Supplementary Information for:

Spatiotemporal prediction of unsafe road conditions using weather and infrastructure proximity information

A comparative study across model learning paradigms

Nirmal Sitaldin*, Bart van Arem*, Maaike Snelder*, Shadi Sharif Azadeh*
*Civil Engineering & Geosciences, Department of Transport & Planning , Delft University of Technology

Online Resource 1: Supplementary tables and figures

This document provides supplementary material for the manuscript, including data composition tables, class-support tables, hyperparameter search spaces and selected configurations, confusion matrices, and probability heat maps. Tables and figures in this document are numbered separately from those in the main manuscript using the prefixes S.

S1 Dataset composition before and after bootstrapping

This section presents the distribution of unsafe road condition (URC) related incidents in the orignal dataset, and after clustering and bootstrapping. It serves as additional supportive information for Section 5.1.2 Bootstrapping.

Table S1: URC class distribution among clusters (original dataset)

URC	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Infrastructure damage	133 (17.5%)	151 (14.3%)	120 (9.4%)	305 (24.7%)	60 (18.6%)
Other	47 (6.2%)	120 (11.3%)	102 (8.0%)	80 (6.5%)	22 (6.8%)
Road ponding	226 (29.7%)	447 (42.2%)	775 (60.5%)	498 (40.3%)	183 (56.8%)
Roadside fires	311 (40.9%)	218 (20.6%)	231 (18.0%)	285 (23.1%)	50 (15.5%)
Slippery pavement	43 (5.7%)	124 (11.7%)	53 (4.2%)	67 (5.4%)	7 (2.2%)

Table S2: URC class distribution across bootstrapped clusters

URC	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Infrastructure damage	366 (16.05%)	437 (13.74%)	375 (9.76%)	900 (24.29%)	173 (17.30%)
Other	136 (5.96%)	372 (11.70%)	291 (7.57%)	242 (6.53%)	71 (7.10%)
Road ponding	682 (29.91%)	1354 (42.58%)	2347 (61.07%)	1495 (40.35%)	564 (56.40%)
Roadside fire	969 (42.50%)	643 (20.22%)	668 (17.38%)	868 (23.43%)	164 (16.40%)
Slippery pavement	127 (5.58%)	374 (11.76%)	162 (4.22%)	200 (5.40%)	28 (2.80%)

S2 Class-support tables

This section reports the per-URC-class support on the test sets of the bootstrapped clusters and original data used in the model evaluation. These tables support the discussion of class imbalance in the main manuscript

in Sections 6.1.1 Overall predictive performance and 6.2 Ablation studies.

Table S3: URC class support on test set - Cluster 1

URC type	Support
Infrastructure damage	90
Other	43
Road ponding	127
Roadside fire	199
Slippery pavement	35

Table S4: URC class support on test set - Cluster 2

URC type	Support
Infrastructure damage	158
Other	114
Road ponding	202
Roadside fire	95
Slippery pavement	109

Table S5: URC class support on test set - Cluster 3

URC type	Support
Infrastructure damage	108
Other	70
Road ponding	340
Roadside fire	166
Slippery pavement	69

Table S6: URC class support on test set - Cluster 4

URC type	Support
Infrastructure damage	349
Other	118
Road ponding	379
Roadside fire	262
Slippery pavement	57

Table S7: URC class support on test set - Cluster 5

URC type	Support
Infrastructure damage	28
Other	20
Road ponding	101
Roadside fire	22
Slippery pavement	1

Table S8: URC class support on test set - original data

URC type	Support
Infrastructure damage	255
Other	109
Road ponding	389
Roadside fire	250
Slippery pavement	86

S3 Hyperparameter search spaces and selected configurations

This section reports the justification for the chosen hyperparameter search spaces used during Bayesian Optimization, together with the selected model configurations, and serves as additional support to Section 5.5 Hyperparameter tuning.

For all deep learning models, LeakyReLU is used as the nonlinear activation function in the hidden layers where an explicit activation function is required. Compared with the standard ReLU activation, LeakyReLU allows a small non-zero gradient for negative input values, reducing the risk of inactive neurons and supporting more stable gradient propagation during training [1], [2]. This is useful for the sparse and imbalanced URC incident data considered in this study, where overly sparse activations may limit the ability of the models to learn informative representations. The Adam optimizer is used for all deep learning models because it adaptively estimates first- and second-order moments of the gradients, making it computationally efficient and well suited for stochastic optimization problems with noisy or sparse gradients [3]. Its adaptive learning-rate behavior is particularly suitable for comparing temporal, spatial, and spatiotemporal neural architectures under a shared optimization strategy.
The learning rate is sampled from the range $[10^{-4}, 10^{-2}]$ on a logarithmic scale. This parameter controls the step size of the optimizer during model training, where smaller values lead to more gradual updates and larger values allow faster but potentially less stable optimization. The L2 regularization coefficient is sampled from the range $[10^{-4}, 10^{-1}]$ on a logarithmic scale. This parameter controls the strength of the weight penalty applied during training, where larger values impose stronger regularization and can help reduce overfitting.
For the standalone LSTM and BLSTM models, explicit L2 regularization is not included in the hyperparameter search space. Regularization for these temporal models was instead controlled through dropout and validation-based model selection, as dropout is widely used to reduce overfitting in neural networks and has been specifically adapted for recurrent architectures [4], [5]. This choice also kept the Bayesian optimization search space compact by limiting the number of interacting hyperparameters to be evaluated [6], [7]. For the GCN-BLSTM and GAT-BLSTM models, L2 regularization is tuned only for the graph-based components, because these layers introduce additional trainable transformations during neighborhood aggregation before the temporal representation is learned. Initial tuning experiments showed that adding dropout to the combined spatiotemporal models did not lead to consistent validation-performance improvements compared with using L2 regularization for the graph components only. Dropout was therefore excluded from the final Bayesian optimization search space for these combined models to reduce unnecessary hyperparameter interactions. The recurrent components of all temporal and spatiotemporal models were therefore treated consistently, while explicit L2 regularization was reserved for the graph feature extraction layers. For the conventional machine learning models, the hyperparameter search spaces were selected to control model capacity, regularization strength, and generalization performance. For RF, the number of estimators, maximum tree depth, minimum samples required for node splitting, and minimum samples per leaf were tuned. The number of estimators controls the size of the ensemble, while the depth and node-splitting parameters regulate the complexity of the individual decision trees and reduce the risk of overfitting. These parameters are commonly used to balance variance reduction and model complexity in RF models [8].
LR was implemented using stochastic gradient descent (SGD), allowing the optimization procedure to tune both the regularization strategy and the learning dynamics. The penalty term was selected from L1, L2, and elastic net regularization, while the regularization coefficient was sampled on a logarithmic scale. L1 regularization can promote sparse feature weights, L2 regularization penalizes large coefficients more smoothly, and

elastic net combines both penalties through the L1 ratio [9], [10]. In addition, the learning-rate schedule and initial learning rate were tuned to control the update behavior of the SGD optimizer, which is important for stable convergence in gradient-based linear classification models [11].
For XGB, the search space was designed to control both the size and regularization behavior of the boosted tree ensemble. The number of estimators determines the number of boosting rounds, while the maximum tree depth controls the complexity of the individual trees. The learning rate, sampled on a logarithmic scale, determines the contribution of each newly added tree and therefore governs the trade-off between fast learning and stable generalization. The subsample and column-sampling ratios were also tuned to introduce randomness during boosting, which can reduce overfitting by limiting the observations and features available to each tree [12], [13]. The evaluation metric was set to multiclass log loss to align hyperparameter selection with the probabilistic multi-class prediction objective of this study.

Tables below present the optimal set of hyperparameters and the associated peak performance on the validation set across all models and datasets (clusters & bootstrapped, and on the original dataset).

Table S9: Optimal hyperparameter set on bootstrapped clusters – Random Forest

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Number of estimators	150	250	150	200	100
Maximum tree depth	3	6	3	13	5
Minimum samples split	2	10	5	4	5
Minimum samples leaf	3	6	2	10	5
Validation CCE loss	1.33	0.82	0.93	1.04	1.08
Validation accuracy	0.50	0.72	0.63	0.54	0.48

Table S10: Optimal hyperparameter set on bootstrapped clusters – Logistic Regression

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Penalty	Elastic net	Elastic net	L1	L1	L1
Regularization coefficient	0.00003	0.00029	0.000001	0.00003	0.000009
L1 ratio	0.82	0.98	n.a.	n.a.	n.a.
Learning-rate schedule	Adaptive	Adaptive	Constant	Adaptive	Adaptive
Initial learning rate	0.0007	0.0865	0.0868	0.0980	0.0995
Validation CCE loss	1.29	0.82	0.76	1.09	1.02
Validation accuracy	0.53	0.70	0.69	0.47	0.53

Table S11: Optimal hyperparameter set on bootstrapped clusters – XGB

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Number of estimators	100	500	500	100	250
Maximum tree depth	4	4	10	5	5
Learning rate	0.016	0.007	0.003	0.056	0.013
Subsample ratio	0.86	0.90	0.69	0.85	0.75
Column sample by tree	0.99	0.87	0.74	0.88	0.91
Validation CCE loss	1.32	0.80	0.92	1.09	1.13
Validation accuracy	0.53	0.69	0.70	0.53	0.54

Table S12: Optimal hyperparameter set on bootstrapped clusters – GCN

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
GCN layer 1[1]	32	64	64	32	64
GCN layer 2[1]	16	64	64	64	8
L2 regularization coefficient	0.0002	0.0001	0.0002	0.0008	0.0001
Learning rate	0.009	0.009	0.005	0.004	0.010
Dropout rate	0.30	0.12	0.68	0.41	0.47
Validation CCE loss	0.11	0.18	0.11	0.14	0.10
Validation accuracy	0.60	0.53	0.63	0.45	0.56

[1]GCN layers 1 and 2 denote the number of hidden units in the first and second graph convolutional layers, respectively.

Table S13: Optimal hyperparameter set on bootstrapped clusters – GAT

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
GAT layer 1[1]	16	16	64	64	32
GAT layer 2	16	32	8	16	32
Attention heads 1	1	3	1	3	2
Attention heads 2	1	2	3	3	1
L2 regularization coefficient	0.0001	0.0001	0.0004	0.0002	0.0001
Learning rate	0.009	0.004	0.010	0.005	0.010
Dropout rate	0.27	0.02	0.26	0.54	0.59
Validation CCE loss	0.14	0.16	0.12	0.15	0.12
Validation accuracy	0.60	0.53	0.61	0.45	0.56

[1]GAT layers 1 and 2 denote the number of hidden units in the first and second graph attention layers, respectively. Attention heads 1 and 2 denote the number of attention heads used in the corresponding GAT layers.

Table S14: Optimal hyperparameter set on bootstrapped clusters – LSTM

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
LSTM layer 1[1]	16	8	8	64	32
LSTM layer 2[1]	8	16	16	32	32
Learning rate	0.006	0.007	0.007	0.01	0.002
Dropout rate	0.28	0.36	0.24	0.28	0.30
Validation accuracy	1.37	0.99	0.88	1.17	1.15
Validation CCE loss	0.47	0.64	0.68	0.53	0.49

[1]LSTM layers 1 and 2 denote the number of LSTM units in the first and second recurrent layers, respectively.

Table S15: Optimal hyperparameter set on bootstrapped clusters – BLSTM

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
BLSTM layer 1[1]	16	32	16	64	8
BLSTM layer 2	64	16	64	16	16
Learning rate	0.005	0.010	0.010	0.009	0.010
Dropout rate	0.48	0.25	0.30	0.25	0.29
Validation CCE loss	1.39	0.97	0.89	1.16	1.14
Validation accuracy	0.46	0.65	0.71	0.49	0.47

[1]BLSTM layers 1 and 2 denote the number of LSTM units in the first and second bidirectional recurrent layers, respectively.

Table S16: Optimal hyperparameter set on bootstrapped clusters – GCN-BLSTM

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
GCN layer 1[1]	16	16	8	16	16
GCN layer 2	16	32	8	8	32
BLSTM layer 1	64	32	32	16	64
BLSTM layer 2	64	8	64	32	64
L2 regularization coefficient	0.0002	0.0002	0.0004	0.0004	0.0013
Learning rate	0.002	0.005	0.005	0.008	0.010
Validation CCE loss	1.42	1.04	0.91	1.21	1.00
Validation accuracy	0.46	0.61	0.71	0.49	0.60

[1]GCN layers 1 and 2 denote the number of hidden units in the first and second graph convolutional layers, respectively. BLSTM layers 1 and 2 denote the number of LSTM units in the first and second bidirectional recurrent layers, respectively. The L2 regularization coefficient is applied only to the graph-based components.

Table S17: Optimal hyperparameter set on bootstrapped clusters – GAT-BLSTM

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
GAT layer 1[1]	32	8	16	16	32
GAT layer 2	8	8	64	32	32
Attention heads 1	2	3	3	2	2
Attention heads 2	2	2	3	3	1
BLSTM layer 1	8	16	64	64	8
BLSTM layer 2	32	32	32	16	16
L2 regularization coefficient	0.0059	0.0010	0.0011	0.0013	0.0108
Learning rate	0.002	0.003	0.001	0.002	0.004
Validation CCE loss	1.54	1.07	0.97	1.27	1.07
Validation accuracy	0.38	0.62	0.66	0.44	0.57

[1]GAT layers 1 and 2 denote the number of hidden units in the first and second graph attention layers, respectively. Attention heads 1 and 2 denote the number of attention heads used in the corresponding GAT layers. BLSTM layers 1 and 2 denote the number of LSTM units in the first and second bidirectional recurrent layers, respectively. The L2 regularization coefficient is applied only to the graph-based components.

S4 Confusion matrices

This section presents the confusion matrices to support model evaluation discussed in Section 6.1 URC prediction on bootstrapped clusters.

