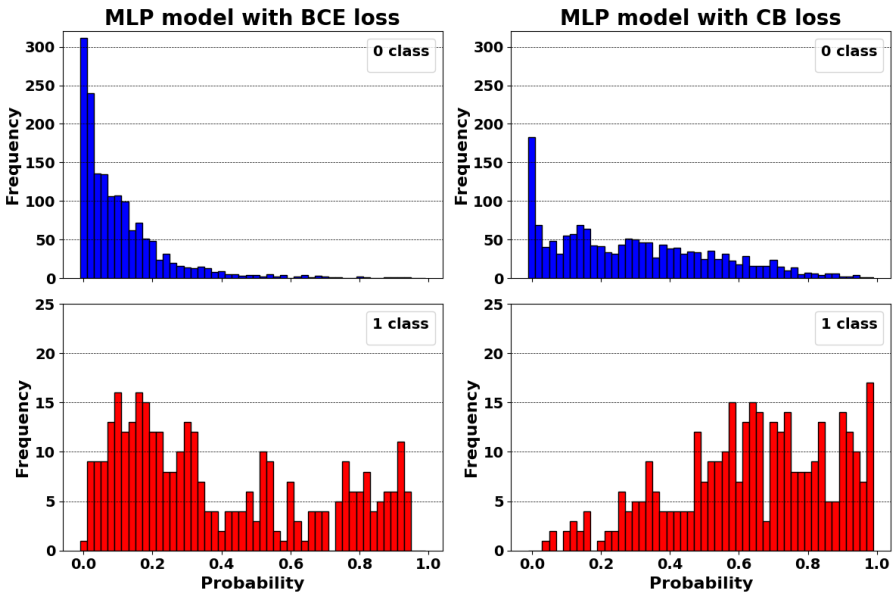

FIGURE 4: Histogram of the Utrecht MLP model

The MLP model with BCE Loss in Figure 4 has higher probabilities of 1-class nodes than the GCN Models in Figure 4. Using Class Balanced loss for the MLP model increases the average probability prediction for 1-class nodes. The average probability prediction of 0-class nodes also increases, indicating the model gives higher scores to 0-class nodes.

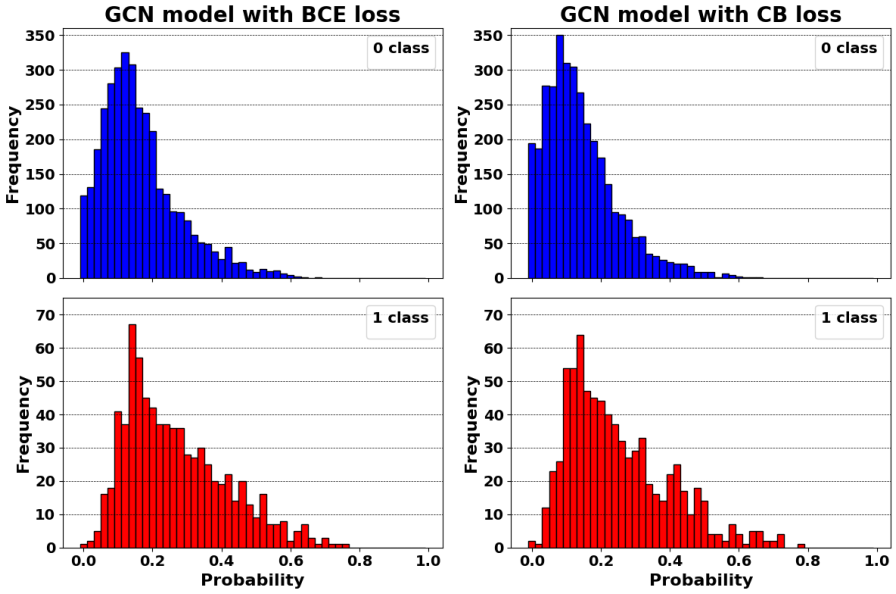

FIGURE 5: Histogram of the Rotterdam GCN model

Looking at the histograms in Figure 5 for Rotterdam, the GCN models have difficulty giving 1-class nodes a high probability. Using Class Balanced Loss does not lead to great improvements in the 1-class, and the probability of the 0-class is a little higher.

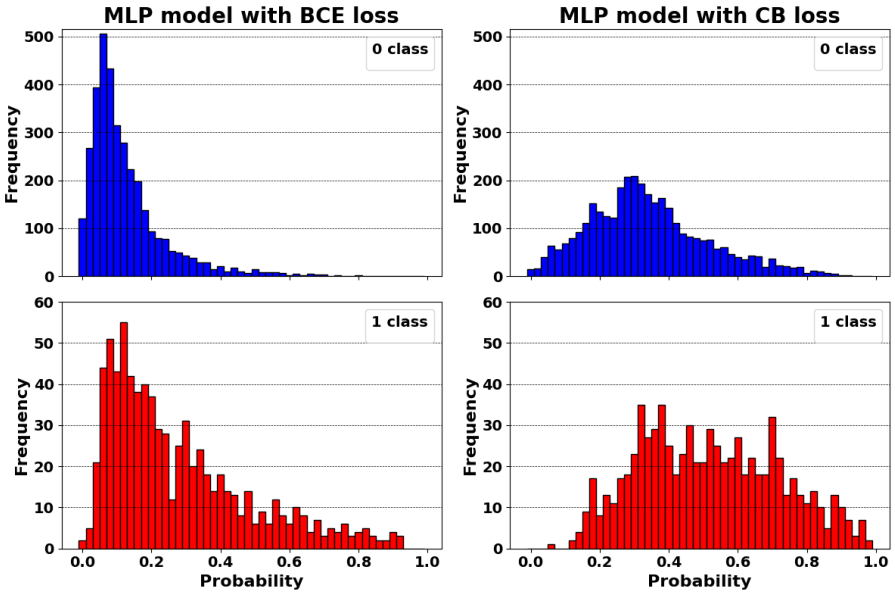

FIGURE 6: Histogram of the Rotterdam MLP model

The MLP model with BCE loss in Figure 6 is better at giving 1-class nodes a higher probability compared to the GCN models of Figure 5. However, most 1-class nodes still get a low probability. Using Class Balanced loss in the MLP increases the likelihood of 1-class nodes. Contrarily, it also increases the probability of 0-class nodes. It seems to increase the overall likelihood in all nodes and not separate the classes.

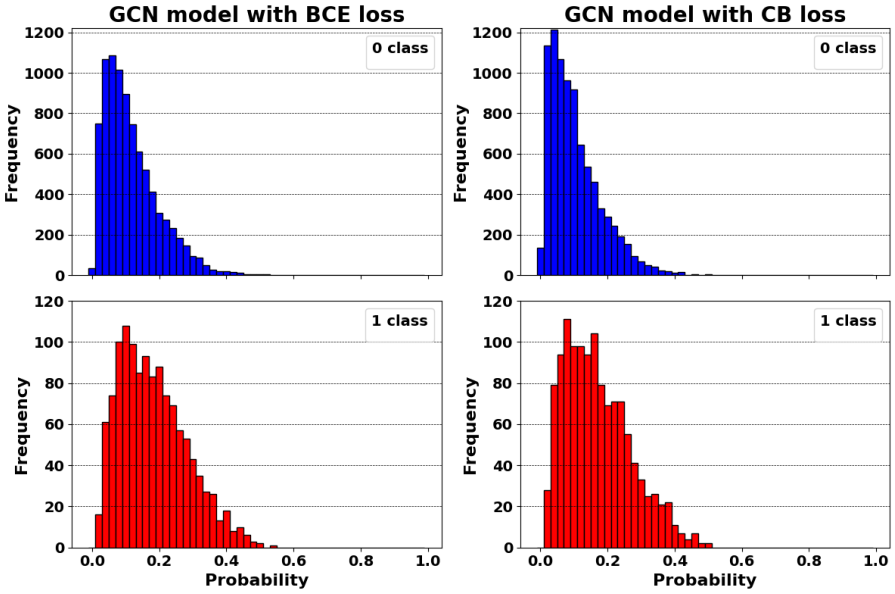

FIGURE 7: Histogram of the The Hague GCN model

For The Hague, the GCN model with BCE Loss in Figure 7 has a hard time giving 1-class nodes a high probability. Class Balanced loss does not improve this.

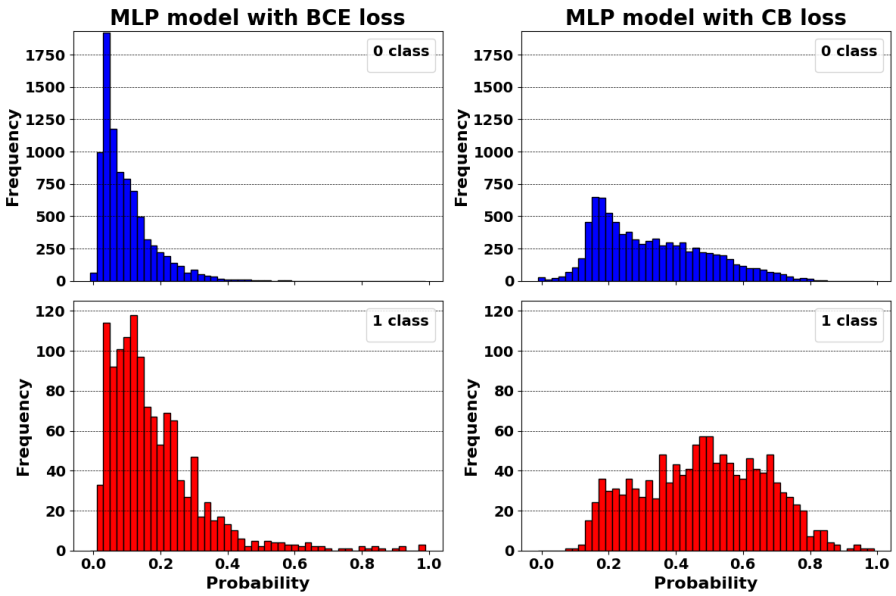

FIGURE 8: Histogram of the The Hague MLP model

The MLP models in Figure 8 is a little better at giving 1-class nodes a high probability. Using Class Balanced Loss has a significant effect on the MLP model. However, the increase in the number of 1-class nodes with a higher likelihood of being crash nodes is accompanied by an increase of 0-class probabilities. The overall probability is higher again, and it does not seem like there is much separation between the classes.

Overall, looking at the histograms, the GCN Model with Class Balanced loss seems to be better at predicting 1-class nodes than the GCN model with binary cross-entropy loss. For the MLP models, introducing class-balanced loss makes the models worse at predicting 0-class nodes. This is especially true for the cities Rotterdam and The Hague.

5.2 Feature Importance

In this Section, we will examine the importance of features according to the GNN Explainers. The variable F_{total} from Section 4.4.1 gives the total importance per feature. Note that feature importances are ordered and might differ per city.