S4.1 Random Forest

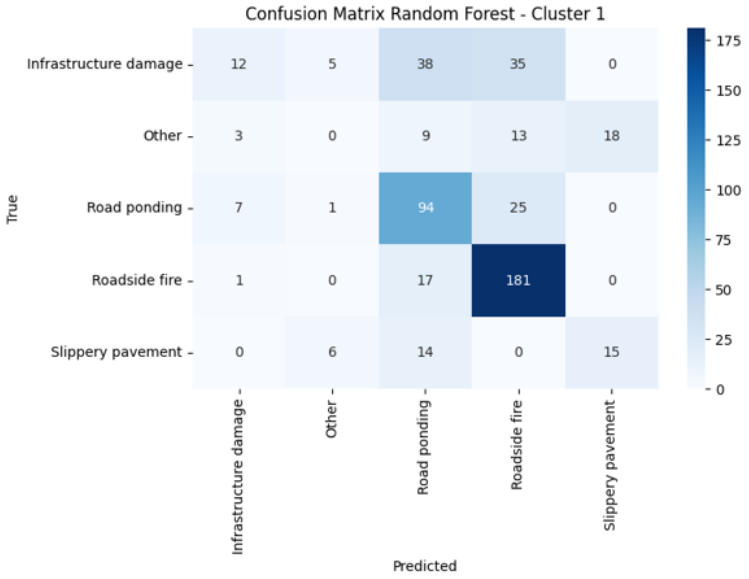

Figure S1: Confusion Matrix RF – Cluster 1

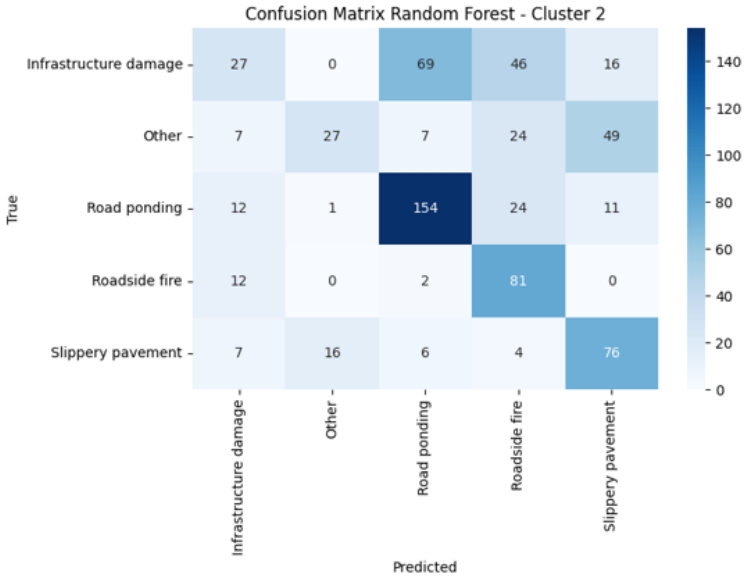

Figure S2: Confusion Matrix RF – Cluster 2

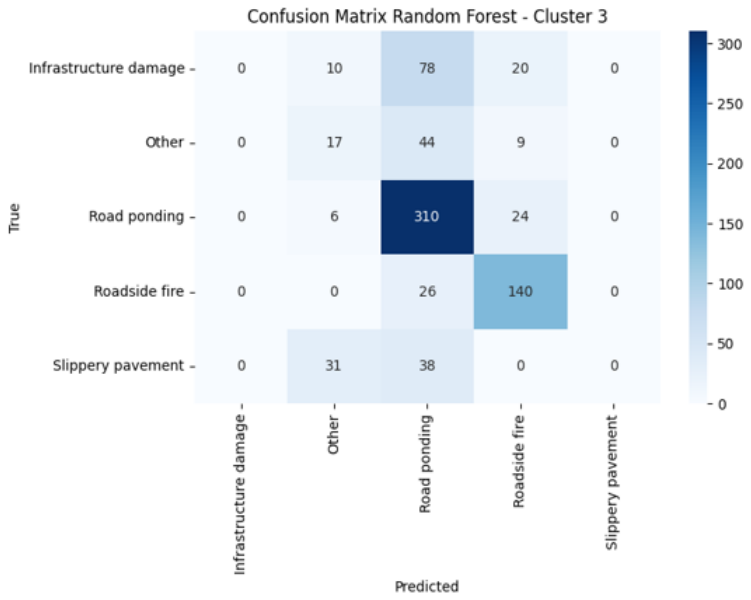

Figure S3: Confusion Matrix RF – Cluster 3

S4.2 Extreme Gradient Boosting

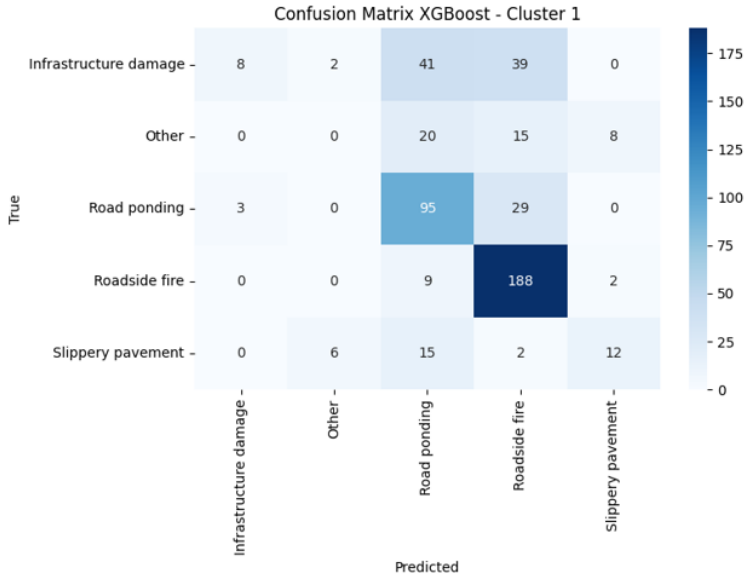

Figure S6: Confusion Matrix XGB – Cluster 1

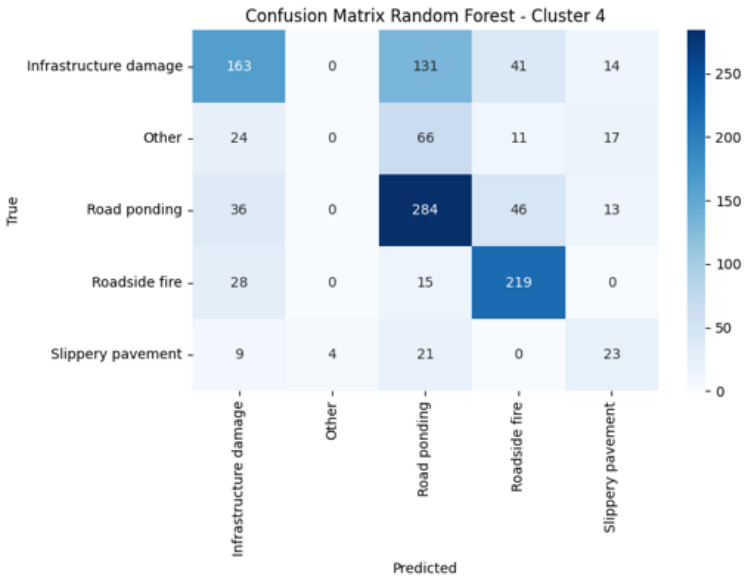

Figure S4: Confusion Matrix RF – Cluster 4

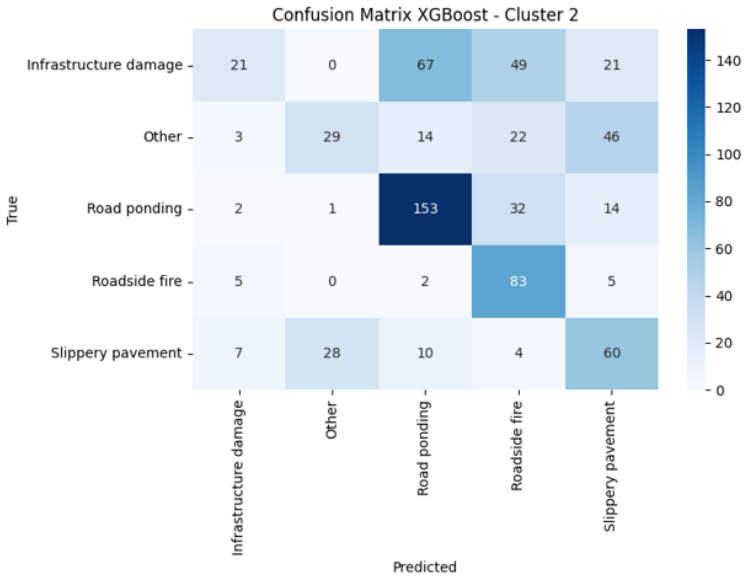

Figure S7: Confusion Matrix XGB – Cluster 2

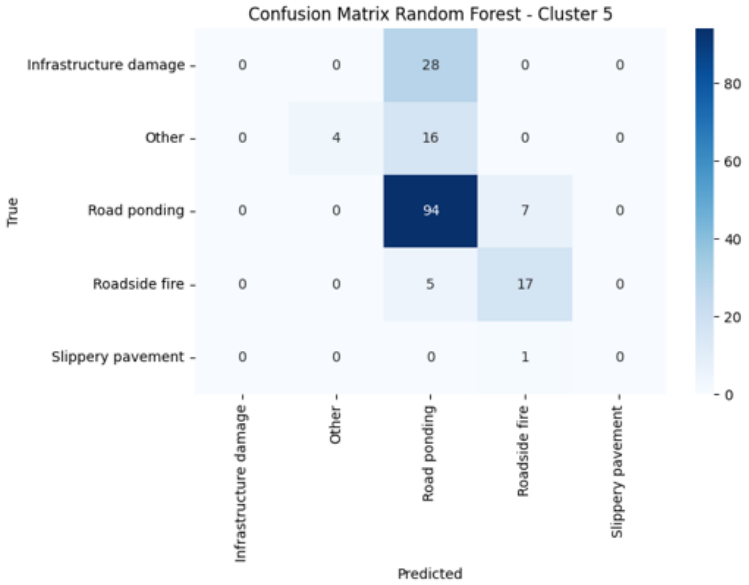

Figure S5: Confusion Matrix RF – Cluster 5

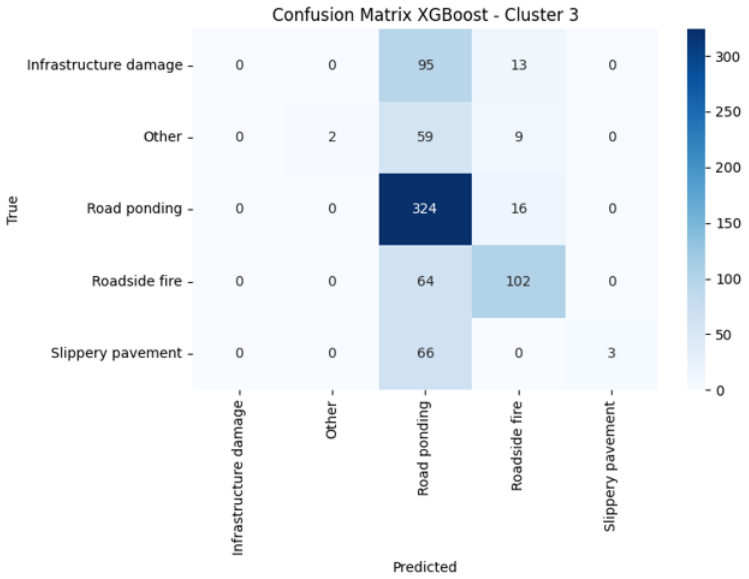

Figure S8: Confusion Matrix XGB – Cluster 3

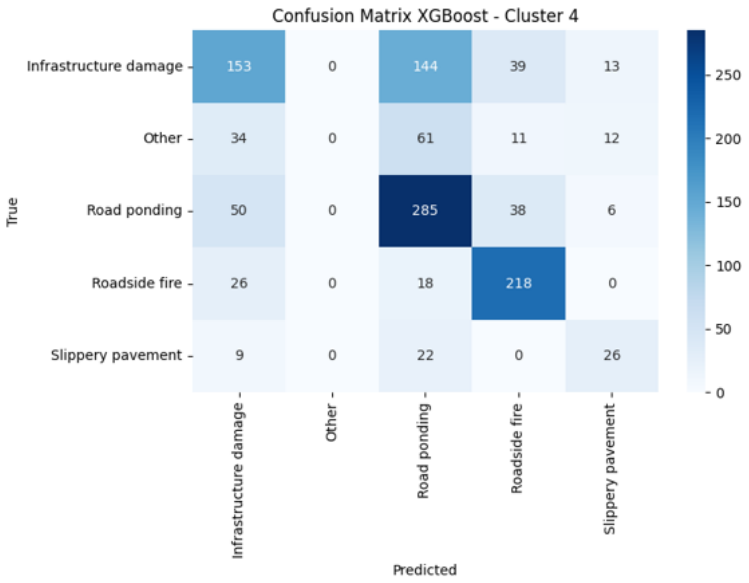

Figure S9: Confusion Matrix XGB – Cluster 4

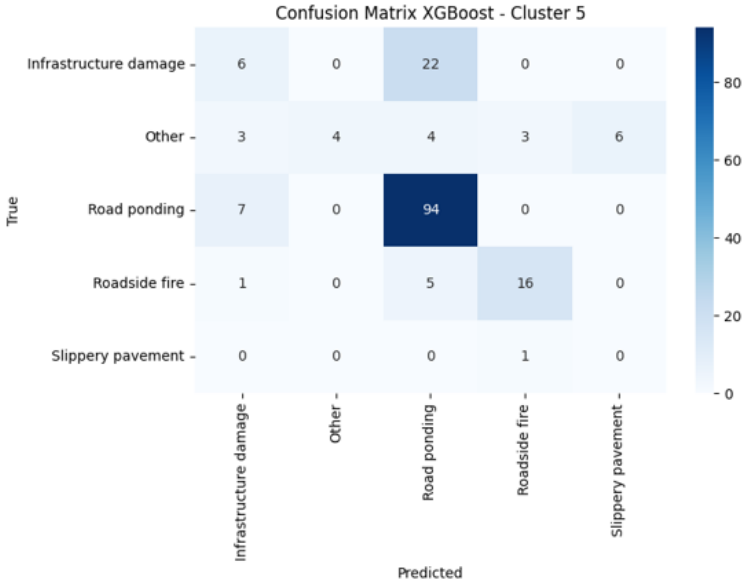

Figure S10: Confusion Matrix XGB – Cluster 5

S4.3 Logistic Regression

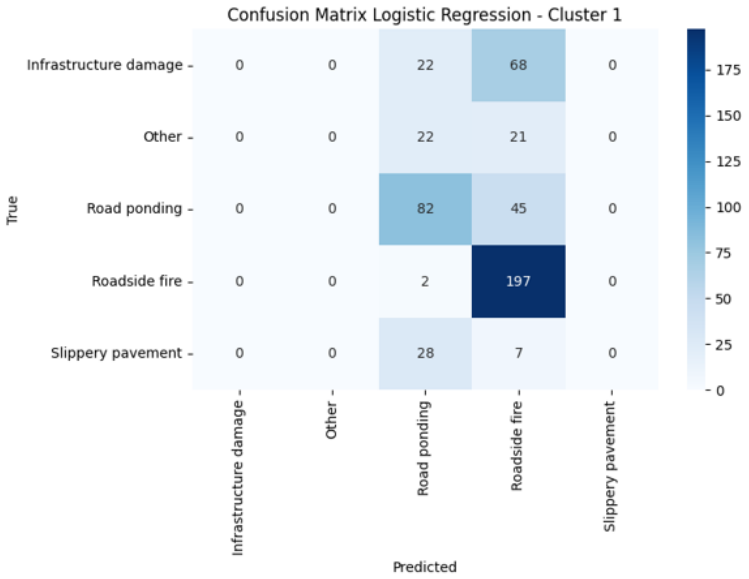

Figure S11: Confusion Matrix LR – Cluster 1

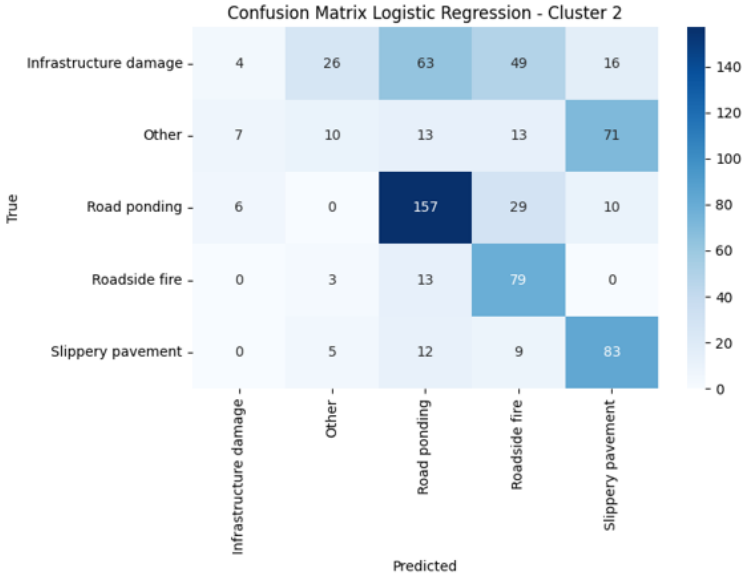

Figure S12: Confusion Matrix LR – Cluster 2

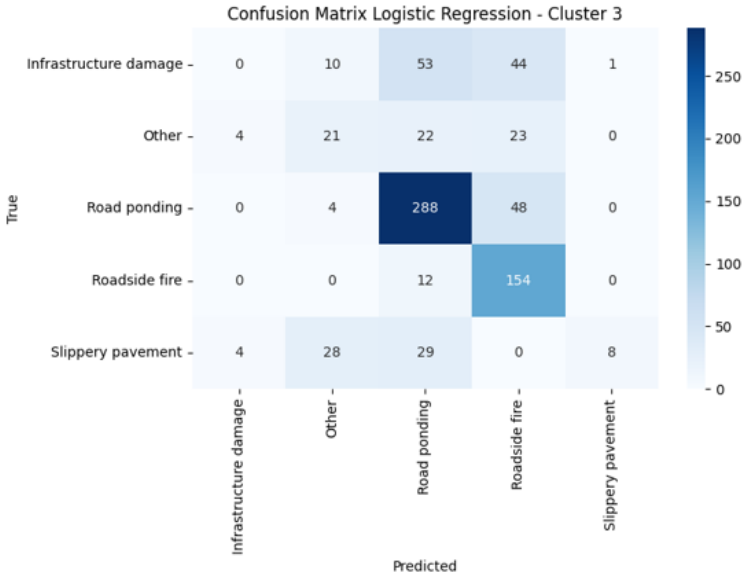

Figure S13: Confusion Matrix LR – Cluster 3

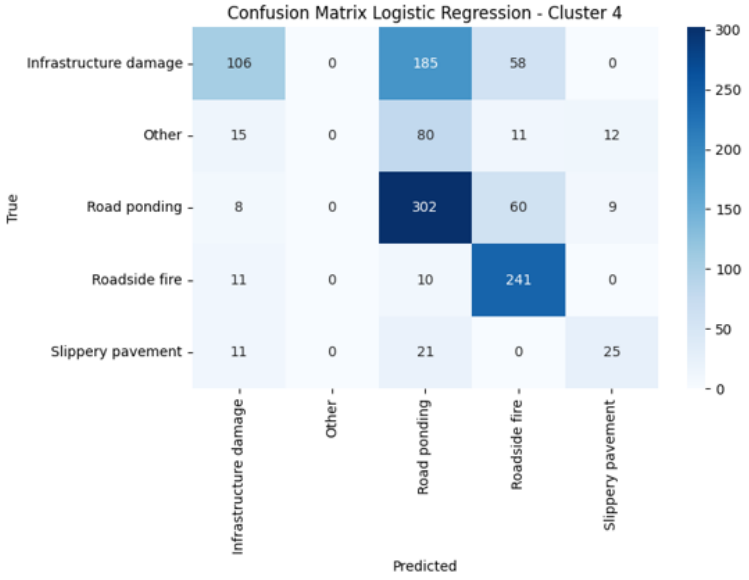

Figure S14: Confusion Matrix LR – Cluster 4

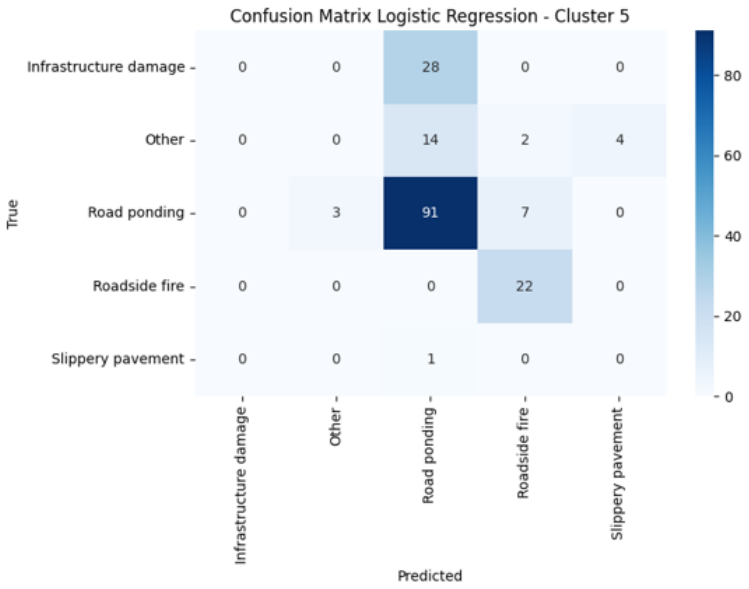

Figure S15: Confusion Matrix LR – Cluster 5

S4.4 Long-Short Term Memory Network

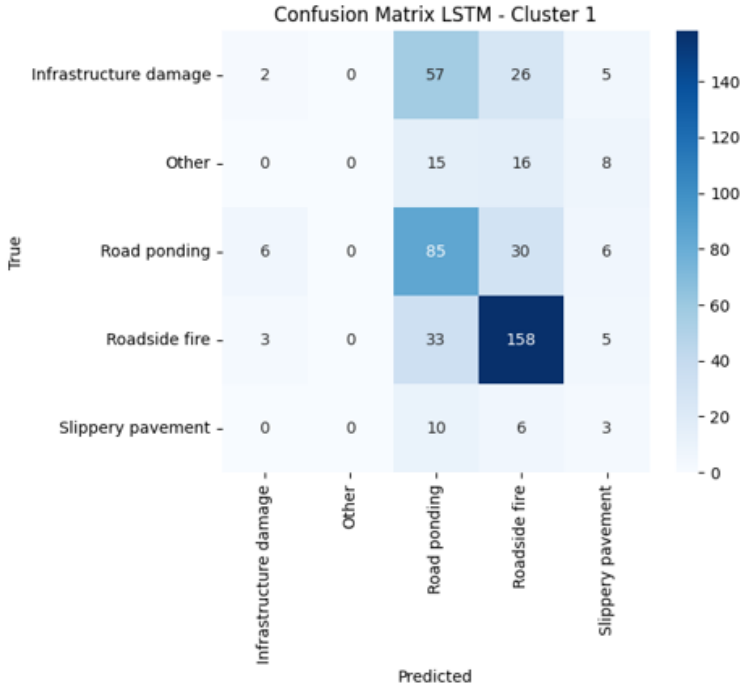

Figure S16: Confusion Matrix LSTM – Cluster 1

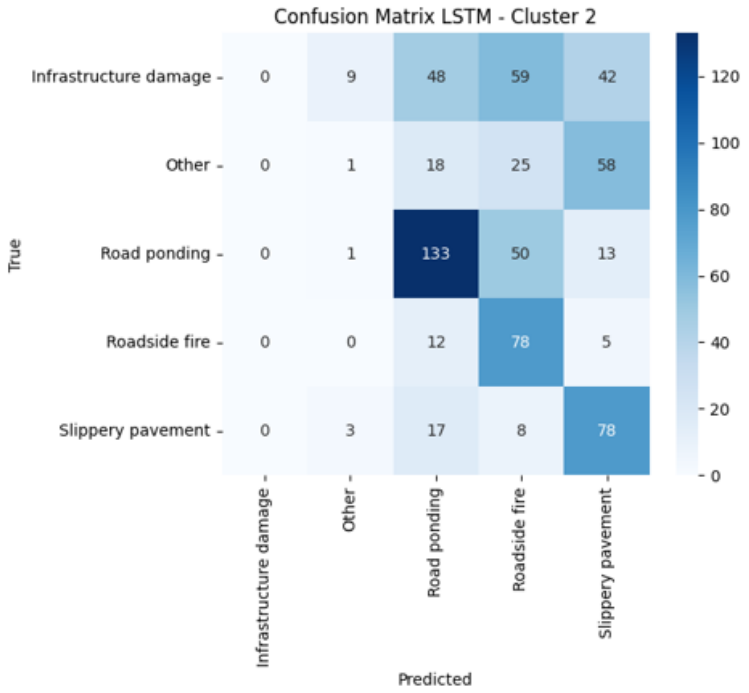

Figure S17: Confusion Matrix LSTM – Cluster 2

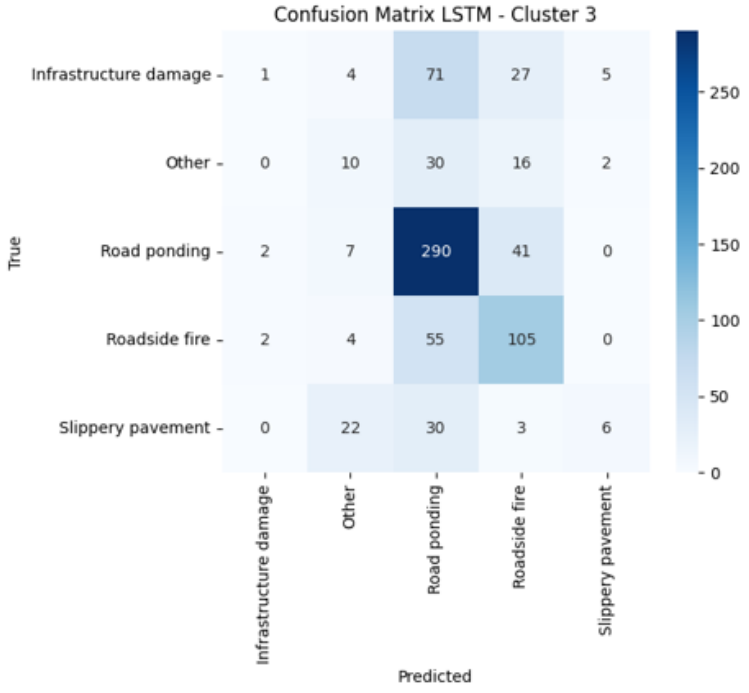

Figure S18: Confusion Matrix LSTM – Cluster 3