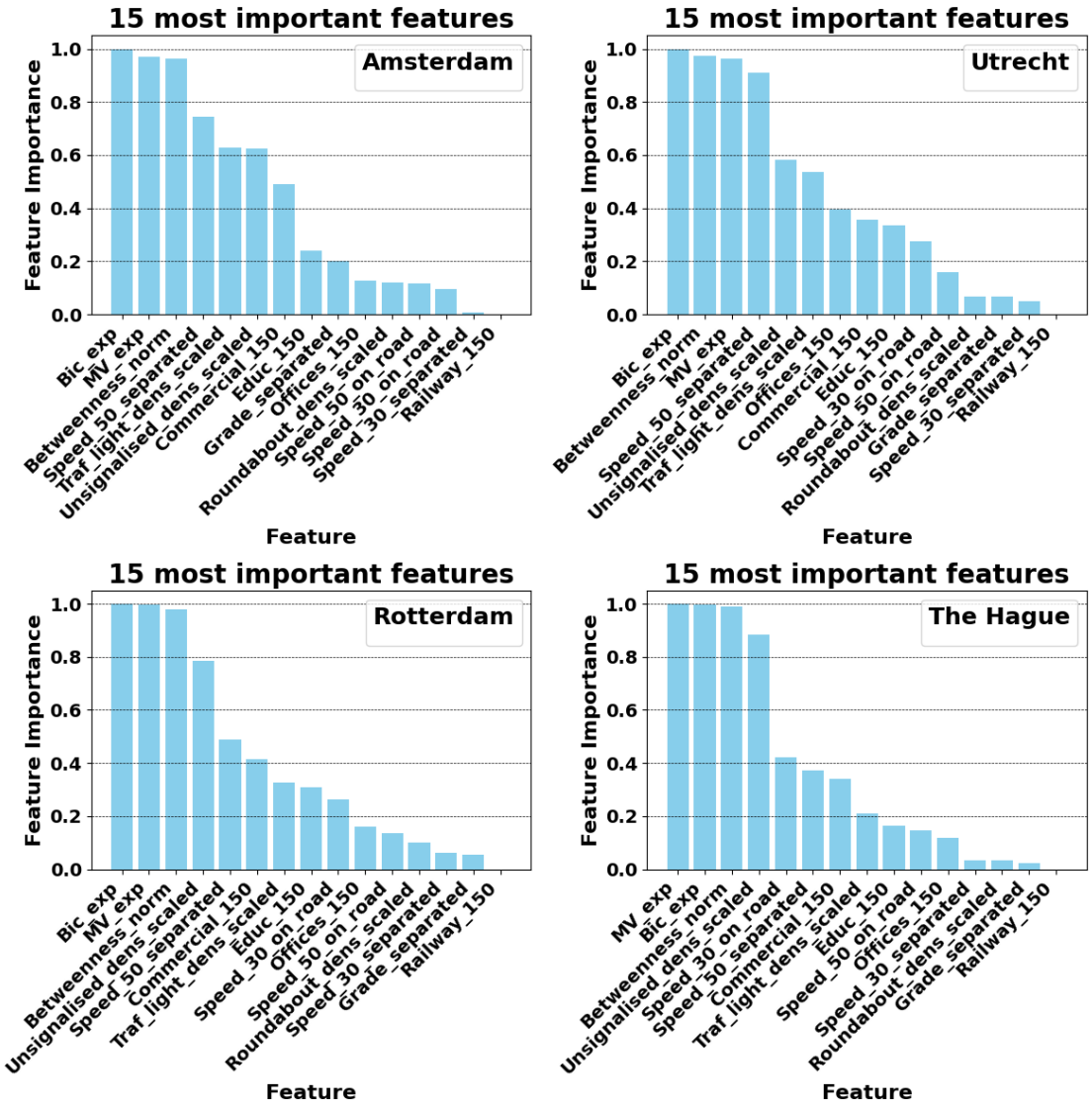

FIGURE 9: Most important features per city for GCN Model with BCE Loss

First, look at the GCN model with the Binary Cross Entropy loss in Figure 9. A common theme among all plots is that Bicycle exposure (*Bic_exp*), Motor Vehicle exposure (*MV_exp*), and the betweenness centrality (*Betweenness_norm*) are always in the top three of the most important features. Other important features are *Speed_50_separated*, *Speed_30_on_road*, *Commercial_150*, and *Educ_150* for the binary features. And, density features like *Unsignalised_dens_scaled* and *Traf_light_dense_scaled* are important for the GCN models to make predictions. For Utrecht, the feature *Offices_150* also scores high, but it is not so important in the other cities. This could be because Utrecht has more roads within a 150 meter radius of offices than other cities.

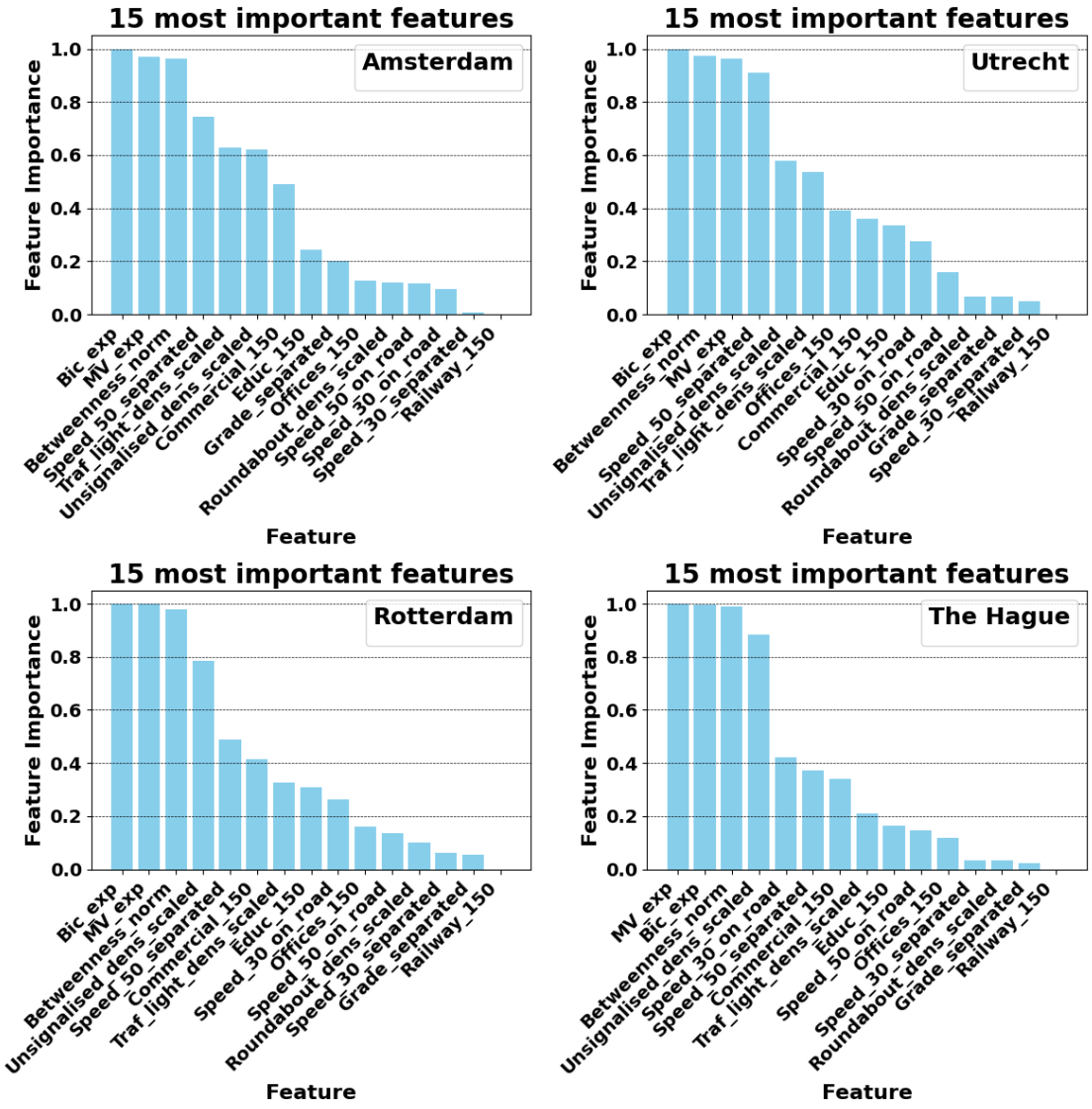

FIGURE 10: Most important features per city for GCN Model with CB Loss

Figure 10 shows the feature importances of the GCN model with Class Balanced loss. The order of importance stays the same as in Figure 9. Visually, nothing seems to be different between the different losses. There are some minor differences, but these are not visible in these Figures. Luckily, the feature importance of the GNNExplainers of each node is saved in the matrix F_{imp} from Section 4.4.1. A boxplot is made from this matrix, shown in Figure 11. The Figure shows some of the minor differences. Even though the overall distribution of *Bic_exp* seems similar, there are differences in the outliers, which means that some nodes in the graph see *Bic_exp* as an essential feature compared to other nodes. This is also true for features *Educ_150* and *Speed_30_on_road* where most nodes do not find them essential, but there are a lot of outlier nodes which give these features a relatively high feature importance. This difference might be explained by not all nodes being within a 150 meter radius of popular destinations. The boxplots of the results of the GNNExplainers from the other cities are also available and can be found in Appendix C.

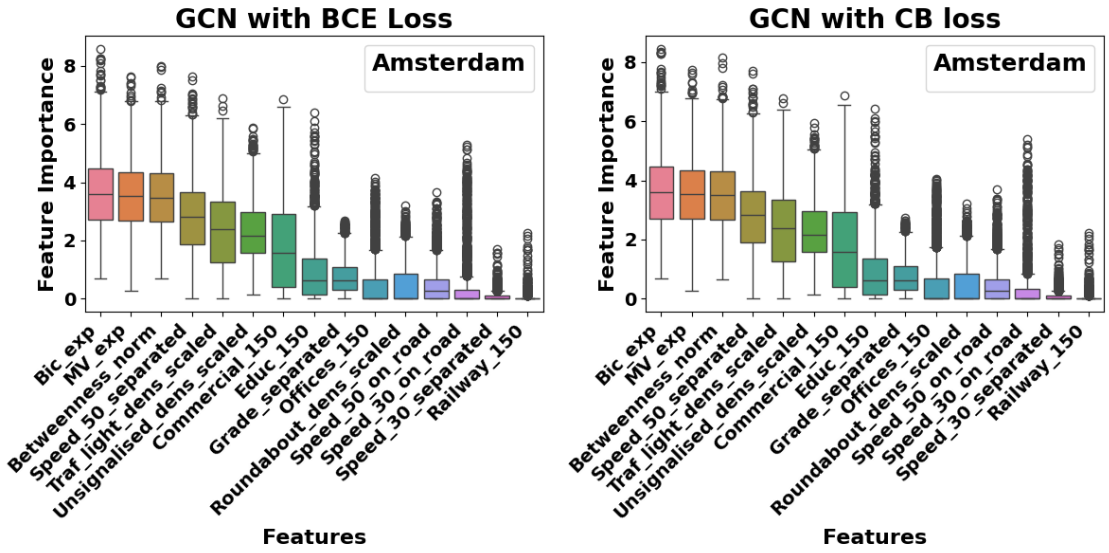

FIGURE 11: Boxplots of the GCN Model with BCE Loss vs the GCN Model with CB Loss for Amsterdam

5.3 Partial Dependence Plots

Based on the Figures in Section 5.1, we established that the GCN with Class Balanced (CB) Loss and the MLP with Binary Cross Entropy (BCE) were the best in separating the classes. In Section 5.2, we established that the most important features are *Bic_exp*, *MV_exp*, *Betweenness_norm*, *Traf_light_dens_scaled*, and *Unsignalised_dens_scaled* for the non-binary features. For the binary features, *Speed_30_*, *Speed_50_separated*, *Commercial_150*, and *Educ_150* had the highest importance scores in the GCN models across all cities. For each city, a partial dependence plot, explained in Section 4.4.2, per non-binary feature was made. Additionally, in each plot, the behaviour of the non-binary feature when the most important binary features are set to 1 is examined. We will only cover these plots for Amsterdam in the main text. The other plots can be found in Appendix D