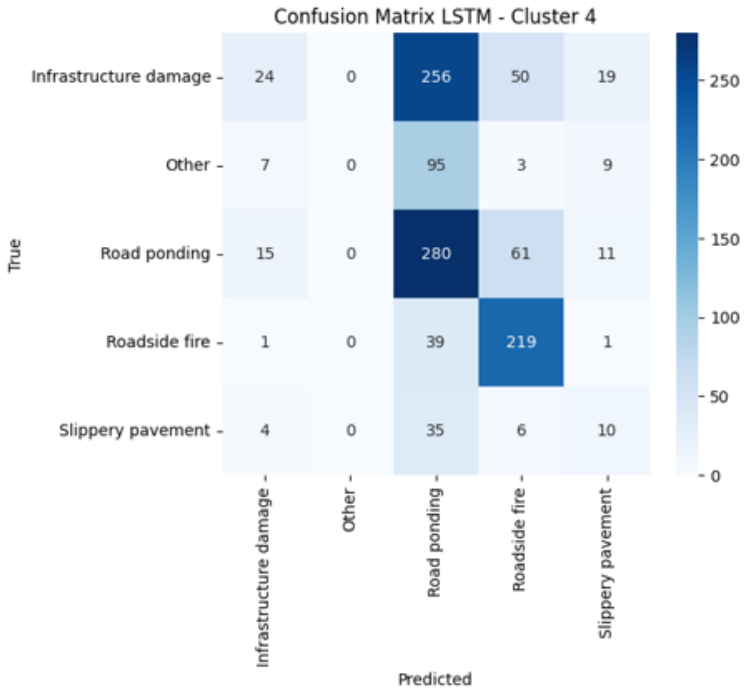

Figure S19: Confusion Matrix LSTM – Cluster 4

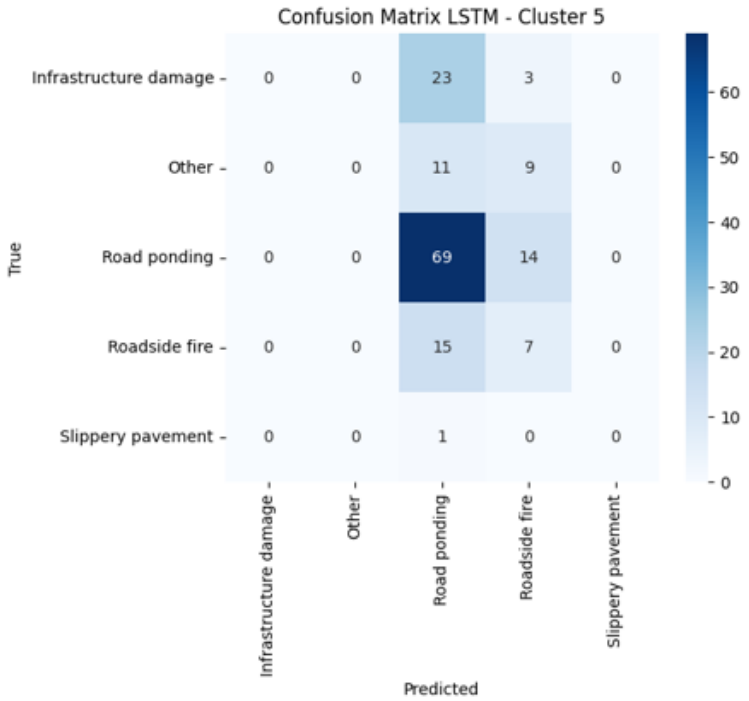

Figure S20: Confusion Matrix LSTM – Cluster 5

S4.5 Bidirectional Long-Short Term Memory Network

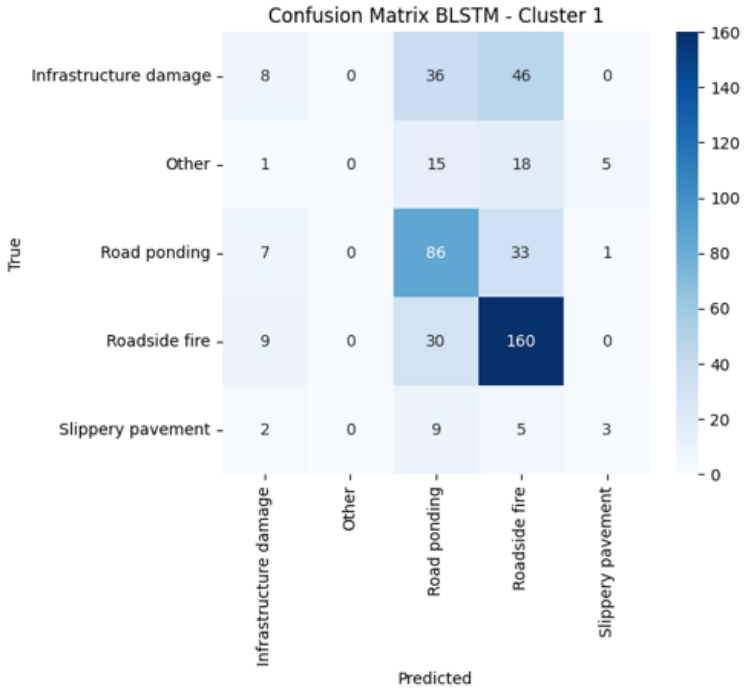

Figure S21: Confusion Matrix BLSTM – Cluster 1

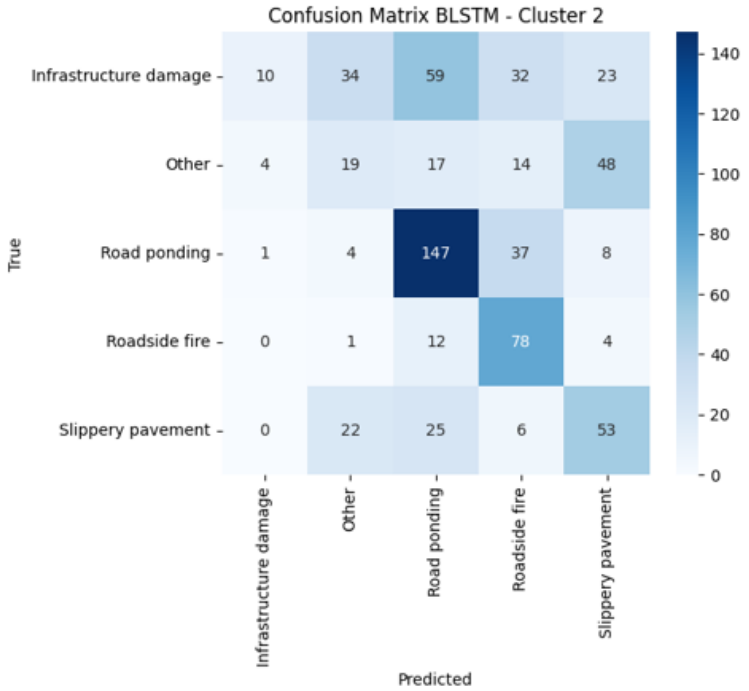

Figure S22: Confusion Matrix BLSTM – Cluster 2

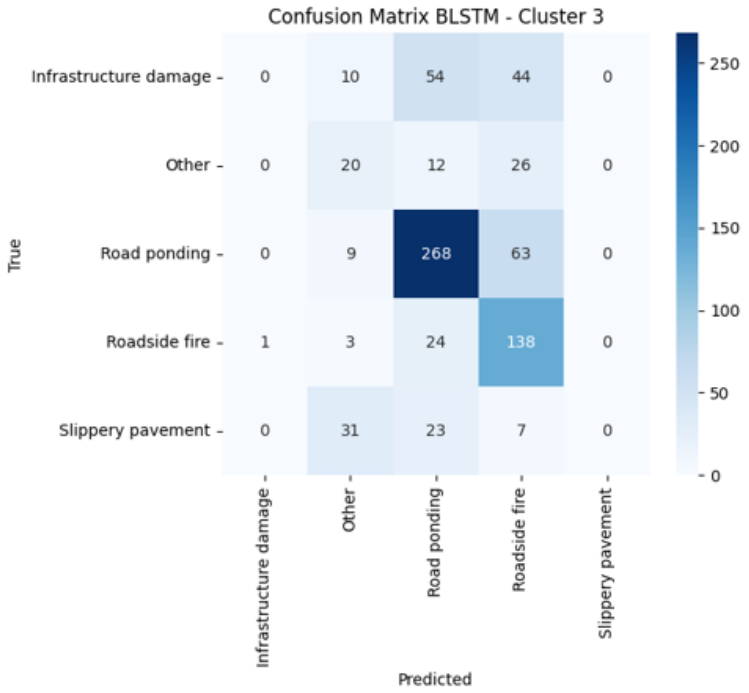

Figure S23: Confusion Matrix BLSTM – Cluster 3

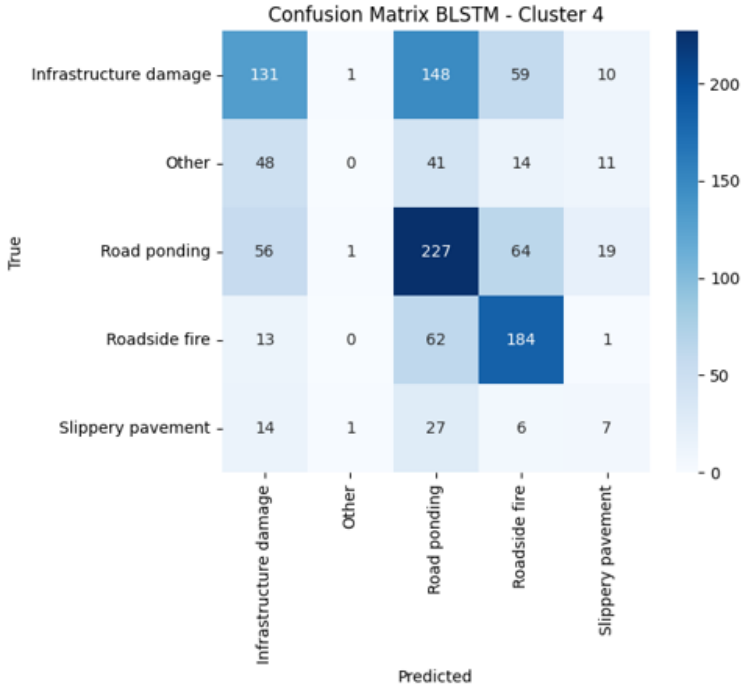

Figure S24: Confusion Matrix BLSTM – Cluster 4

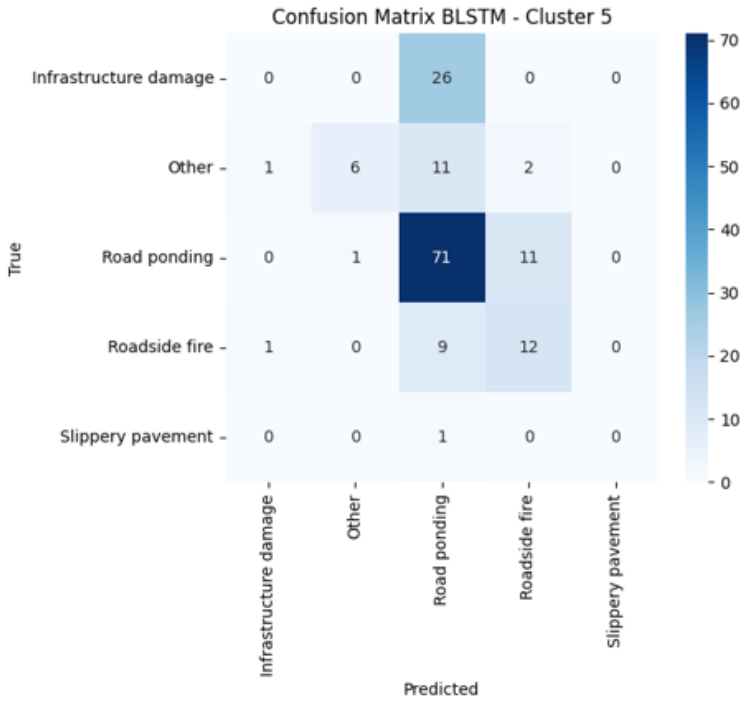

Figure S25: Confusion Matrix BLSTM – Cluster 5

S4.6 Graph Convolution Neural Network

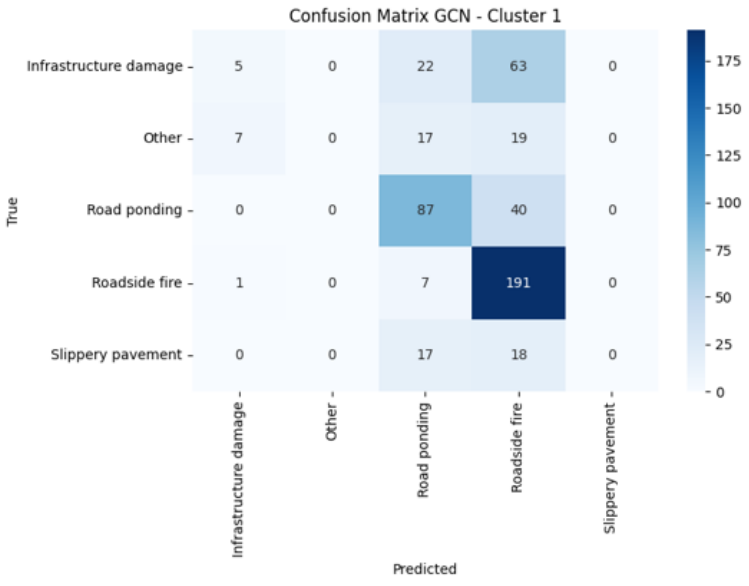

Figure S26: Confusion Matrix GCN – Cluster 1

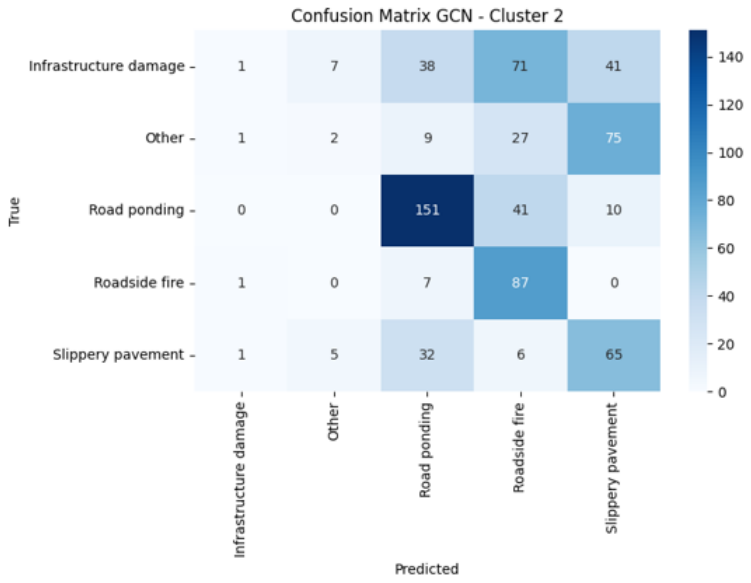

Figure S27: Confusion Matrix GCN – Cluster 2

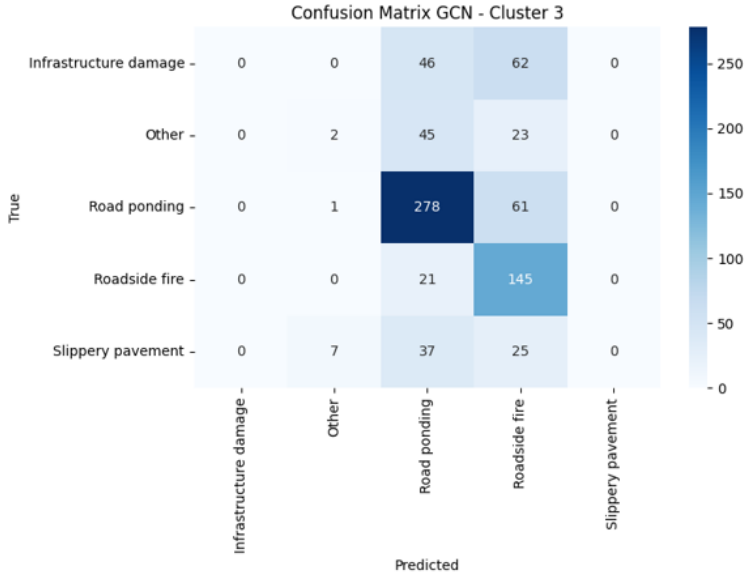

Figure S28: Confusion Matrix GCN – Cluster 3

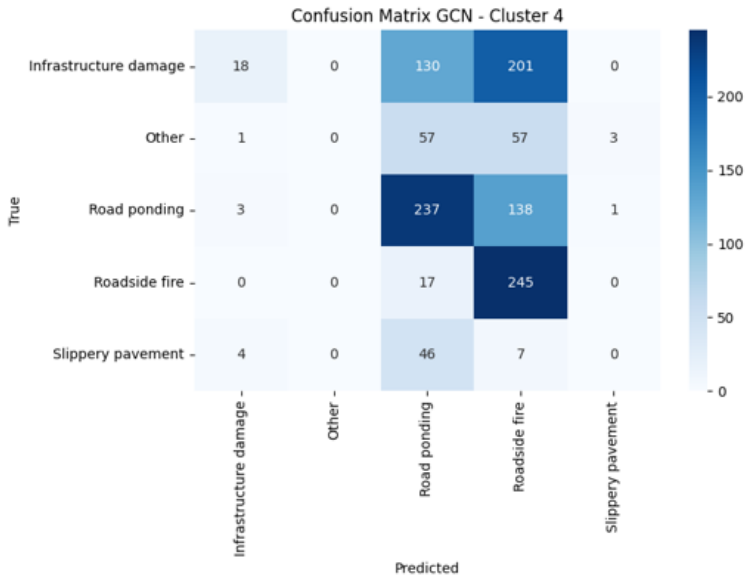

Figure S29: Confusion Matrix GCN – Cluster 4

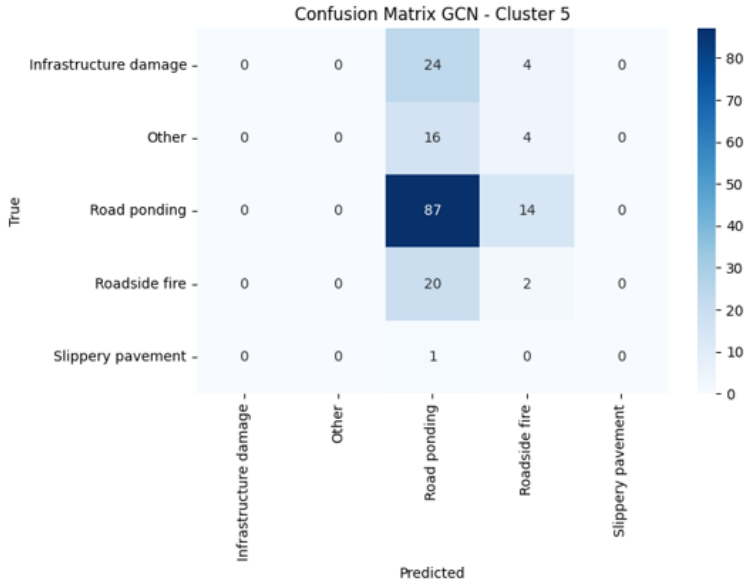

Figure S30: Confusion Matrix GCN – Cluster 5

S4.7 Graph Attention Network

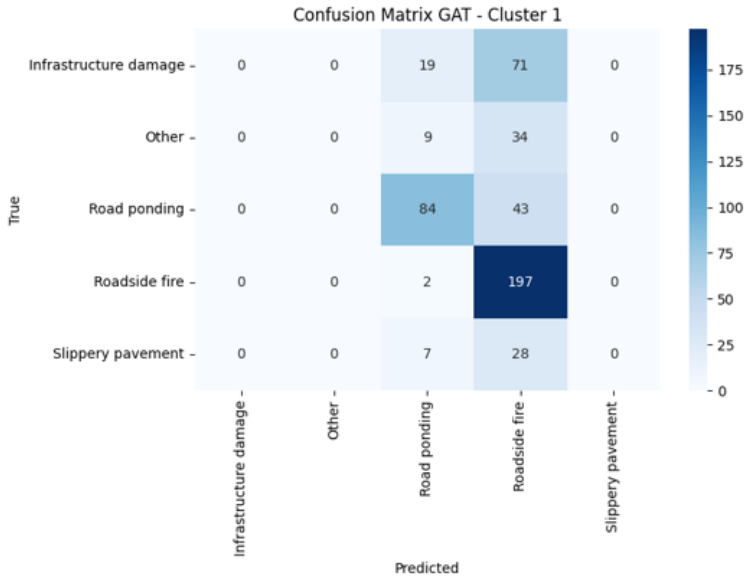

Figure S31: Confusion Matrix GAT – Cluster 1

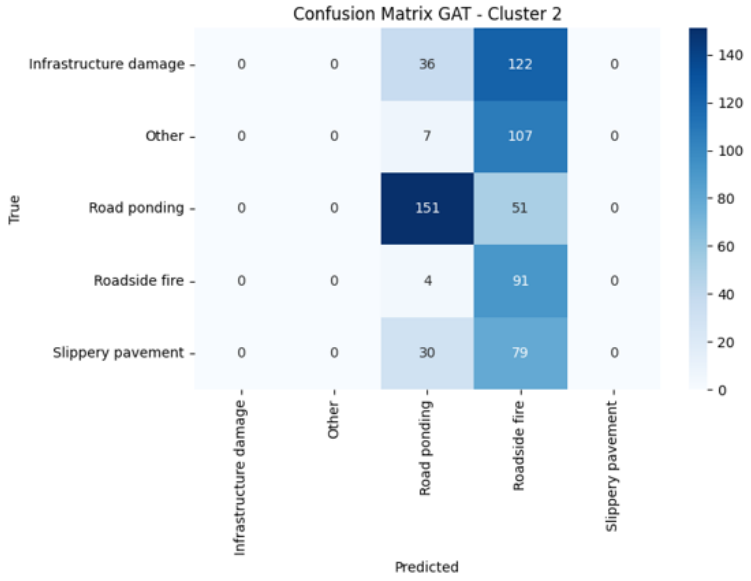

Figure S32: Confusion Matrix GAT – Cluster 2

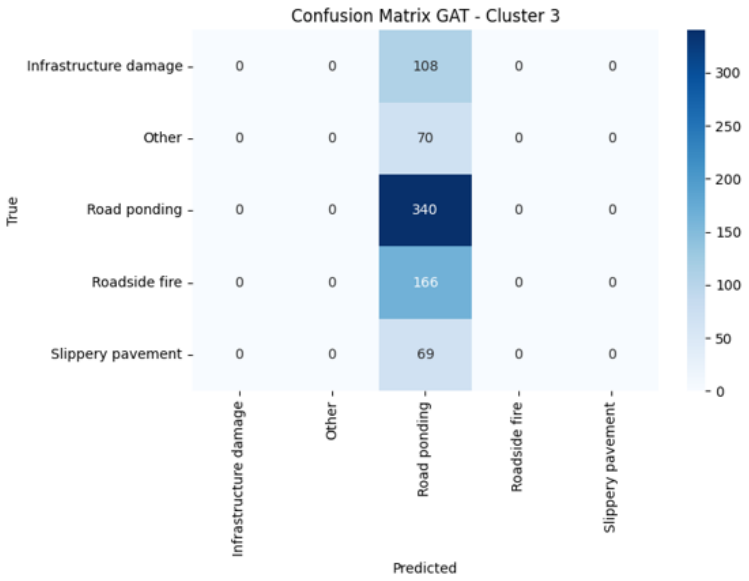

Figure S33: Confusion Matrix GAT – Cluster 3

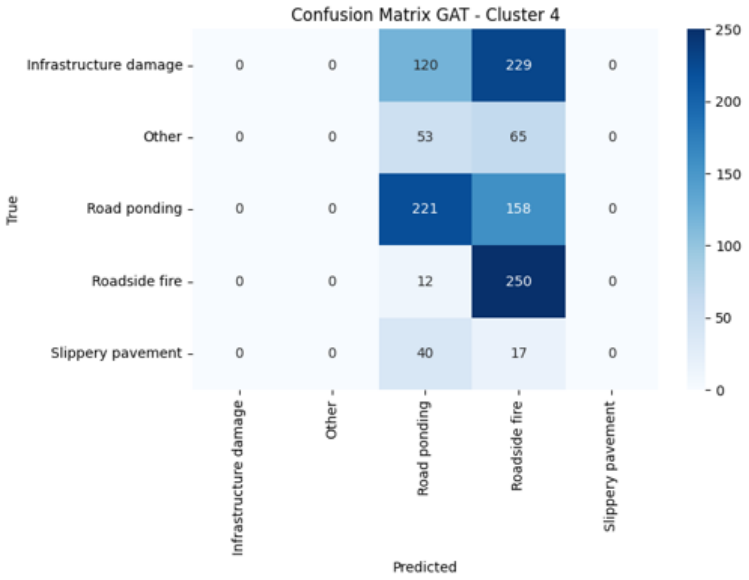

Figure S34: Confusion Matrix GAT – Cluster 4

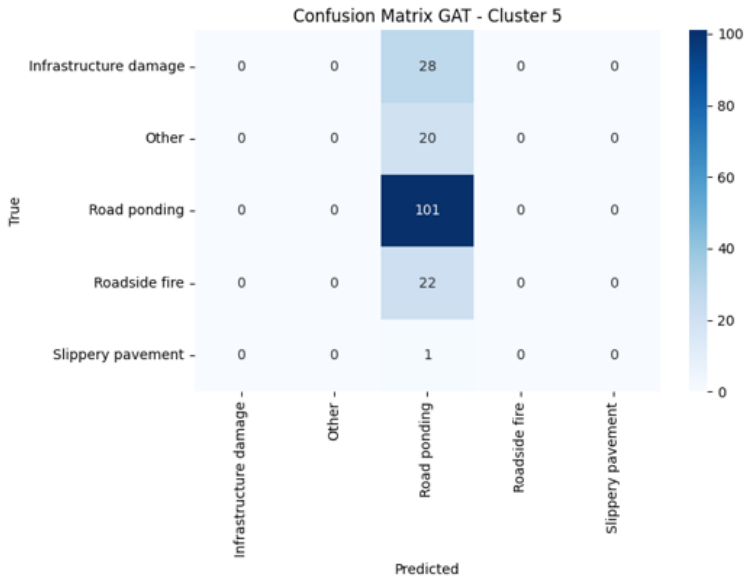

Figure S35: Confusion Matrix GAT – Cluster 5

S4.8 Graph Convolution Neural Network–Bidirectional Long-Short Term Memory Network