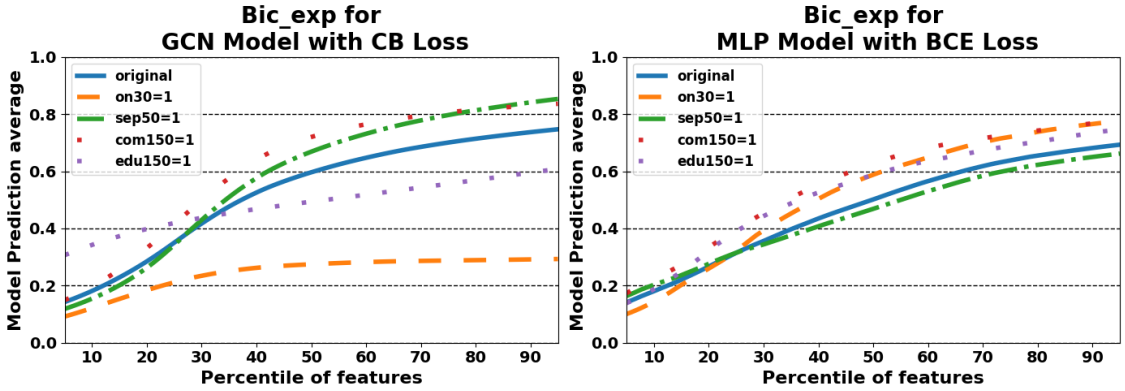

FIGURE 12: Partial Dependence plots for feature *Bic_exp* in Amsterdam

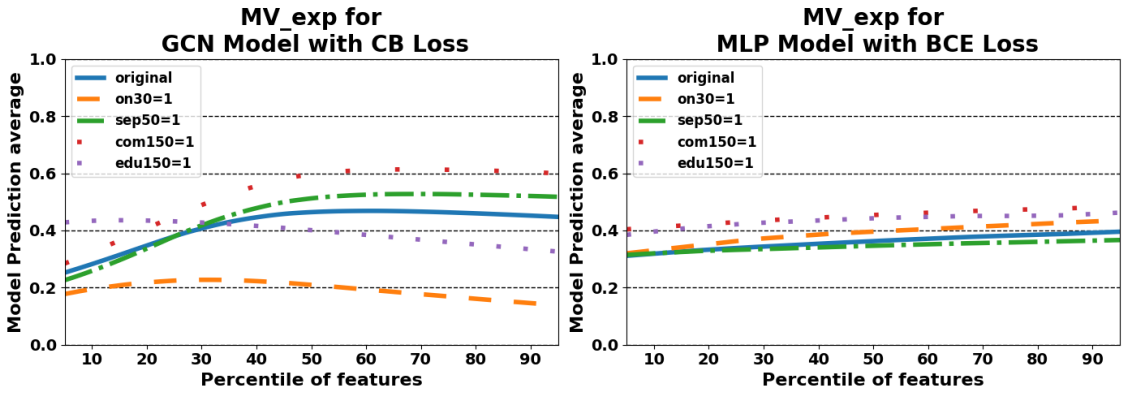

FIGURE 13: Partial Dependence plots for feature *MV_exp* in Amsterdam

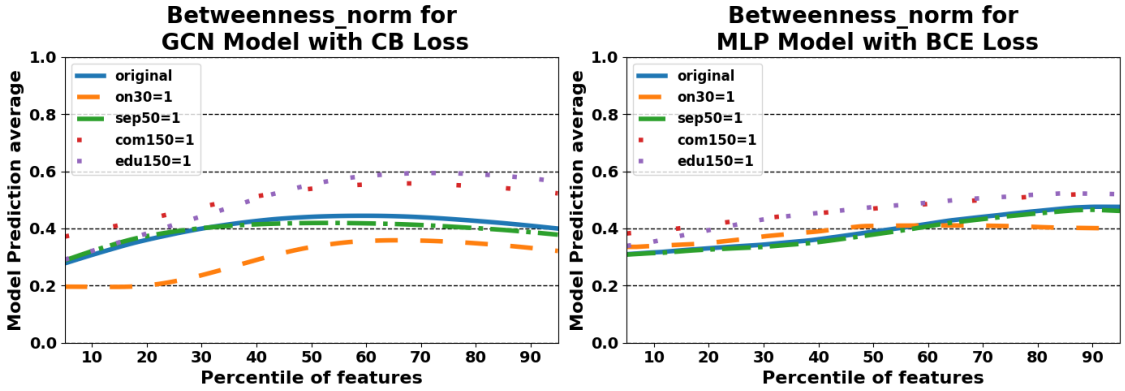

FIGURE 14: Partial Dependence plots for feature *Betweenness_norm* in Amsterdam

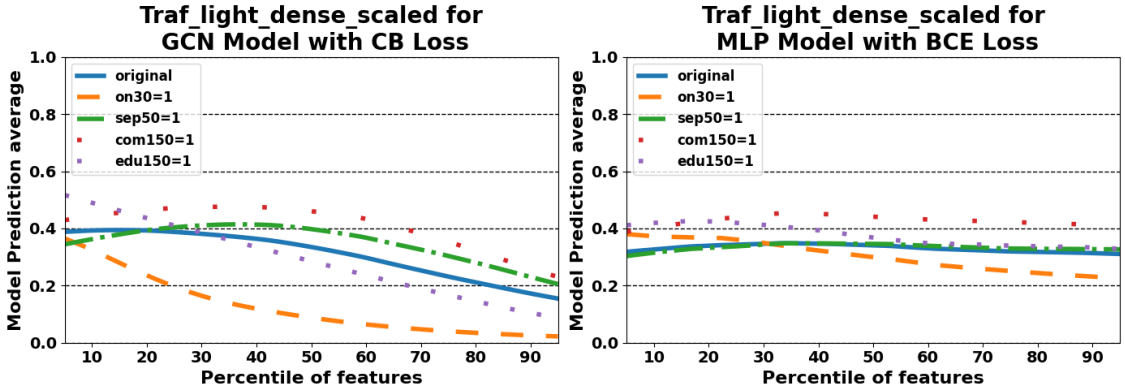

FIGURE 15: Partial Dependence plots for feature *Traf_light_dens_scaled* in Amsterdam

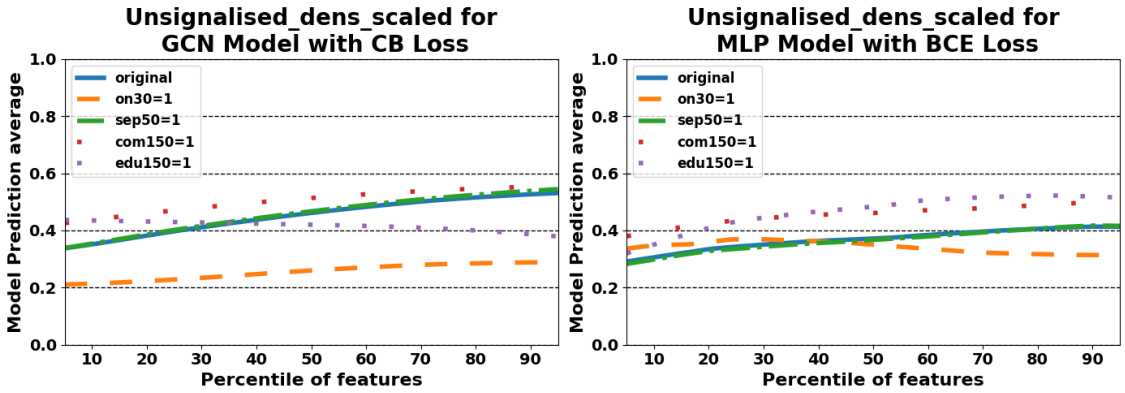

FIGURE 16: Partial Dependence plots for feature *Unsignalised_ dens_ scaled* in Amsterdam

It is obvious from Figure 12 that as number of cyclists (*Bic_ exp*) increases, the overall crash probability increases. Interestingly, when all roads are set to *Speed_ 50_ separated* roads, the crash probability for the GCN Model with CB Loss increases faster than in the original case. For the MLP Model with BCE Loss it decreases slightly. The converse is true for setting all roads to *Speed_ 30_ on_ road* roads. In the GCN Model with CB loss, this reduces the overall crash probability, and in the MLP model with BCE Loss, it increases more with increasing *Bic_ exp*. Close-by commercial facilities increase the overall likelihood of accidents. Close-by educational facilities also increase the probability of crashes as the number of cyclists rises. For the GCN Model, it crosses the original, and for the MLP model, it is higher than the original. All lines seem to increase and then converge.

MV_ exp in Figure 13 slightly increases the overall crash probability and then converges or decreases. Like Figure 12, *Speed_ 50_ separated* increases probability in the GCN models and decreases in the MLP model. Again, the converse is true for *Speed_ 30_ on_ road*, which decreases crash probability in the GCN model. *Commercial_ 150* increases the probability as *MV_ exp* increases. *Educ_ 150* decreases in the GCN model, whereas it increases in the MLP model.

As a road segment's centrality (*Betweenness_ norm*) in Figure 14 increases, the probability increases and then decreases. Both *Commercial_ 150* and *Educ_ 150* increase crash risk probability as *Betweenness_ norm* increases. *Speed_ 30_ on_ road* roads again decrease crash risk in the GCN model.

From Figure 15, we can see that the overall crash risk either stays constant or decreases as *Traf_ light_ dens_ scaled* increases. Making all roads *Speed_ 50_ separated* does not seem to have a significant effect compared to the original case; it increases crash probability slightly in the GCN model at a higher *Traf_ light_ dens_ scaled*, but it still decreases eventually. *Speed_ 30_ on_ road* does decrease crash risk probability faster for the GCN model. *Commercial_ 150* increases the overall crash risk compared to the original curve.

Figure 16 shows that increasing *Unsignalised_ dens_ scaled* increases the probability of a crash. This is lower in the GCN model if we set all roads to *Speed_ 30_ on_ road*. Setting all roads to *Speed_ 50_ separated* has almost no effect compared to the original. Nearby commercial facilities increase crash risk compared to the original case. Nearby educational facilities have a higher crash risk likelihood in the MLP model when *Unsignalised_ dens_ scaled* increases.

5.4 Network Visualisation

In this Section, we will look at some of the visualisations and see if they contain any information. Figure 17 shows the 4 networks of cities based on the Road Return Frequency, f_{road}, explained in Section 4.4.1.

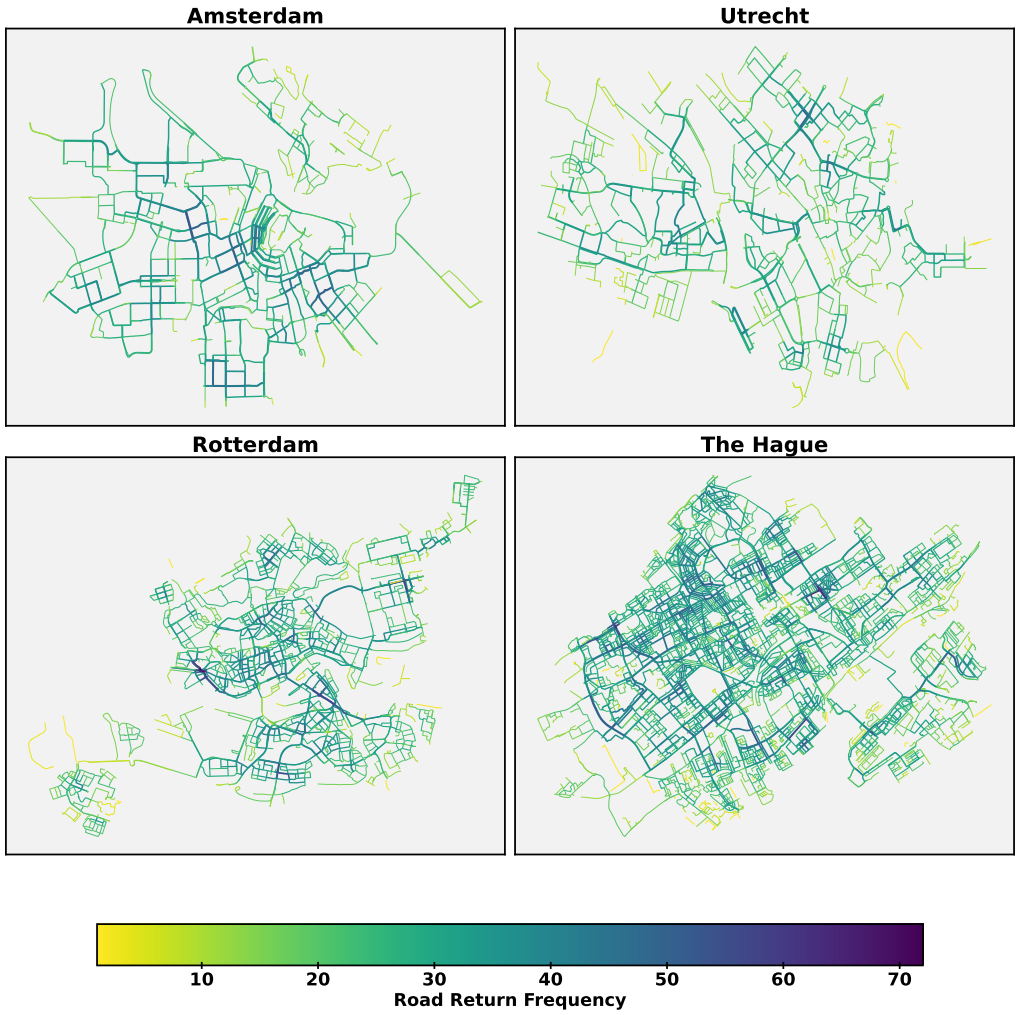

FIGURE 17: Road Return Frequency for all 4 cities

Figure 17 shows some road segments (nodes) are present in the explanation subgraphs $G_S(v)$ of 70 other roads. Also, The Hague has more road segments with a high Road Return Frequency. The Hague has the most roads so this is logical. For further analysis of the networks, the city of Amsterdam is used.

One possibility is that the number of times a road would appear in the explanation subgraphs of other road segments would be correlated with centrality since better interconnected roads might occur in more subgraphs. Therefore, Figure 18 compares the road centrality feature of Amsterdam with the Road Return Frequency. The Figures show that Road Return Frequency is not primarily based on the centrality feature.