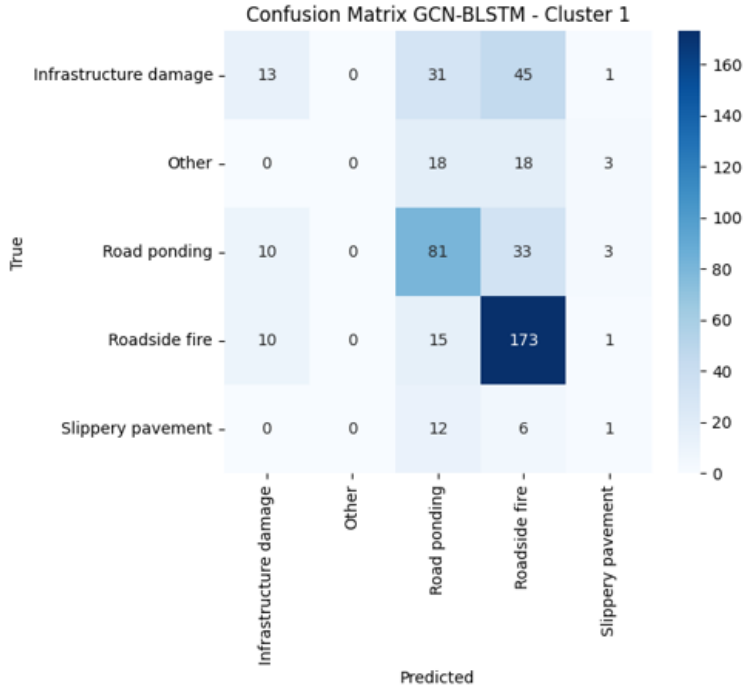

Figure S36: Confusion Matrix GCN-BLSTM – Cluster 1

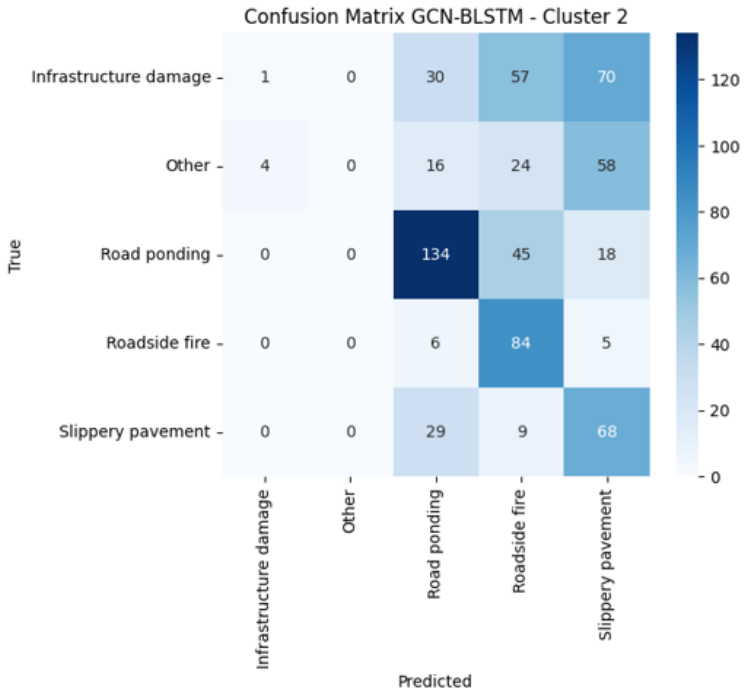

Figure S37: Confusion Matrix GCN-BLSTM – Cluster 2

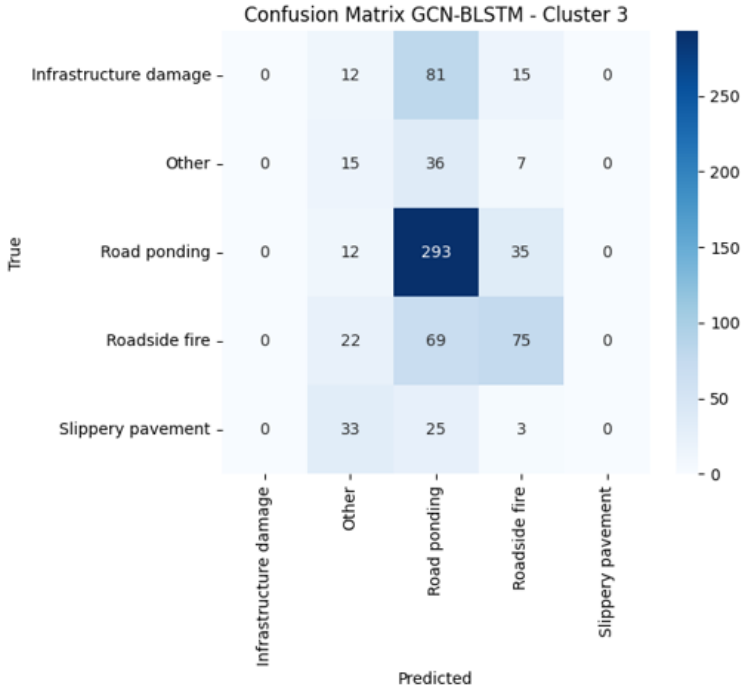

Figure S38: Confusion Matrix GCN-BLSTM – Cluster 3

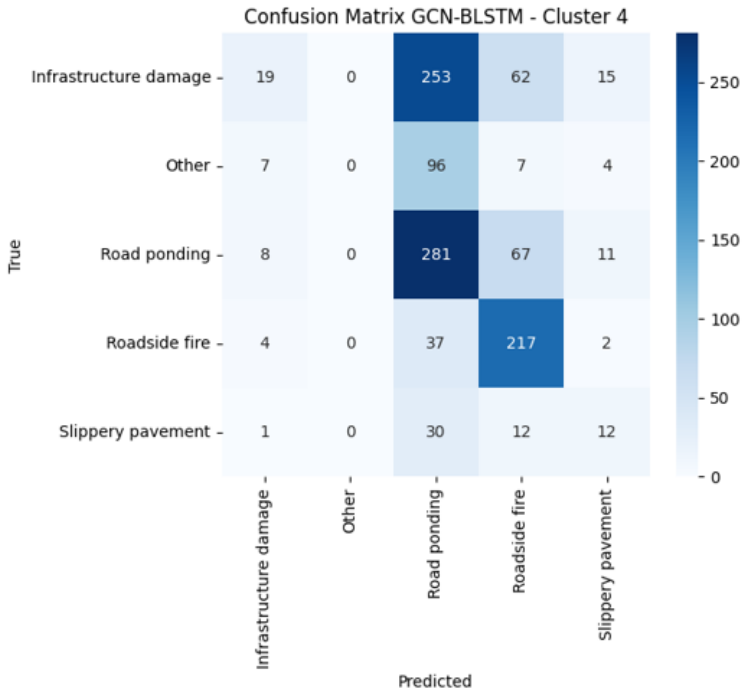

Figure S39: Confusion Matrix GCN-BLSTM – Cluster 4

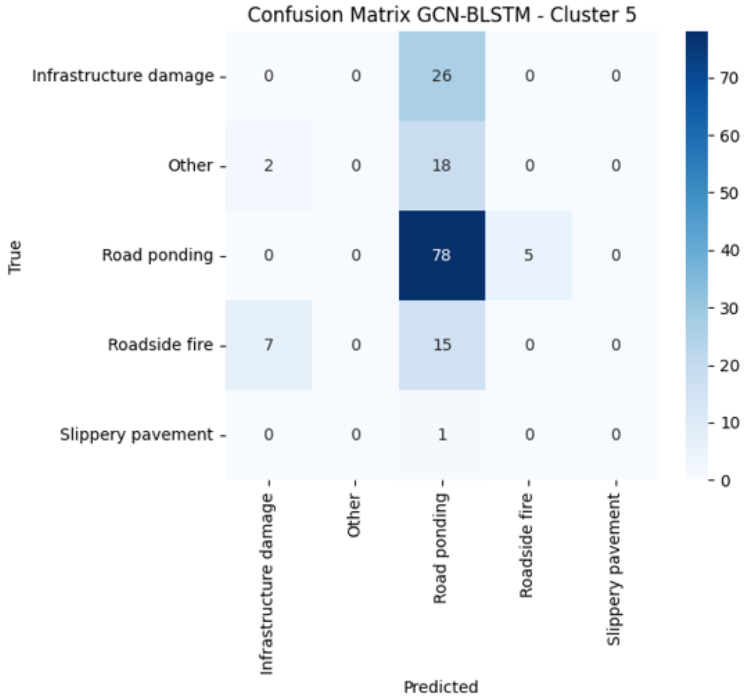

Figure S40: Confusion Matrix GCN-BLSTM – Cluster 5

S4.9 Graph Attention Network–Bidirectional Long-Short Term Memory Network

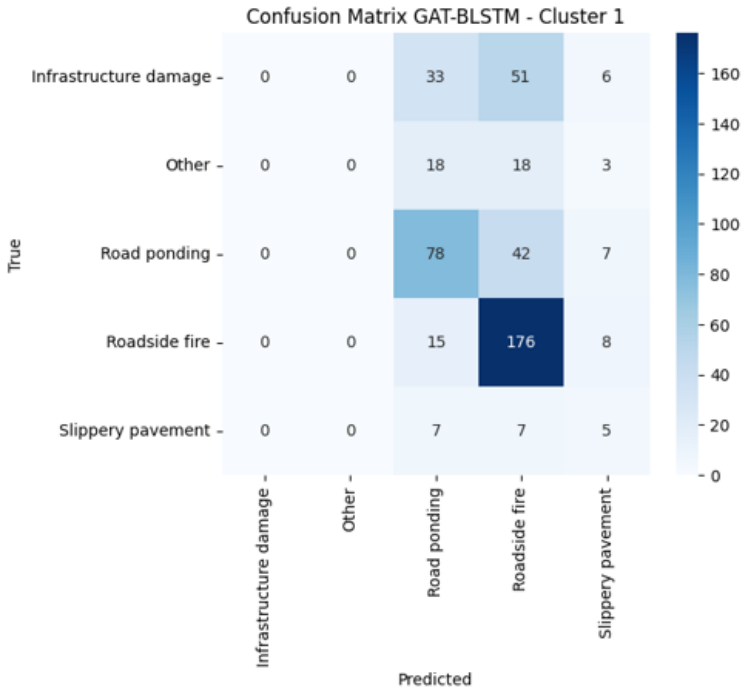

Figure S41: Confusion Matrix GAT-BLSTM – Cluster 1

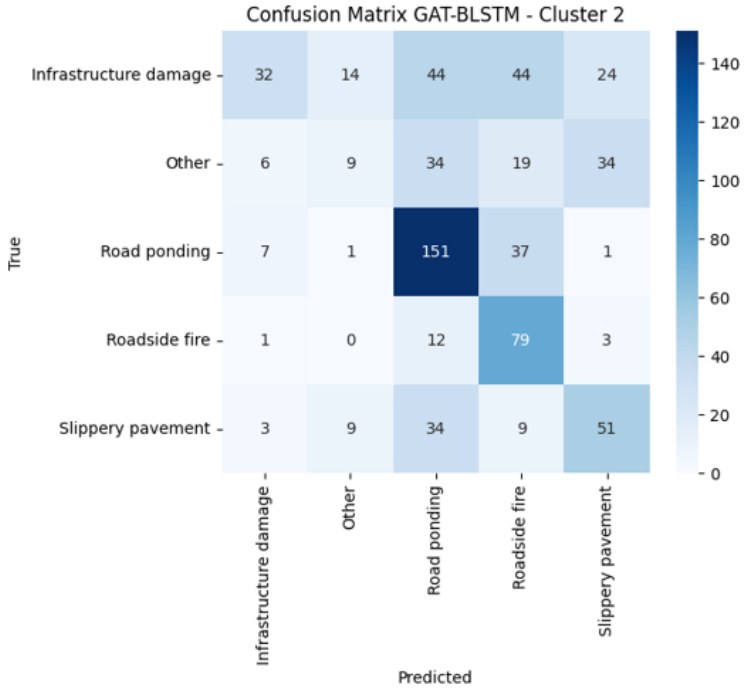

Figure S42: Confusion Matrix GAT-BLSTM – Cluster 2

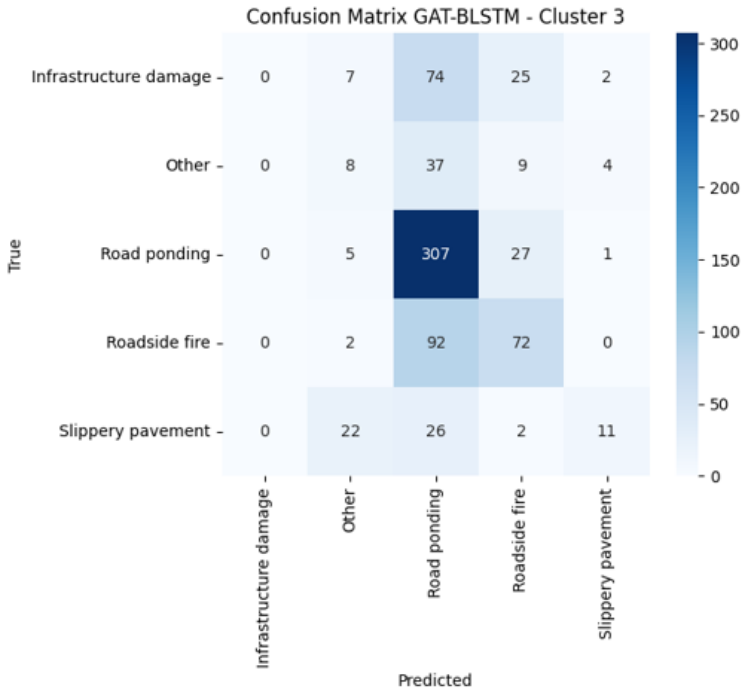

Figure S43: Confusion Matrix GAT-BLSTM – Cluster 3

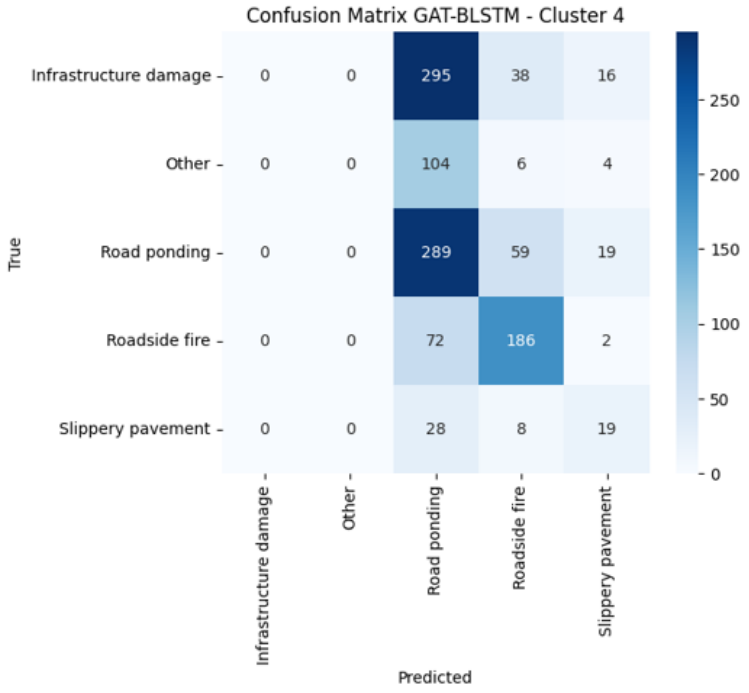

Figure S44: Confusion Matrix GAT-BLSTM – Cluster 4

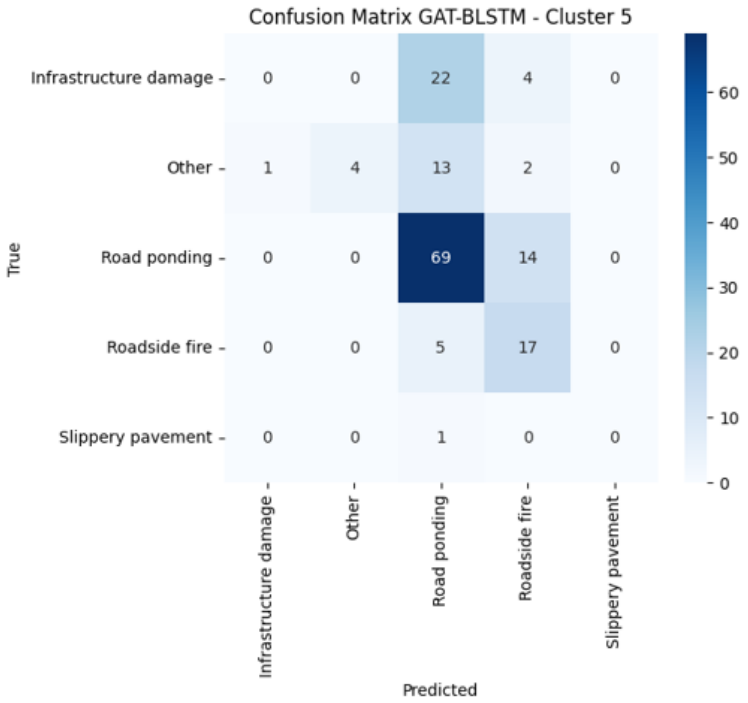

Figure S45: Confusion Matrix GAT-BLSTM – Cluster 5

S5 Probability heat maps

This section provides additional probability heat maps for the evaluated models. These maps are included as supplementary material to support the hotspot-map analysis presented in Section 6.3 URC hot spot probability maps.

S5.1 Logistic Regression

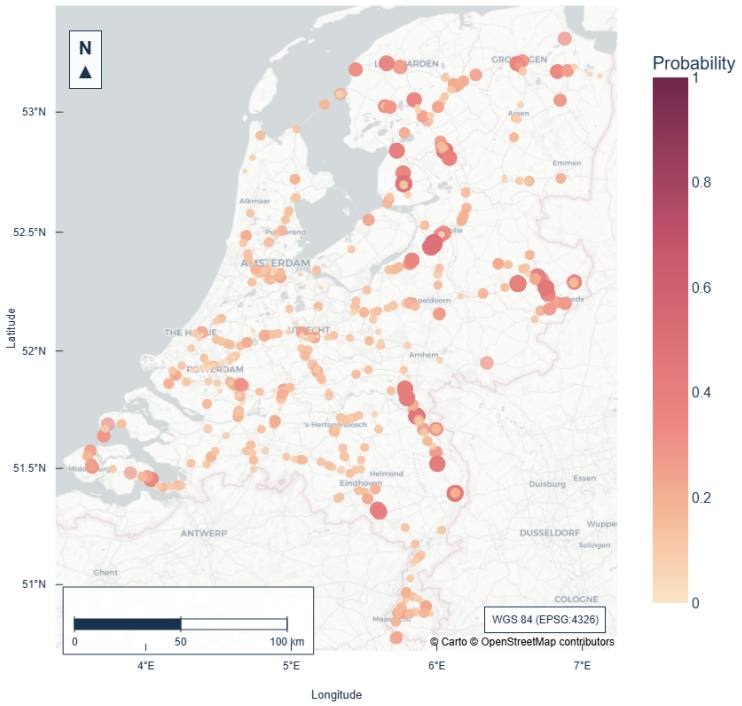

Figure S46: Probability heatmap for infrastructure damage predictions – LR

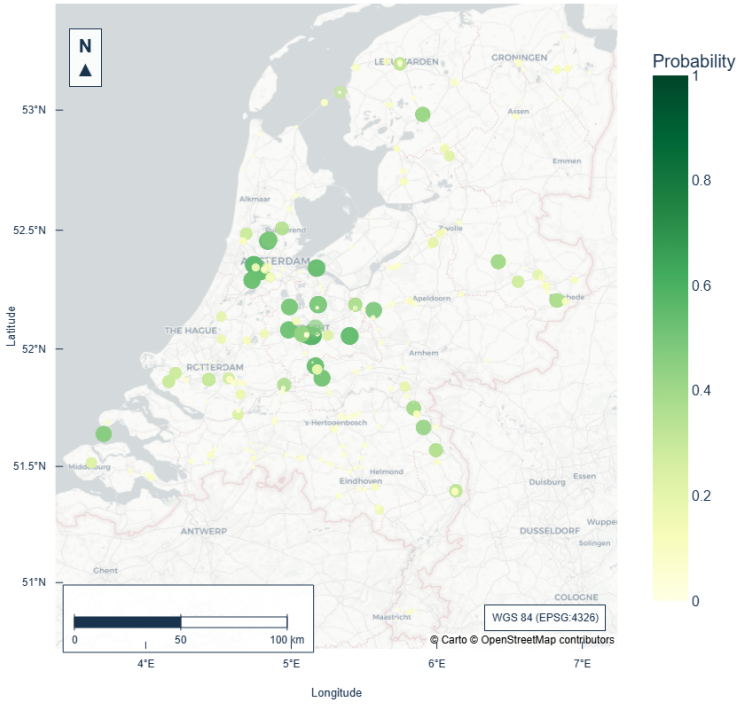

Figure S47: Probability heatmap for slippery pavement predictions – LR

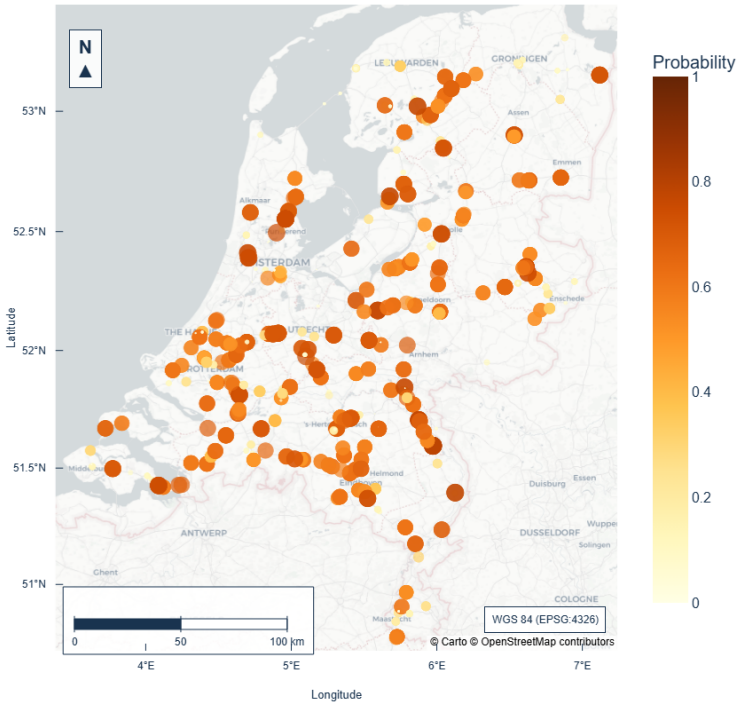

Figure S48: Probability heatmap for roadside fire predictions – LR

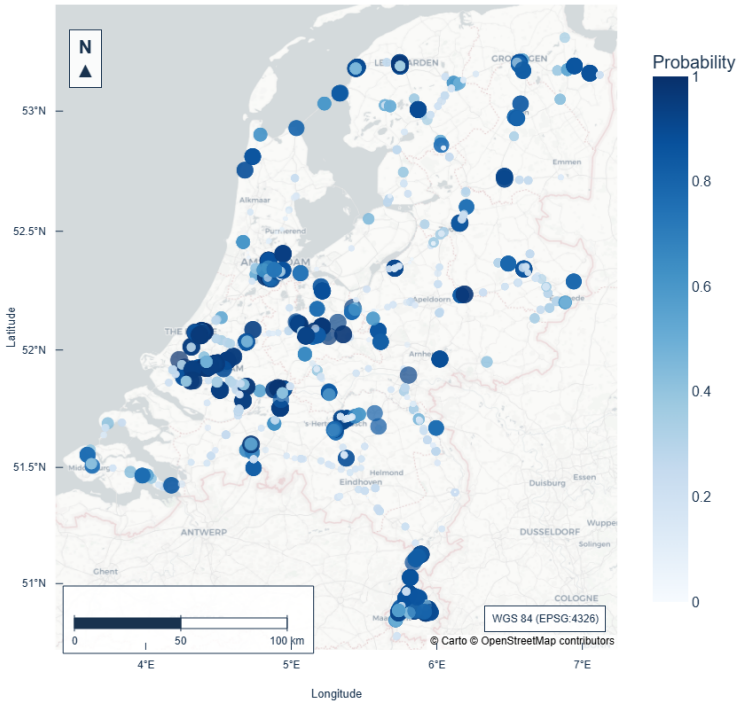

Figure S49: Probability heatmap for road ponding predictions – LR

S5.2 Extreme Gradient Boosting

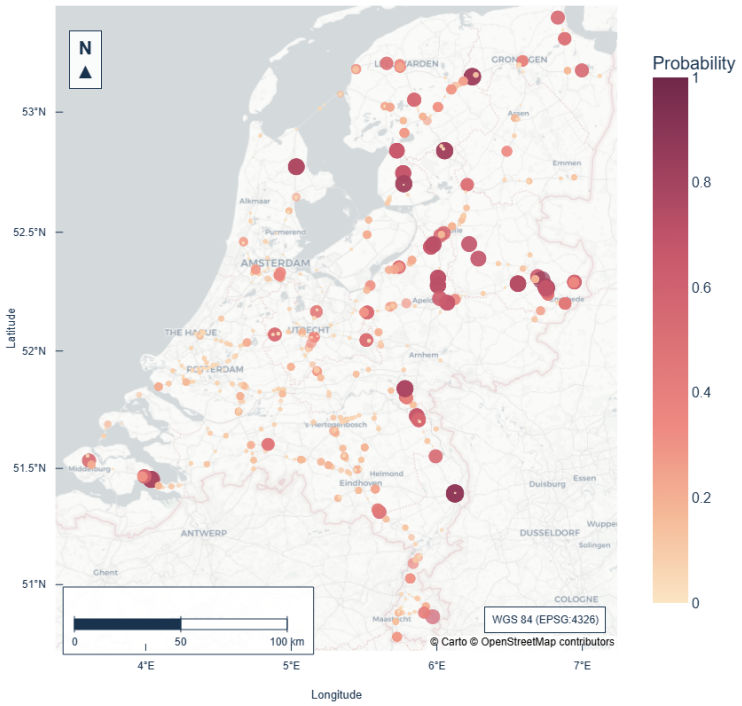

Figure S50: Probability heatmap for infrastructure damage predictions – XGB

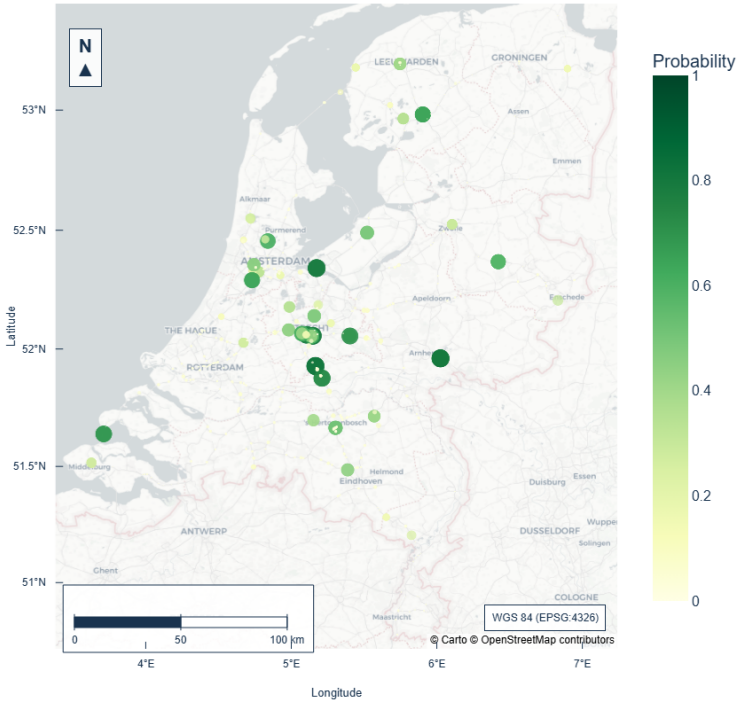

Figure S51: Probability heatmap for slippery pavement predictions – XGB

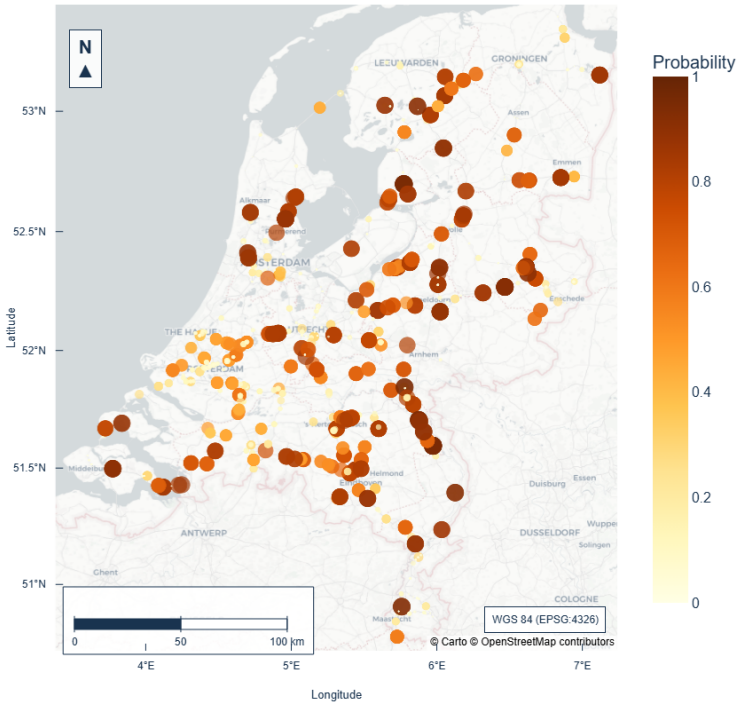

Figure S52: Probability heatmap for roadside fire predictions – XGB

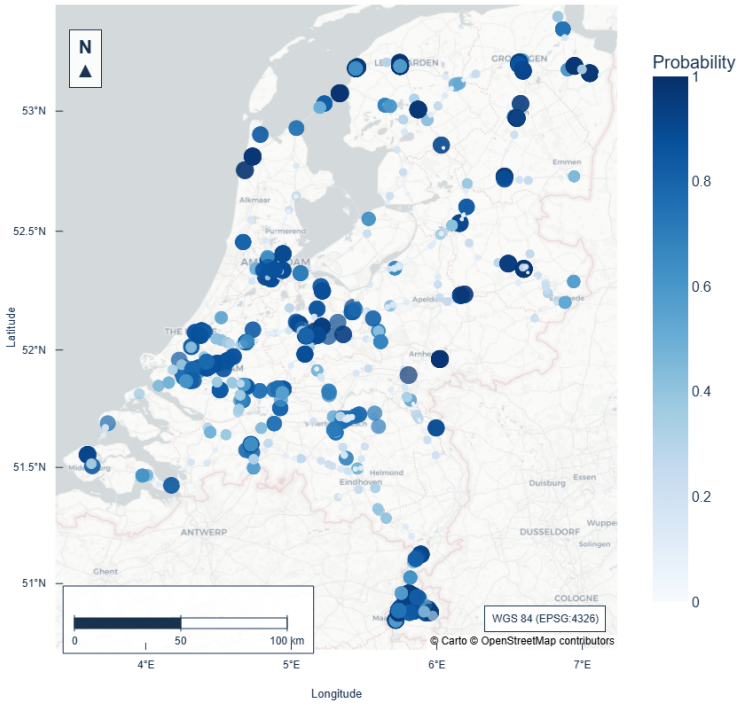

Figure S53: Probability heatmap for road ponding predictions – XGB

S5.3 Long-Short Term Memory Network

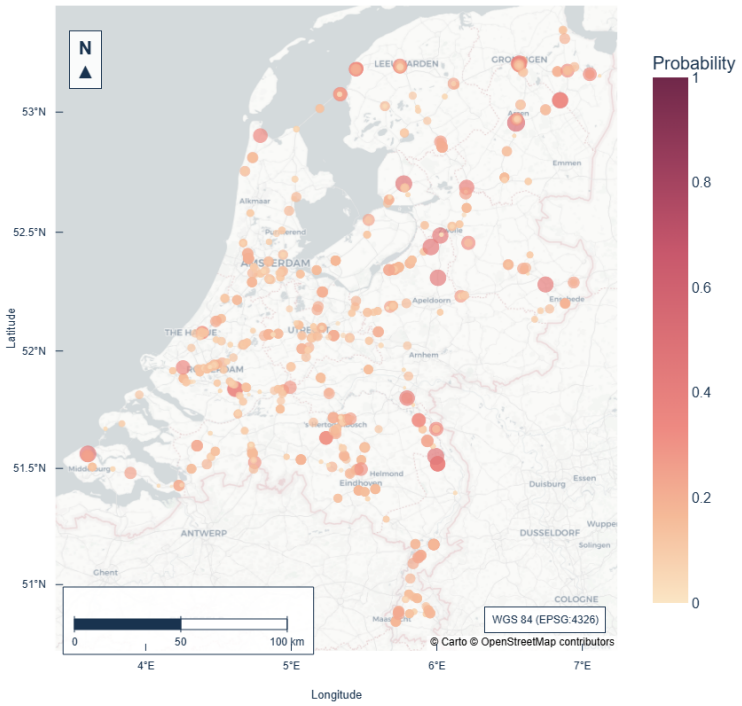

Figure S54: Probability heatmap for infrastructure damage predictions – LSTM