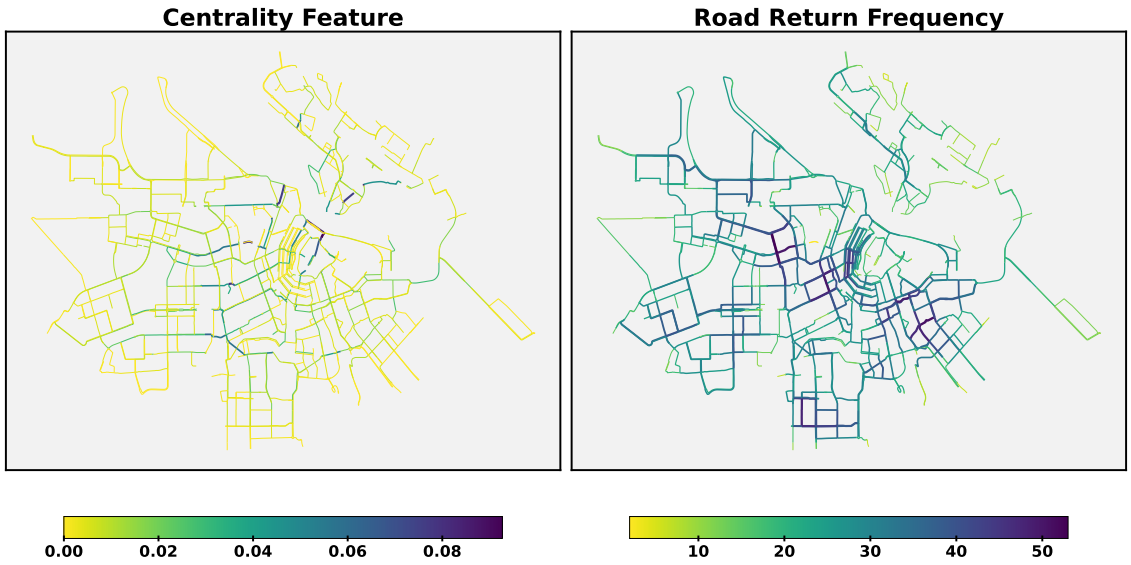

FIGURE 18: Comparison between Centrality and Road Return Frequency

Another possibility is that roads, v, with high feature importance values, $X_S^F(v)$, contained high values for the input features, X_v, and therefore, the features are deemed more important on the roads. Figures 19, 20 and 21 compare the initial feature values in the network with the feature importance given by the GNNExplainer, X_S^F. For comparison, the features and feature importances are normalized using Min-Max Scaling from Section 4.1. The Figures show that the feature importance, X_S^F, given by the GNNExplainer is independent of the values of the input features, X.

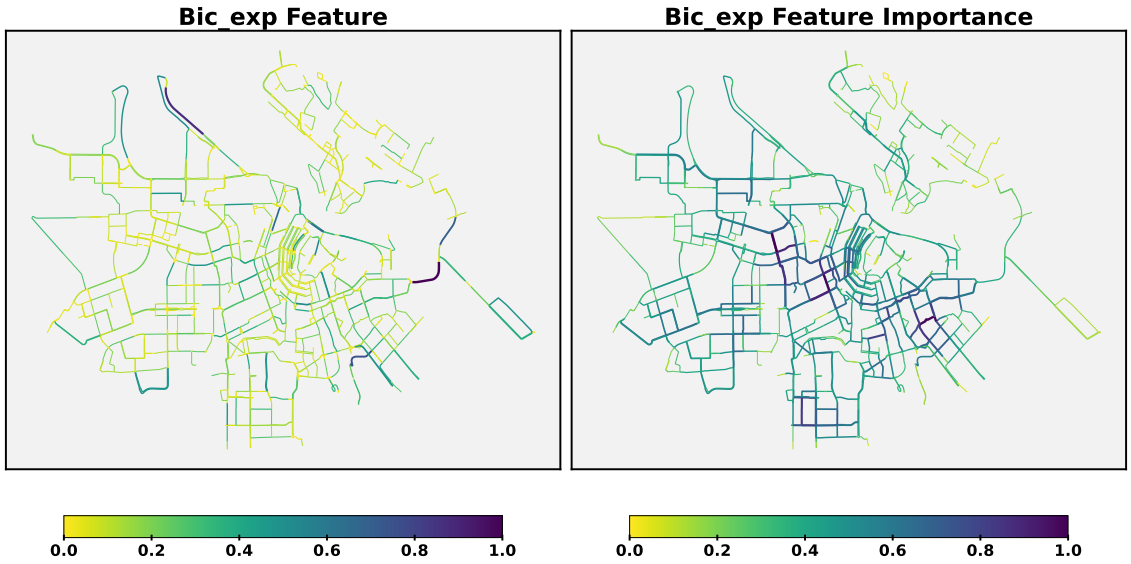

FIGURE 19: Initial feature values and GNNExplainer feature importance based on cyclist volumes

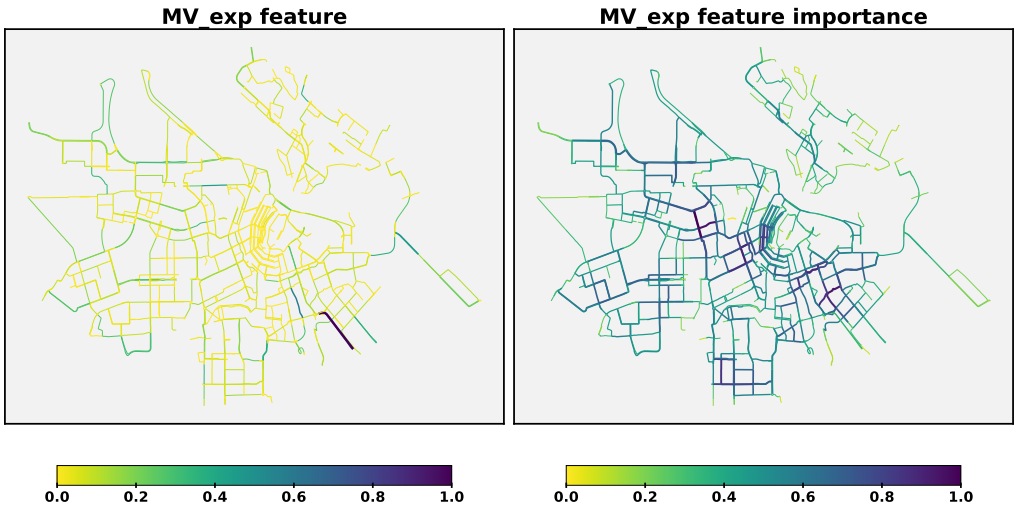

FIGURE 20: Initial feature values and GNNExplainer feature importance for motor vehicle volumes

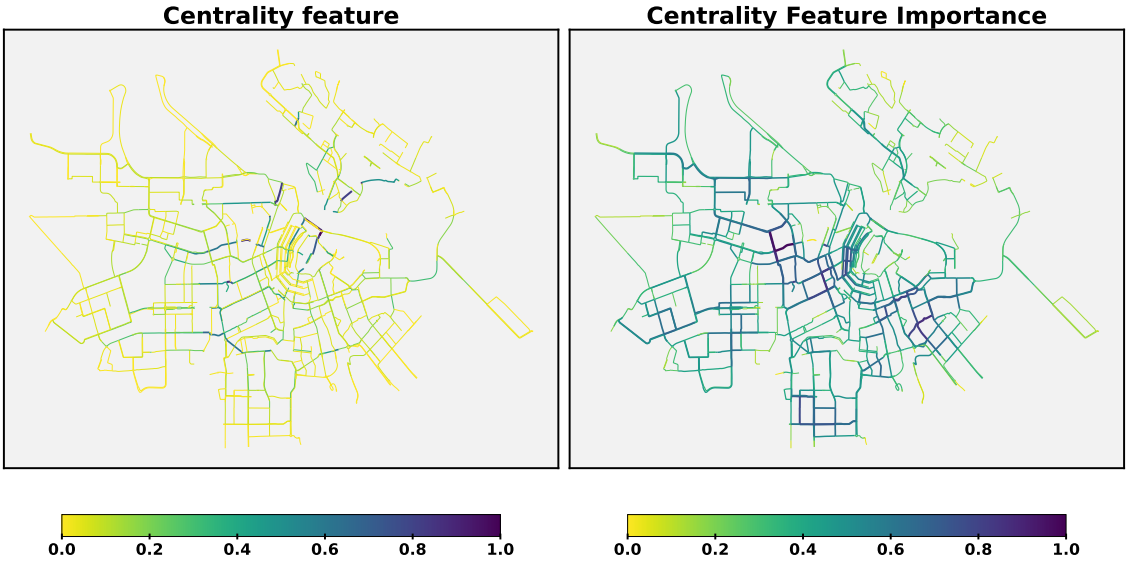

FIGURE 21: Initial feature values and GNNExplainer feature importance based on centrality

6 Discussion

In this Section, we discuss the implications of the results. First, we discuss how the different models performed and why there are differences. Then, we discuss how the different features respond to the different results. Lastly, we discuss the limitations of this research and possible directions for future research.

6.1 Model Performance

From the Figures in Section 5.1, we established that the models did not perform well in separating the classes. Of course, a complete separation of the classes was never expected. There are many things we do not know that could influence the data. The weather, for example, may have created bad visual conditions, leading to accidents. Or human failure could have caused an accident. Therefore, some safe roads might have a 1-class classification, and some unsafe roads might have a 0-class classification. Still, the GCN models find it difficult to split between crash and non-crash nodes. This can be due to many reasons. First, the utilised GCN models were not too complex, with a maximum 3-hop neighbourhood to reflect the real world. We also noticed that the larger the class imbalance is in the network, the harder it is for the GCN to make accurate predictions. One reason might be that when the class imbalance is significant, the more each node is surrounded by 0-class nodes. And since GCN uses all its neighbours to aggregate information, these models will have difficulty adjusting the weights for accurate predictions. Furthermore, Class Balanced loss improves in class separation in GCNs. The downside is that it reduces accurately classifying 0-class nodes.
The MLP model seems better at separating the 0- from the 1-classes than the GCN model. Introducing Class Balanced loss improves 1-class classification for the cities of Amsterdam and Utrecht. However, the MLP with Class Balanced loss performs worse for Rotterdam and The Hague because it now prioritises accurate prediction of 1-class nodes, making it much worse in predicting 0-class nodes. This is because the weighting factor in Class Balanced loss (Equation (14)) becomes significantly small for 0-class predictions since it is proportional to the number of samples in each class. There are more road segments in Rotterdam and The Hague; thus, the loss function will be penalised more heavily. This does not influence the GCN model with Class Balanced loss similarly because this model already has issues correctly classifying 1-class node in Rotterdam and The Hague.
The GNNExplainer provided an easy-to-understand overview from which explanations could be made. However, it does not say how features positively or negatively affect the crash risk prediction. Other explainability methods, such as partial dependence plots, are necessary to draw conclusions from the GNNExplainer's results.

6.2 Feature Influence

The partial dependence plots of Section 4.4.2 show some interesting results. If the number of cyclists on the road (*Bic_exp*) is high, the probability of crashes also rises. However, the increase in crash probability seems to converge, maybe indicating safety in numbers for the cyclists [53]. Larger volumes of motorised vehicles (*MV_exp*) also increase road crash risk. For the cities Amsterdam and Utrecht, these values converge similarly to *Bic_exp*. For Rotterdam and The Hague, the increase is more linear. Higher road centrality (*Betweenness_norm*) increases the crash risk and then converges, the same as *Bic_exp*. In all models, increasing the traffic light density *Traf_light_dens_scaled* either decreased or slowly increased and then decreased the crash probability. There are no gen-

eral trends in the behaviour of the partial dependence plots of the roads' unsignalised density (*Unsignalised_dens_scaled*). The behaviour was city- and model-dependent.
The proximity of commercial facilities in 150 metres (*Commercial_150*) increases the likelihood that a crash will happen. For educational facilities (*Educ_150*), it was model dependent if there was an increase or decrease. In all models, except the MLP model in Rotterdam, setting all roads to *Speed_30_on_road* reduced the probability of a traffic crash. When all streets were set to *Speed_50_separated* roads, all models in Utrecht, Rotterdam and The Hague showed decreased crash probability. In Amsterdam, this was not always the case. This is probably due to Amsterdam already having a large proportion of *Speed_50_separated* roads (74.34%). Therefore, it has less effect when more of these roads are added. This indicates that roads with a separate bicycle lane on 50 km/h speed limit roads decrease crash probabilities.

6.3 Limitations and Future research directions

There are limitations to this study. One limitation is that no data is available for all road segments in each city. These roads might influence the data in unseen ways. The second limitation is that other factors, such as weather and human influence, might affect the data. Third, turning the problem from a multi-class spatio-temporal problem in Section 4.1 to a binary class spatial problem results in the loss of information about temporal dependencies. Also, information about the specific effect features might have on class in the multi-class problem has been lost due to this simplification. Last, using no more than 15 intermediate feature representations in the hidden layers of Section 4.2 resulted in faster computation but might decrease complexity and, therefore, the model's ability to separate classes.

For future work, we recommend adding other Explainable Artificial Intelligence (XAI) methods to determine whether they prioritise features differently from those identified as most important by the GNN Explainer. The next step would be to try different models on the binary data. Another model, like Graph Attention Networks, might perform better in more class-imbalanced data like The Hague's by blocking unnecessary feature information during aggregation. Also, different loss functions could be inspected to see if they increase model performance. After trying these methods on the binary class spatial problem, we would turn the problem into a multi-class problem again since all the code for this already exists. Lastly, we recommend making the problem spatio-temporal and using different spatio-temporal models to optimise prediction. This will be computationally more expensive.

7 Conclusion

This study aims to identify the important road features that influence traffic crashes using Graph Neural Networks. The class imbalance in the data is handled by reducing the problem to a binary classification problem in Section 3 and using Class Balanced loss from Section 4. Taking the GCN and MLP as a model, we show what different effects aggregation can have on predictions in Section 5. By using a GNNExplainer in combination with Partial Dependence Plots, in Section 5, this paper shows what features are the most important and have the highest effect on road safety.
The results show that GCNs find it more challenging to classify minority classes compared to MLPs. Furthermore, the study finds that the volume of cyclists and motor vehicles and the centrality of a road segment have the most significant influence on crash likelihood. When the volume of cyclists in motor vehicles rises, so does the amount of crashes. A more connected road also has a higher likelihood of a crash occurring than less connected roads. An increased amount of traffic lights decreases the probability that a crash occurs. Furthermore, roads with a 50 km/h speed limit and a separate cycling lane have a lesser crash risk probability. The crash likelihood is even lower for roads with a 30 km/h speed limit and an on-road cycling lane. Commercial facilities within 150 metres of a road segment increase the probability of traffic accidents.
These conclusions show that local policymakers must consider road design when aiming to improve road safety. Specifically, increasing the number of traffic lights in areas with high traffic volumes or within 150 metres of commercial facilities can help reduce crashes. Furthermore, roads with a 30 km/h speed limit are by far the safest; however, separating the cycling lane on 50 km/h speed limit roads also provides safer roads whilst not reducing traffic flow. Overall, targeted interventions based on these findings can help create safer environments for cyclists.

8 Acknowledgments

I would like to express my sincere gratitude to my supervisors Clara and Baran for their guidance and support throughout this project. Clara, for always making the time, helping me with problems and guiding me in new directions. And Baran, who always re-energized my enthusiasm for the project with his knowledge of traffic safety, especially after I had some long coding hours with the data. I would also like to thank the Mathematics Department of the University of Twente for their support and help during the project.

References

[1] Servet Yanatma. *Cycling in Europe: Which countries and cities are the most and least bicycle-friendly?* Sept. 2023. URL: https://www.euronews.com/next/2023/09/19/cycling-in-europe-which-countries-and-cities-are-the-most-and-least-bicycle-friendly.

[2] Freya Slootmans. *Facts and Figures Cyclists*. Tech. rep. Brussels: European Commission, Oct. 2021. URL: https://road-safety.transport.ec.europa.eu/document/download/3057cf21-2770-4d31-982f-946b2eb1e4ae_en?filename=FF_cyclists_20220209.pdf.

[3] Ezra Hauer. "Statistical Road Safety Modeling". In: *Transportation Research Record* 1897.1 (2004), pp. 81–87. DOI: 10.3141/1897-11. URL: https://doi.org/10.3141/1897-11.

[4] Muhammad Marizwan Abdul Manan et al. "Road characteristics and environment factors associated with motorcycle fatal crashes in Malaysia". In: *IATSS Research* 42.4 (Dec. 2018), pp. 207–220. ISSN: 03861112. DOI: 10.1016/j.iatssr.2017.11.001.

[5] Shunchao Wang, Jingcai Yu, and Jingfeng Ma. "Identifying the heterogeneous effects of road characteristics on Motorcycle-Involved crash severities". In: *Travel Behaviour and Society* 33 (Oct. 2023). ISSN: 2214367X. DOI: 10.1016/j.tbs.2023.100636.

[6] Dominique Lord and Fred Mannering. "The statistical analysis of crash-frequency data: A review and assessment of methodological alternatives". In: *Transportation Research Part A: Policy and Practice* 44.5 (2010), pp. 291–305. ISSN: 09658564. DOI: 10.1016/j.tra.2010.02.001.

[7] Le Yu et al. "Deep spatio-temporal graph convolutional network for traffic accident prediction". In: *Neurocomputing* 423 (Jan. 2021), pp. 135–147. ISSN: 18728286. DOI: 10.1016/j.neucom.2020.09.043.

[8] Xian Liu et al. "Attention based spatio-temporal graph convolutional network with focal loss for crash risk evaluation on urban road traffic network based on multi-source risks". In: *Accident Analysis and Prevention* 192 (Nov. 2023). ISSN: 00014575. DOI: 10.1016/j.aap.2023.107262.

[9] Clint Morris and Jidong J. Yang. "Effectiveness of resampling methods in coping with imbalanced crash data: Crash type analysis and predictive modeling". In: *Accident Analysis and Prevention* 159 (Sept. 2021). ISSN: 00014575. DOI: 10.1016/j.aap.2021.106240.

[10] Teun Uijtdewilligen et al. "Examining the crash risk factors associated with cycling by considering spatial and temporal disaggregation of exposure: Findings from four Dutch cities". In: *Journal of Transportation Safety and Security* (2023). ISSN: 19439970. DOI: 10.1080/19439962.2023.2273547.

[11] Benjamin Sanchez-Lengeling et al. "A Gentle Introduction to Graph Neural Networks". In: *Distill* 6.8 (Aug. 2021). ISSN: 2476-0757. DOI: 10.23915/distill.00033.

[12] Jie Zhou et al. *Graph neural networks: A review of methods and applications*. Jan. 2020. DOI: 10.1016/j.aiopen.2021.01.001.

[13] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Feb. 2017, pp. 1263–1272. URL: https://proceedings.mlr.press/v70/gilmer17a.html.

[14] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of the 5th International Conference on Learn-*

ing Representations. ICLR '17. 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems*. Ed. by I Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

[16] Lingfei Wu et al., eds. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, 2022. ISBN: 978-981-16-6054-2. DOI: 10.1007/978-981-16-6054-2. URL: https://doi.org/10.1007/978-981-16-6054-2.

[17] Petar Veličković et al. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.

[18] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. "Large-scale learnable graph convolutional networks". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, July 2018, pp. 1416–1424. ISBN: 9781450355520. DOI: 10.1145/3219819.3219947.

[19] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (Jan. 2021), pp. 4–24. ISSN: 21622388. DOI: 10.1109/TNNLS.2020.2978386.

[20] Yujia Li et al. "Gated graph sequence neural networks". In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*. 2016. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083951493&partnerID=40&md5=e9f83489e0ce119ec4a2e8a305544f9a.

[21] Sijie Yan, Yuanjun Xiong, and Dahua Lin. "Spatial temporal graph convolutional networks for skeleton-based action recognition". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[22] Youngjoo Seo et al. "Structured Sequence Modeling with Graph Convolutional Recurrent Networks". In: *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2018, pp. 362–373. ISBN: 978-3-030-04166-3. DOI: 10.1007/978-3-030-04167-0{_}33. URL: https://doi.org/10.1007/978-3-030-04167-0_33.

[23] Jiani Zhang et al. "GaAN: Gated attention networks for learning on large and spatiotemporal graphs". In: *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*. Vol. 1. 2018, pp. 339–349. URL: https://www.auai.org/uai2018/proceedings/papers/139.pdf.

[24] William L Hamilton, Rex Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 1025–1035. ISBN: 9781510860964.

[25] Fred L. Mannering and Chandra R. Bhat. "Analytic methods in accident research: Methodological frontier and future directions". In: *Analytic Methods in Accident Research* 1 (2014), pp. 1–22. ISSN: 22136657. DOI: 10.1016/j.amar.2013.09.001.

[26] Matthias Schlögl et al. "A comparison of statistical learning methods for deriving determining factors of accident occurrence from an imbalanced high resolution dataset".

In: *Accident Analysis and Prevention* 127 (June 2019), pp. 134–149. ISSN: 00014575. DOI: 10.1016/j.aap.2019.02.008.

[27] Srishti Agrawal et al. "Cycle fatalities in Delhi and their risk factors". In: *International Journal of Injury Control and Safety Promotion* (2024). ISSN: 17457319. DOI: 10.1080/17457300.2024.2389527.

[28] Shakil Ahmed et al. "A study on road accident prediction and contributing factors using explainable machine learning models: analysis and performance". In: *Transportation Research Interdisciplinary Perspectives* 19 (May 2023). ISSN: 25901982. DOI: 10.1016/j.trip.2023.100814.

[29] Jiahui Zhao, Pan Liu, and Zhibin Li. "Exploring the impact of trip patterns on spatially aggregated crashes using floating vehicle trajectory data and graph Convolutional Networks". In: *Accident Analysis and Prevention* 194 (Jan. 2024). ISSN: 00014575. DOI: 10.1016/j.aap.2023.107340.

[30] Yang Zhang et al. "A Multi-modal Graph Neural Network Approach to Traffic Risk Forecasting in Smart Urban Sensing". In: *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 2020, pp. 1–9. DOI: 10.1109/SECON48991.2020.9158447.

[31] Yaqin Ye et al. "Dynamic multi-graph neural network for traffic flow prediction incorporating traffic accidents". In: *Expert Systems with Applications* 234 (Dec. 2023). ISSN: 09574174. DOI: 10.1016/j.eswa.2023.121101.

[32] Mingyao Wu et al. "A multi-attention dynamic graph convolution network with cost-sensitive learning approach to road-level and minute-level traffic accident prediction". In: *IET Intelligent Transport Systems* 17.2 (Feb. 2023), pp. 270–284. ISSN: 17519578. DOI: 10.1049/itr2.12254.

[33] Yihong Ma et al. "Class-Imbalanced Learning on Graphs: A Survey". In: (Apr. 2023). URL: http://arxiv.org/abs/2304.04300.

[34] Yongxu Liu et al. "GATSMOTE: Improving Imbalanced Node Classification on Graphs via Attention and Homophily". In: *Mathematics* 10.11 (June 2022). ISSN: 22277390. DOI: 10.3390/math10111799.

[35] Joonhyung Park, Jaeyun Song, and Eunho Yang. "GraphENS: Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=MXEl7i-iru.

[36] Lirong Wu et al. "GraphMixup: Improving Class-Imbalanced Node Classification by Reinforcement Mixup and Self-supervised Context Prediction". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part IV*. Berlin, Heidelberg: Springer-Verlag, 2023, pp. 519–535. ISBN: 978-3-031-26411-5. DOI: 10.1007/978-3-031-26412-2{_}32. URL: https://doi.org/10.1007/978-3-031-26412-2_32.

[37] Liang Qu et al. "ImGAGN: Imbalanced Network Embedding via Generative Adversarial Graph Networks". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, Aug. 2021, pp. 1390–1398. ISBN: 9781450383325. DOI: 10.1145/3447548.3467334.

[38] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (2020), pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826.

[39] Yin Cui et al. "Class-Balanced Loss Based on Effective Number of Samples". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9260–9269. DOI: 10.1109/CVPR.2019.00949.

[40] Qi Dong, Shaogang Gong, and Xiatian Zhu. "Imbalanced Deep Learning by Minority Class Incremental Rectification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.6 (2019), pp. 1367–1381. DOI: 10.1109/TPAMI.2018.2832629.

[41] Deli Chen et al. "Topology-imbalance learning for semi-supervised node classification". In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2021. ISBN: 9781713845393.

[42] Jaeyun Song, Joonhyung Park, and Eunho Yang. "TAM: Topology-Aware Margin Loss for Class-Imbalanced Node Classification". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, Feb. 2022, pp. 20369–20383. URL: https://proceedings.mlr.press/v162/song22a.html.

[43] Kenza Amara et al. "GraphFramEx: Towards Systematic Evaluation of Explainability Methods for Graph Neural Networks". In: *Proceedings of the First Learning on Graphs Conference*. Ed. by Bastian Rieck and Razvan Pascanu. Vol. 198. Proceedings of Machine Learning Research. PMLR, Feb. 2022, 44:1–44:23. URL: https://proceedings.mlr.press/v198/amara22a.html.

[44] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *Workshop at International Conference on Learning Representations*. 2014.

[45] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Feb. 2017, pp. 3319–3328. URL: https://proceedings.mlr.press/v70/sundararajan17a.html.

[46] Ramprasaath R Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[47] Zhitao Ying et al. "GNNExplainer: Generating Explanations for Graph Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/d80b7040b773199015de6d3b4293c8ff-Paper.pdf.

[48] Dongsheng Luo et al. "Parameterized Explainer for Graph Neural Network". In: *Advances in Neural Information Processing Systems*. Ed. by H Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 19620–19631. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/e37b08dd3015330dcbb5d6663667b8b8-Paper.pdf.

[49] Hao Yuan et al. "On Explainability of Graph Neural Networks via Subgraph Explorations". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, Feb. 2021, pp. 12241–12252. URL: https://proceedings.mlr.press/v139/yuan21c.html.

[50] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. "Interpreting Graph Neural Networks for {NLP} With Differentiable Edge Masking". In: *International Con-*

ference on Learning Representations. 2021. URL: https://openreview.net/forum?id=WznmQa42ZAx.

[51] Minh Vu and My T Thai. "PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 12225–12235. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/8fb134f258b1f7865a6ab2d935a897c9-Paper.pdf.

[52] Fionn Murtagh. "Multilayer perceptrons for classification and regression". In: *Neurocomputing* 2.5 (1991), pp. 183–197. ISSN: 0925-2312. DOI: https://doi.org/10.1016/0925-2312(91)90023-5. URL: https://www.sciencedirect.com/science/article/pii/0925231291900235.

[53] Rune Elvik and Rahul Goel. "Safety-in-numbers: An updated meta-analysis of estimates". In: *Accident Analysis and Prevention* 129 (Aug. 2019), pp. 136–147. ISSN: 00014575. DOI: 10.1016/j.aap.2019.05.019.

Appendix

A Feature Table

Feature	Type	Description
NWB_ID	Text	Road segment ID
Day	Integer	Day of the week (1-7)
Hour	Integer	Hour of the day (0-24)
Daytime	Binary	Indicator if it is daytime (between 6:00 and 18:00)
Night_time	Binary	Indicator if it is nighttime (between 18:00 and 6:00)
Day_night_fct_	Text	Text indicating if it is daytime or nighttime
Crash_freq	Integer	Number of crashes on the road segment
Crash_fatal	Integer	Number of fatal crashes on the road segment
Crash_injury	Integer	Number of injury crashes on the road segment
Crash_binary	Binary	Indicator if there was any crash on the road segment
length_km	Continuous	Length of road segment
Bic_exp	Continuous	Volume of cyclists on the road segment
MV_exp	Continuous	Volume of motor vehicles on the road segment
Bic_exp_log	Continuous	Natural logarithm of "Bic_exp"
MV_exp_log	Continuous	Natural logarithm of "MV_exp"
Betweenness_norm	Continuous	road segment betweenness centrality score
Speed_50_separated	Binary	Indicator if the road segment is 50km/h separated
Speed_30_separated	Binary	Indicator if the road segment is 30km/h separated
Speed_50_on_road	Binary	Indicator if the road segment is 50km/h not separated
Speed_30_on_road	Binary	Indicator if the road segment is 30km/h not separated
Speed_infra_fct__	Text	Road type in text
Grade_separated	Binary	Grade separation of the road segment
Traf_light_dens_scaled	Continuous	Traffic light density
Roundabout_dens_scaled	Continuous	Roundabout density
Unsignalised_dens_scaled	Continuous	Unsignalized intersection density
Offices_150	Binary	Indicator if there is an office within 150 meters
Commercial_150	Binary	Indicator if there is a commercial facility within 150 meters
Railway_150	Binary	Indicator if there is a railway station within 150 meters
Educ_150	Binary	Indicator if there is an education facility within 150 meters
UTR_mun	Binary	Indicator of Utrecht
AMS_mun	Binary	Indicator of Amsterdam
ROT_mun	Binary	Indicator of Rotterdam
TH_mun	Binary	Indicator of The Hague
mun_fct_	Text	Municipality indicator

Table 8: Table of features

B GNNExplainer Background

Let ϕ be the function that indicates the GNN model. It learns a prediction based on the computation graph $G_c(v)$, and the node features $X_c(v)$ from the neighbouring nodes of v. The prediction is expressed as $\hat{t} = \phi(G_c(v), X_c(v))$. The goal is to find a small subgraph $G_s \subseteq G_c(v)$ that includes the most critical nodes and edges. We also want to find a subset $X_f \subseteq X_c(v)$ that includes the most important features. GNNExplainer does this by maximising the Mutual Information (MI) between the entropy $H(Y)$ and the conditional entropy $H(Y|G = G_S, X = X_S)$. It measures how much knowing the subgraph G_s and the subset X_f reduces the uncertainty about label Y:

$$\max_{G_s} \mathrm{MI}(Y, (G_S, X_S)) = H(Y) - H(Y|G = G_S, X = X_S). \tag{22}$$

Here, $H(Y)$ quantifies how uncertain the GNN is about the label Y before the graph is known.

$$H(Y) = -\sum_{c=1}^{C} P(Y = c) \log P(Y = c). \tag{23}$$

Here, $P(Y = c)$ is the predicted probability that node Y belongs to class c. The model computes a probability distribution over all possible classes C to compute the Entropy $H(Y)$. Since ϕ is fixed for a trained GNN, the entropy $H(Y)$ is constant. Therefore, maximising $\mathrm{MI}(Y, (G_S, X_S))$ equals minimizing $H(Y|G = G_S, X = X_S)$, which can be rewritten as:

$$H(Y|G = G_S, X = X_S) = -\mathbb{E}_{Y|G_S,X_S} \left[\log P_\Phi(Y|G = G_S, X = X_S)\right]. \tag{24}$$

G_S is constrained by $|G_S| \leq K_m$, so that it has at most K_m nodes. This ensures that G_s does not get too large.

Since there are many possible subgraphs G_S for G_C, let $\mathbf{A_S} \in [0, 1]^{n \times n}$ be the fractional adjacency matrix for the subgraph G_S and enforce the subgraph constraint as:

$$\mathbf{A_S}[j, k] \leq \mathbf{A_c}[j, k] \quad \text{for all } j, k. \tag{25}$$

This continuous relaxation allows the edge weights to be between 0 and 1, unlike hard binary choices like 0 or 1. The constraint makes sure that there are no edges that do not exist in the original graph exist in the subgraph. The continuous relaxation can be seen as a variational approximation of the distribution of the subgraphs G_C. Now Equation (24) turns into:

$$\min_{G} \mathbb{E}_{G_S \sim G} \left[H(Y \mid G = G_S, X = X_S)\right]. \tag{26}$$

To estimate $\mathbb{E}[G_S]$, GNNExplainer applies a mean-field variational approximation, where the graph G is decomposed into a multivariate Bernoulli distribution over edges:

$$P_G(G_S) = \prod_{(j,k) \in G_c} A_S[j, k]. \tag{27}$$

In short, we want to detect the probability that an edge is in the subgraph. To optimise the conditional entropy, a mask is applied on the computation graph of the adjacency matrix by $A_c \odot \sigma(M)$, where $M \in \mathbb{R}^{n \times n}$ is the learnable mask matrix and σ is the sigmoid function mapping al values to $[0, 1]^{n \times n}$

If we want to explain why a certain class label is predicted, the objective function can be changed to:

$$\min -\sum_{c=1}^{C} \mathbf{1}[y=c] \log P_{\Phi}(Y=y \mid G=A_c \odot \sigma(M), X=X_c). \tag{28}$$

This is better for understanding why the model has chosen a certain class.

Furthermore, GNNExplainer defines X_S^F as a subset of features of nodes in G_S. GNNExplainers learns a binary feature selector $F \in \{0,1\}^d$ for the nodes in X_S.

$$X_S^F = \{x_j^F \mid v_j \in G_S\}, \quad x_j^F = [x_{j,t_1}, \ldots, x_{j,t_k}] \text{ for } F_{t_i} = 1 \tag{29}$$

Thereafter, the explanation (G_S, X_S) is jointly optimized by:

$$\max_{G_S,F} \mathrm{MI}(Y, (G_S, F)) = H(Y) - H(Y \mid G = G_S, X = X_S^F) \tag{30}$$

The features subset X_S^F is obtained by:

$$X_S^F = X_S \odot F. \tag{31}$$

Here, F is the feature mask that needs to be learned. Lastly, a parametrisation trick is used so the gradients of Equation (30) can be backpropagate to F, so F can be optimised through gradient descent. The parametrization trick is given by:

$$X = Z + (X_S - Z) \odot F \quad \text{subject to} \quad \sum_j F_j \leq K_F \tag{32}$$

Here, $Z \in \mathbb{R}^d$ is a random variable sampled from the empirical distribution of the features, and K_F is a parameter that controls the maximum number of features that can be kept in the explanation. This ensures that the explanation is concise and includes only the most important features.

C GNN Explainer Boxplots

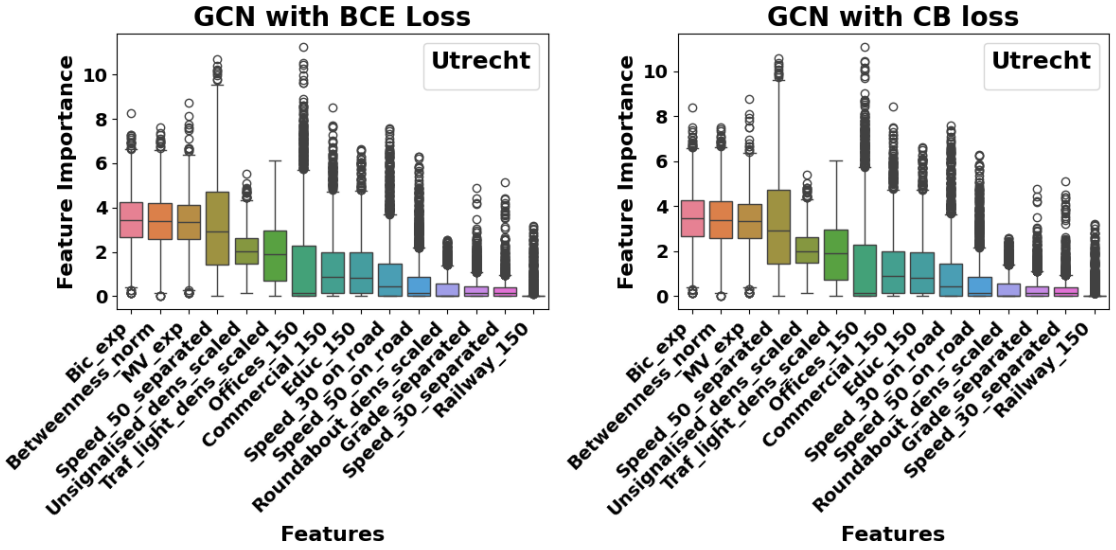

FIGURE 22: Boxplots of the GCN Model with BCE Loss vs the GCN Model with CB Loss for Utrecht

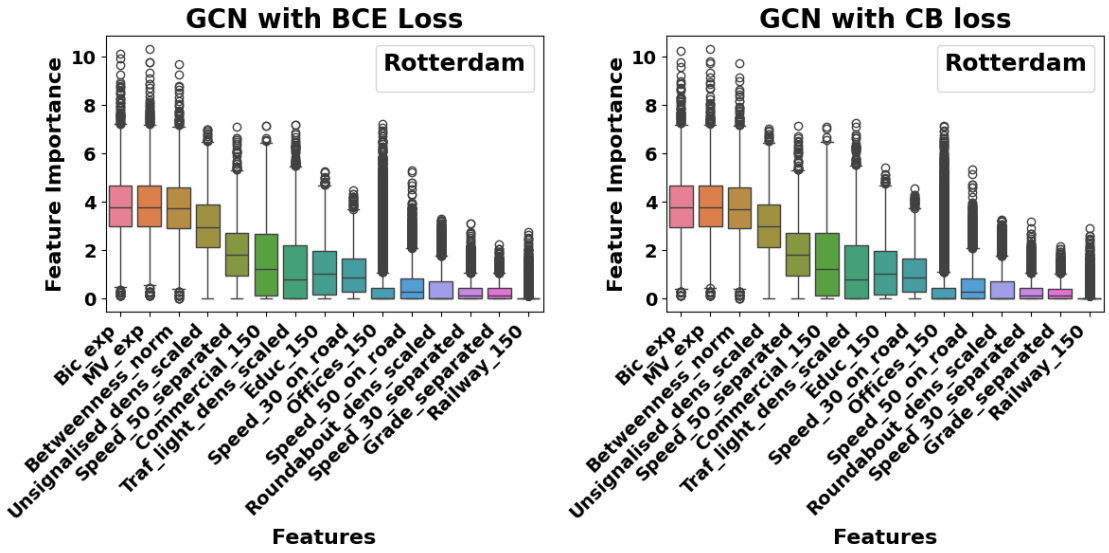

FIGURE 23: Boxplots of the GCN Model with BCE Loss vs the GCN Model with CB Loss for Rotterdam

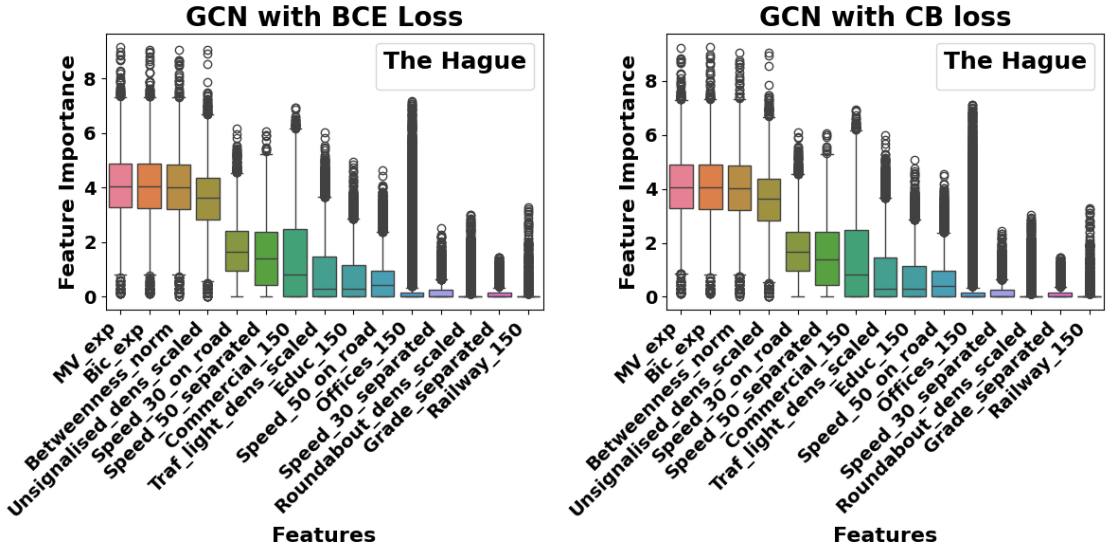

FIGURE 24: Boxplots of the GCN Model with BCE Loss vs the GCN Model with CB Loss for The Hague

D Partial Dependence Plots

Utrecht

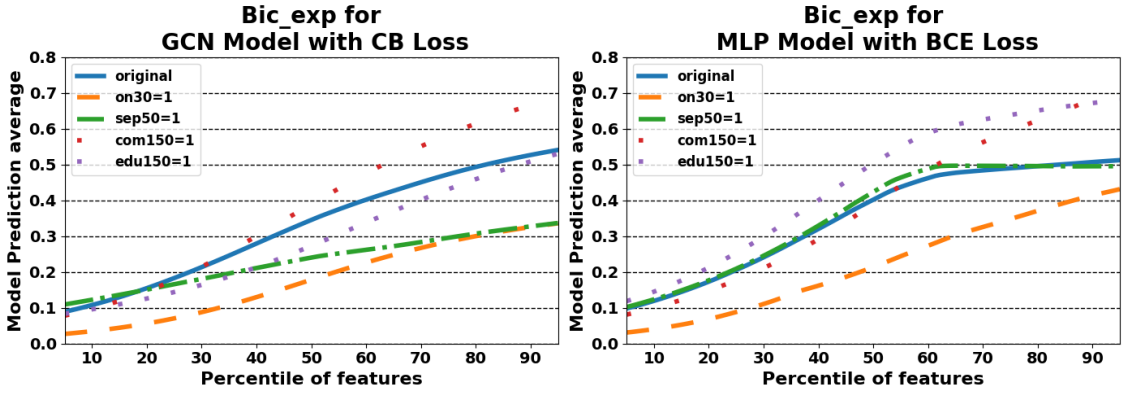

FIGURE 25: Partial Dependence plots for feature *Bic_exp* in Utrecht

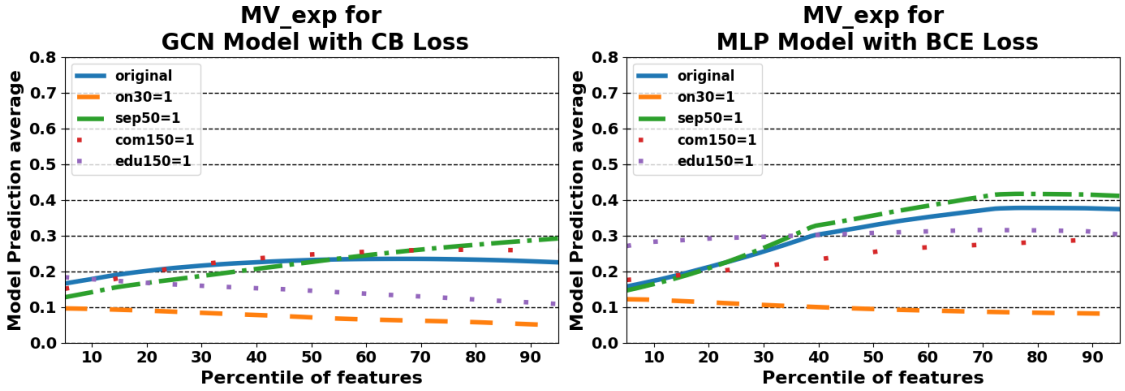

FIGURE 26: Partial Dependence plots for feature *MV_exp* in Utrecht

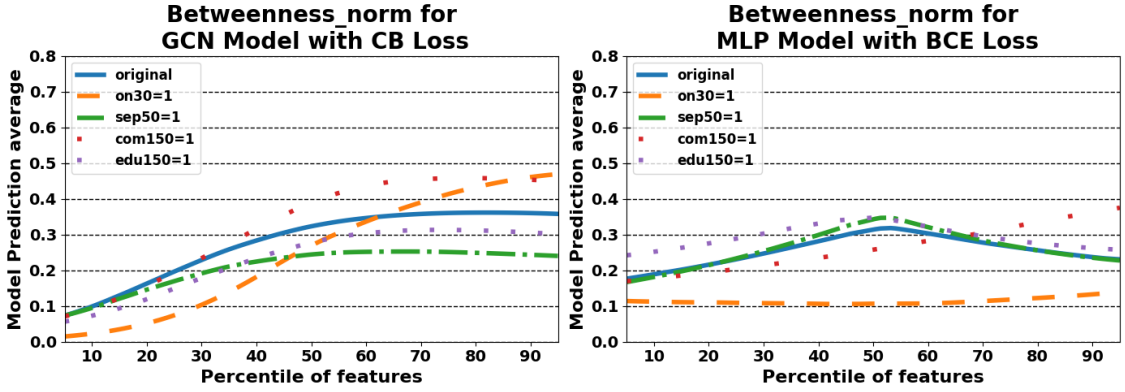

FIGURE 27: Partial Dependence plots for feature *Betweenness_norm* in Utrecht

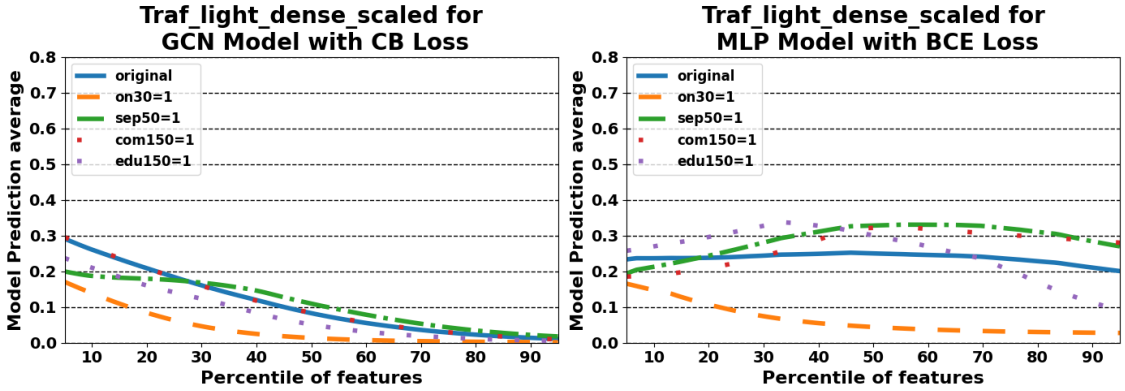

FIGURE 28: Partial Dependence plots for feature *Traf_light_dens_scaled* in Utrecht

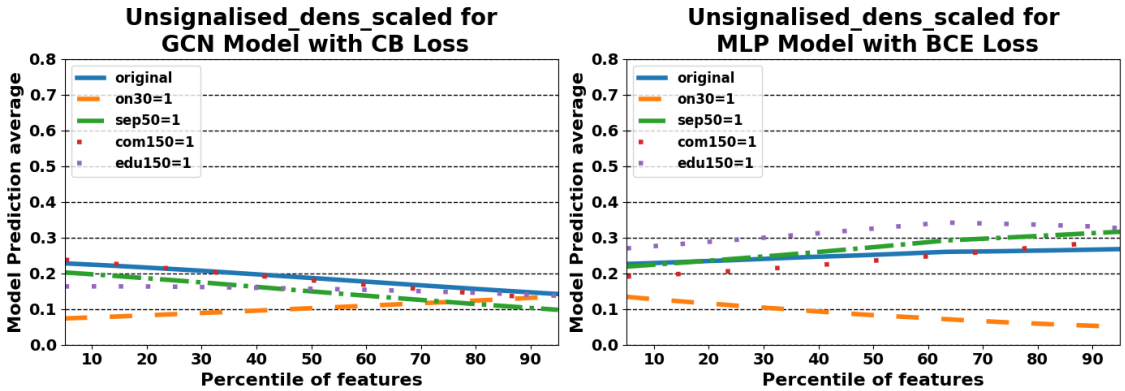

FIGURE 29: Partial Dependence plots for feature *Unsignalised_dens_scaled* in Utrecht

Rotterdam

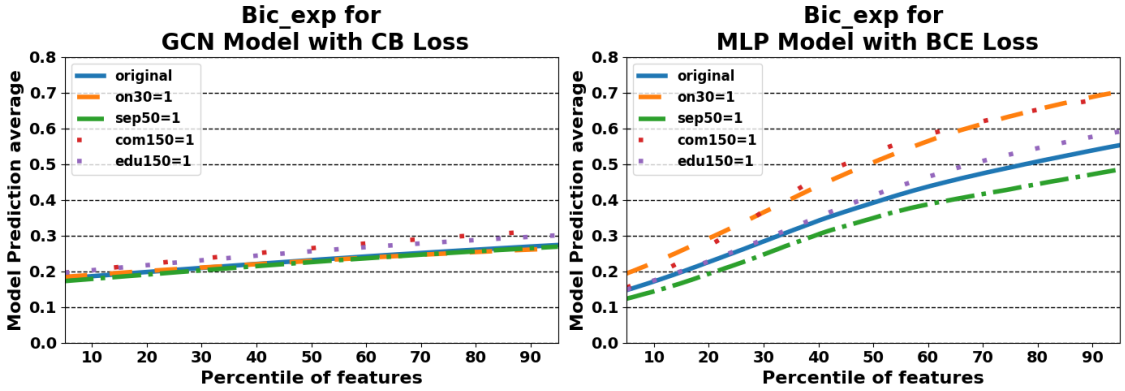

FIGURE 30: Partial Dependence plots for feature *Bic_exp* in Rotterdam

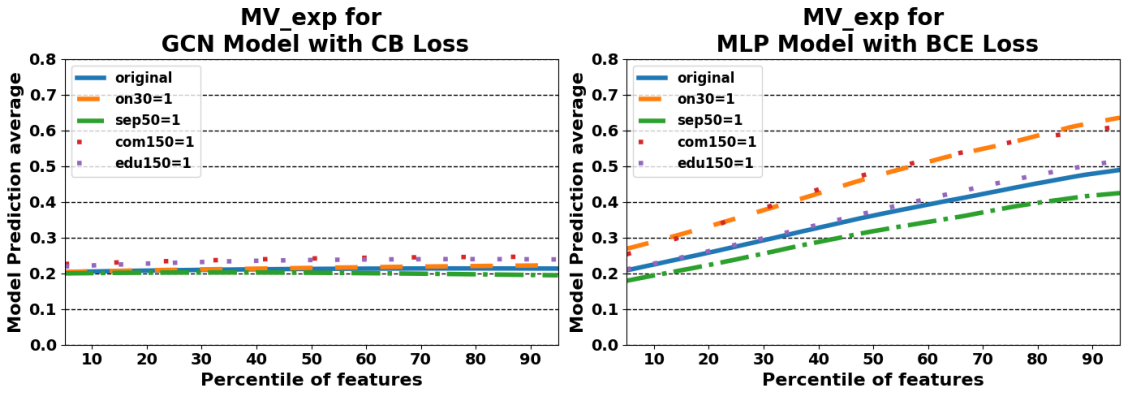

FIGURE 31: Partial Dependence plots for feature *MV_exp* in Rotterdam

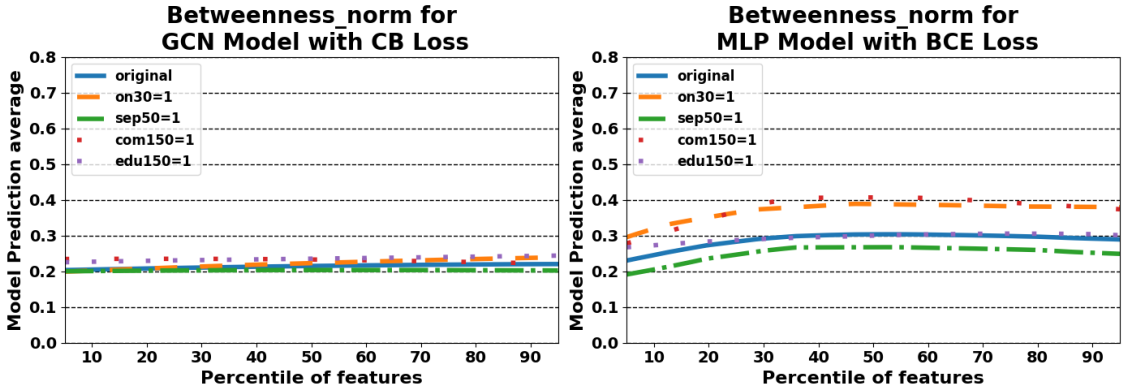

FIGURE 32: Partial Dependence plots for feature *Betweenness_norm* in Rotterdam

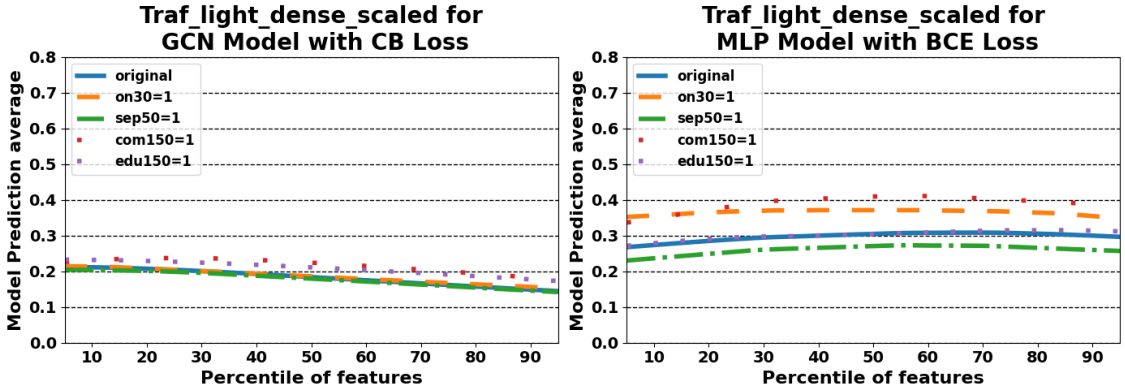

FIGURE 33: Partial Dependence plots for feature *Traf_light_dens_scaled* in Rotterdam

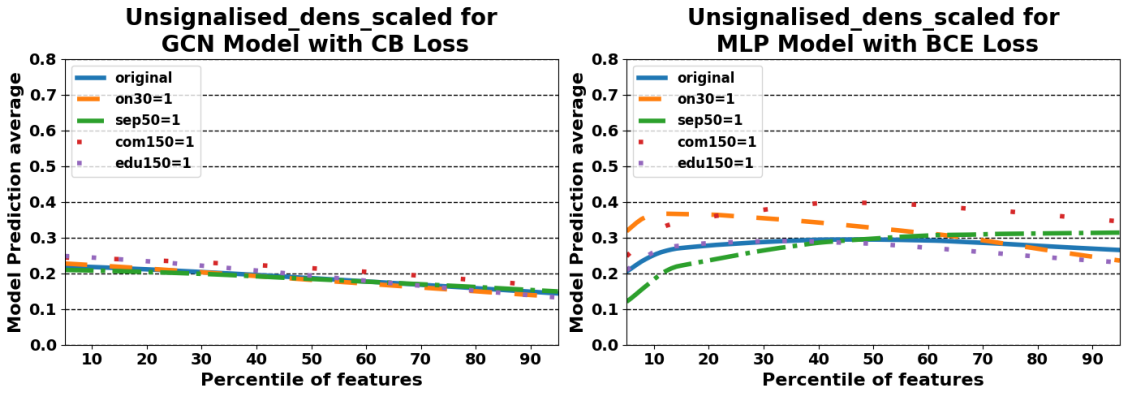

FIGURE 34: Partial Dependence plots for feature *Unsignalised_dens_scaled* in Rotterdam

The Hague

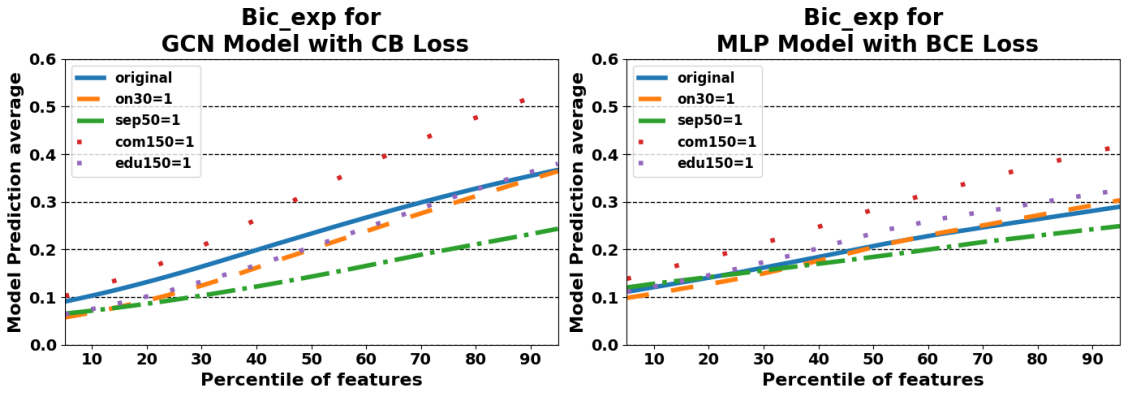

FIGURE 35: Partial Dependence plots for feature *Bic_exp* in The Hague

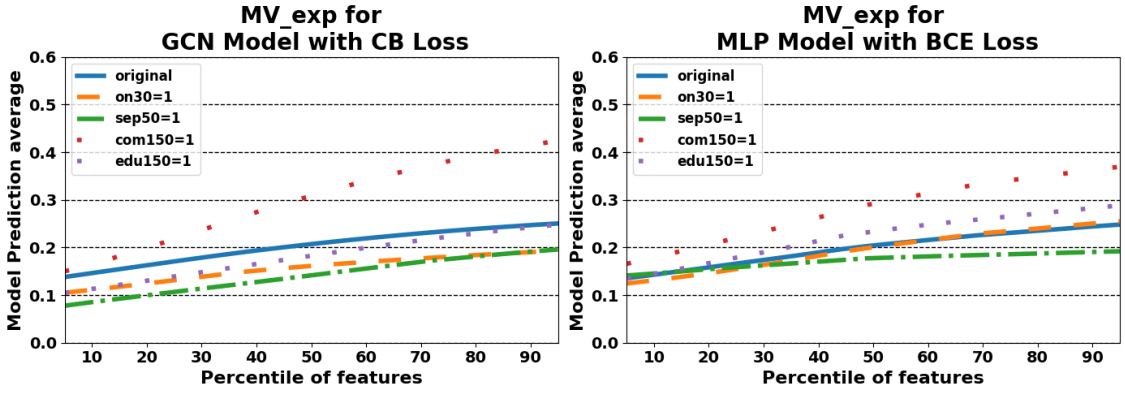

FIGURE 36: Partial Dependence plots for feature *MV_exp* in The Hague

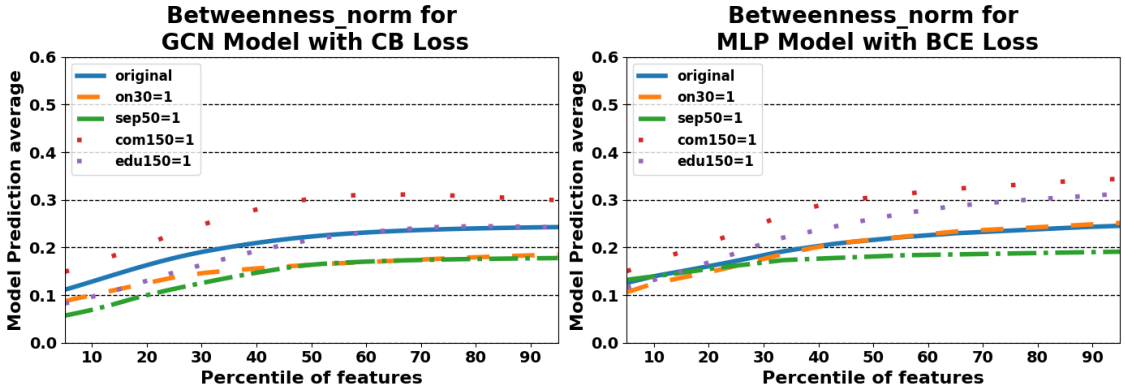

FIGURE 37: Partial Dependence plots for feature *Betweenness_norm* in The Hague

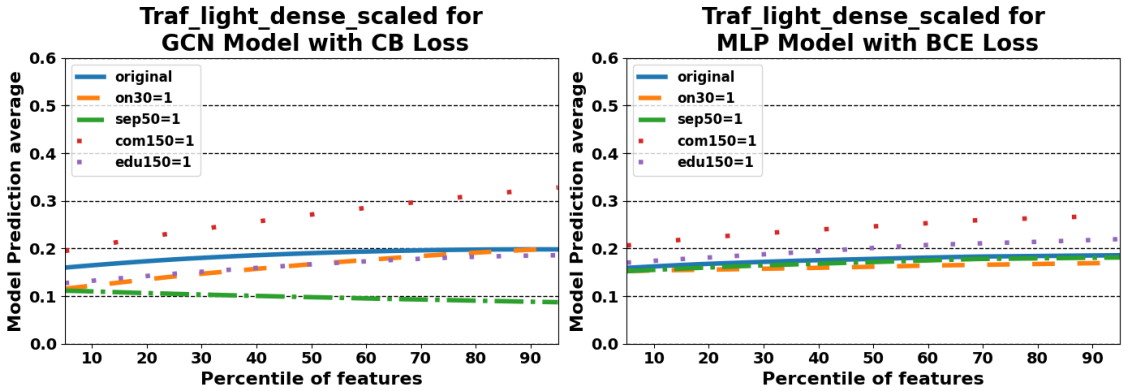

FIGURE 38: Partial Dependence plots for feature *Traf_light_dens_scaled* in The Hague

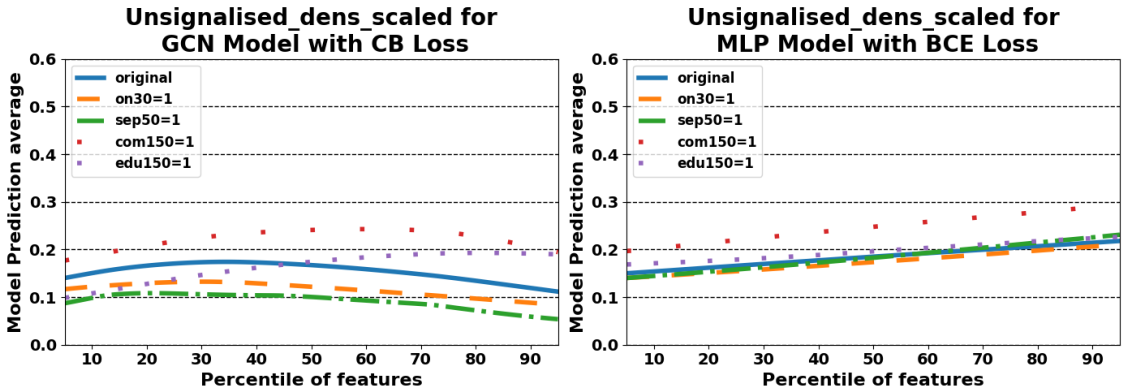

FIGURE 39: Partial Dependence plots for feature *Unsignalised_dens_scaled* in The Hague