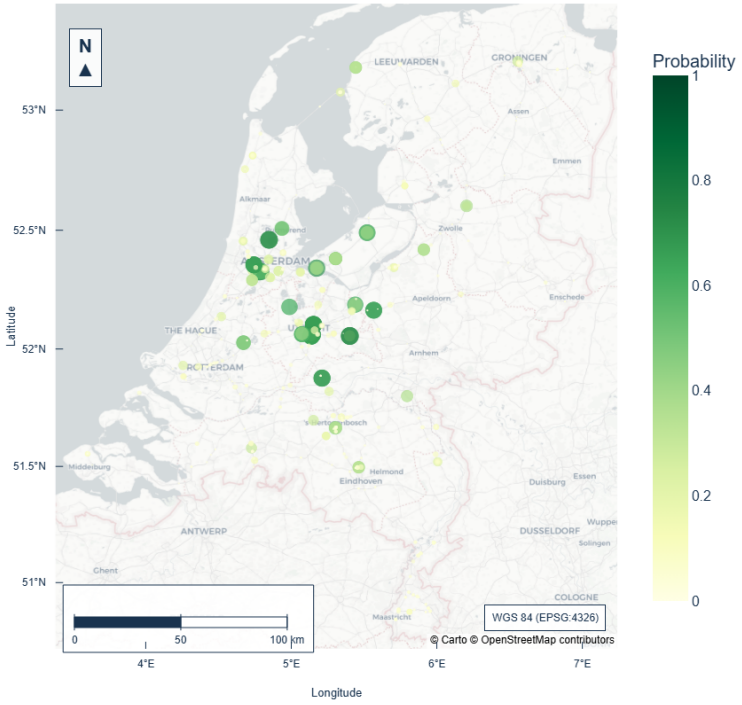

Figure S55: Probability heatmap for slippery pavement predictions – LSTM

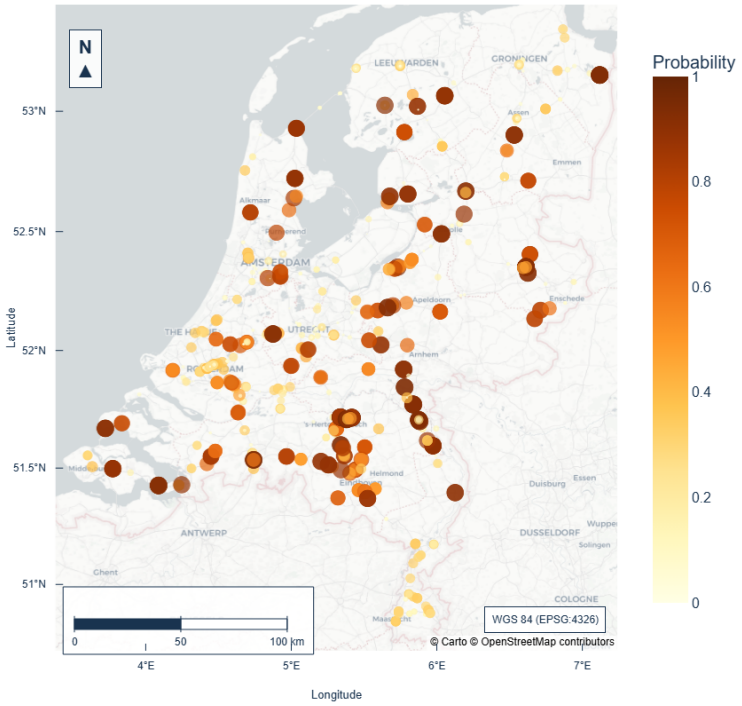

Figure S56: Probability heatmap for roadside fire predictions – LSTM

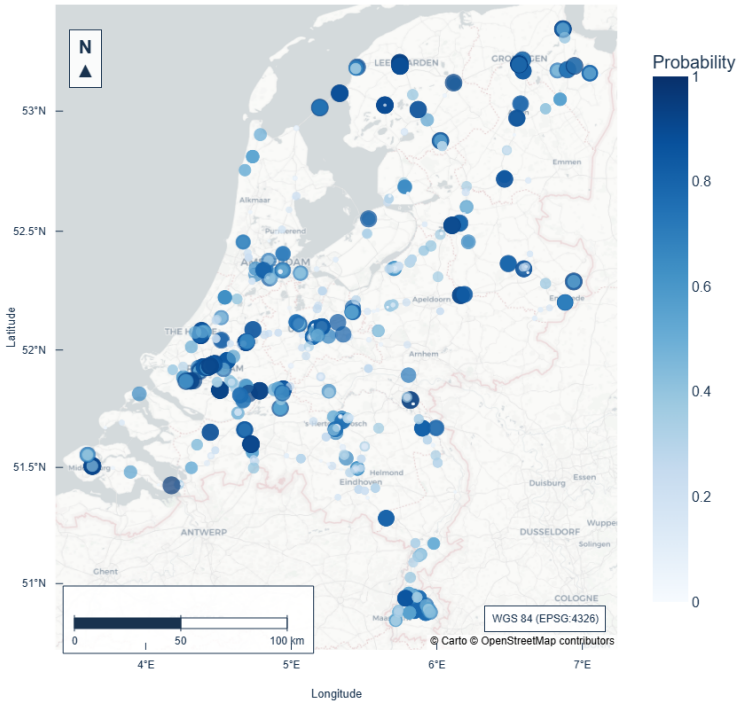

Figure S57: Probability heatmap for road ponding predictions – LSTM

S5.4 Bidirectional Long-Short Term Memory Network

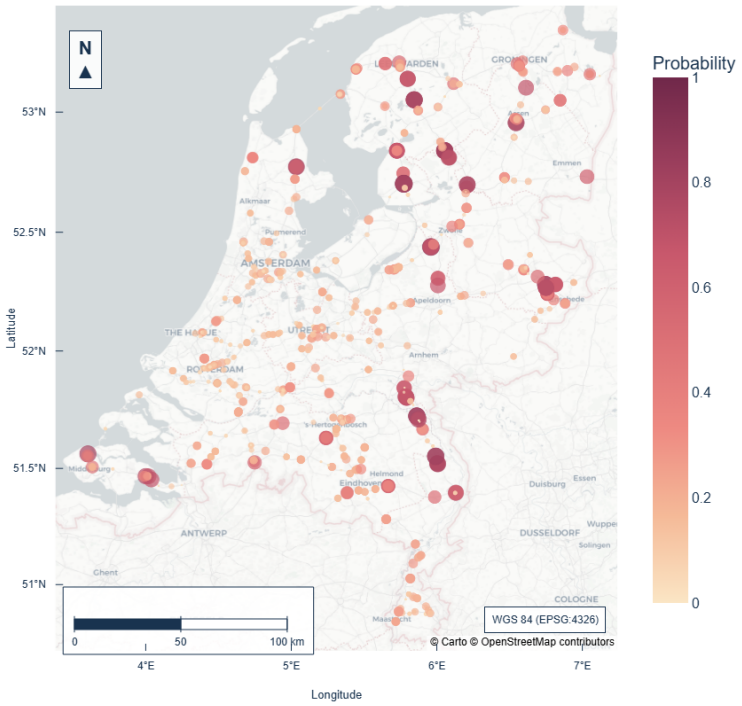

Figure S58: Probability heatmap for infrastructure damage predictions – BLSTM

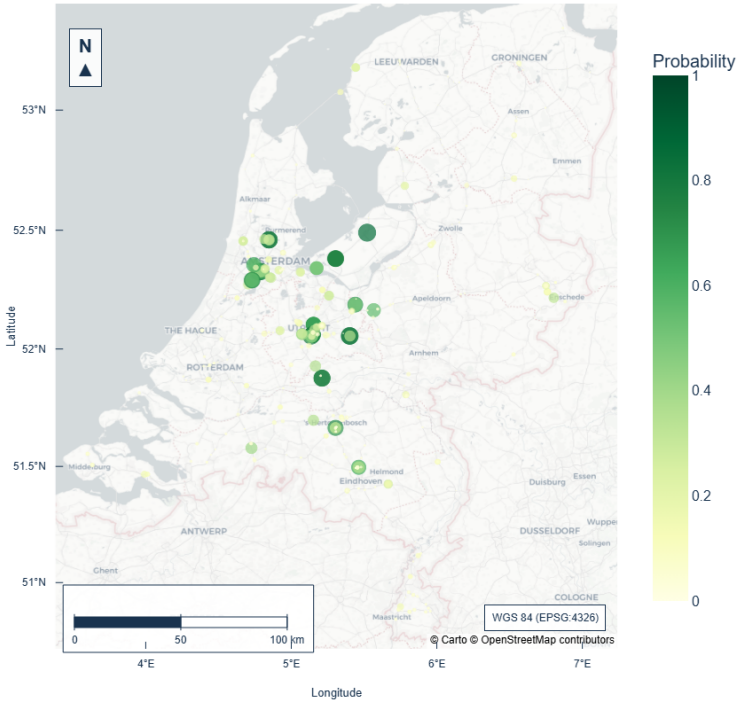

Figure S59: Probability heatmap for slippery pavement predictions – BLSTM

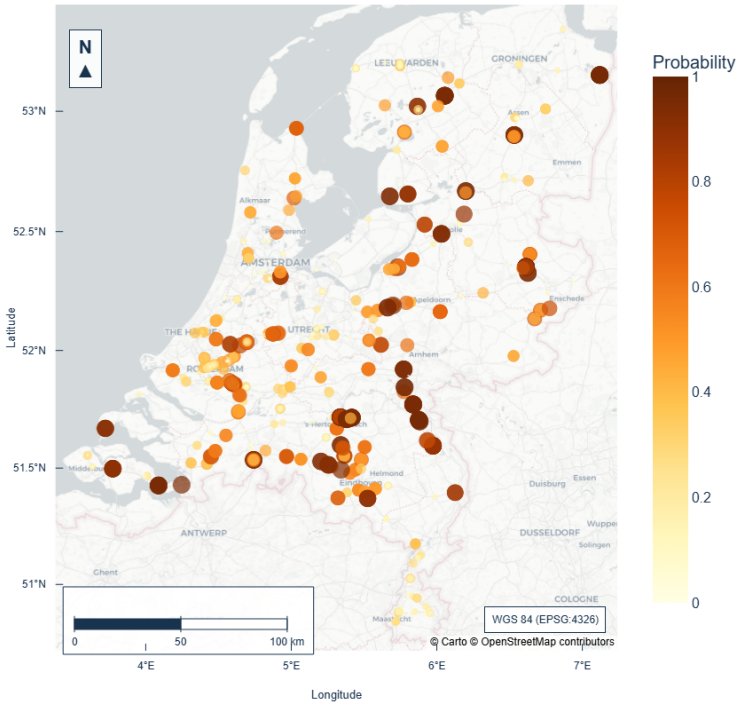

Figure S60: Probability heatmap for roadside fire predictions – BLSTM

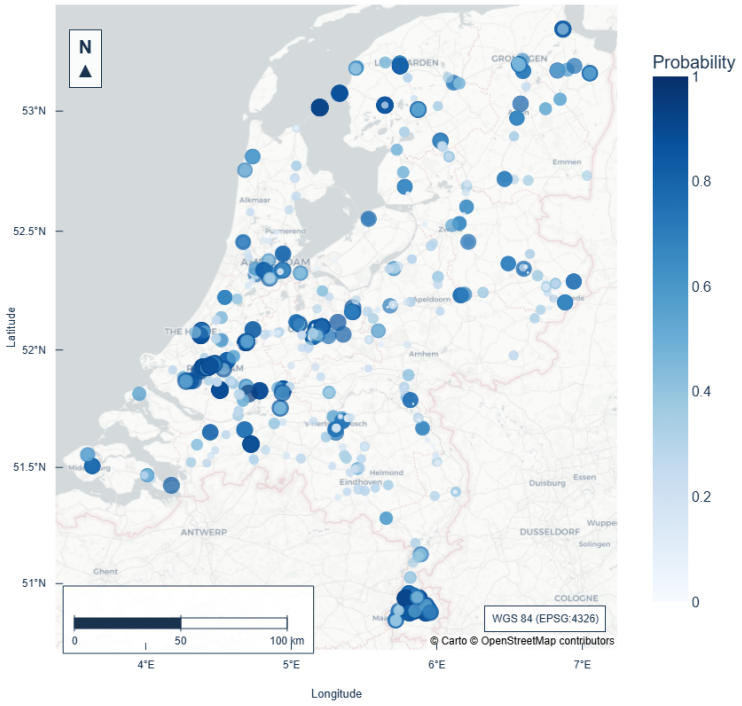

Figure S61: Probability heatmap for road ponding predictions – BLSTM

S5.5 Graph Convolution Network

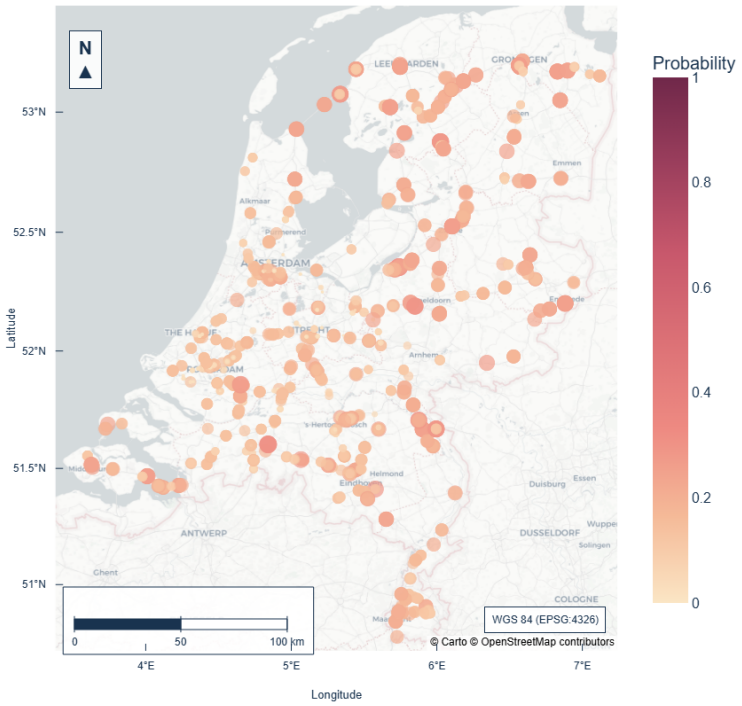

Figure S62: Probability heatmap for infrastructure damage predictions – GCN

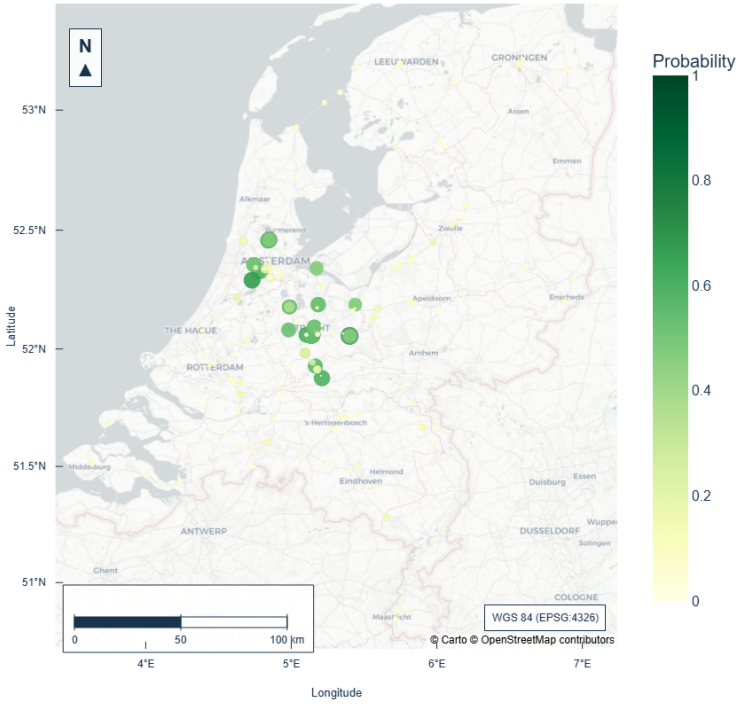

Figure S63: Probability heatmap for slippery pavement predictions – GCN

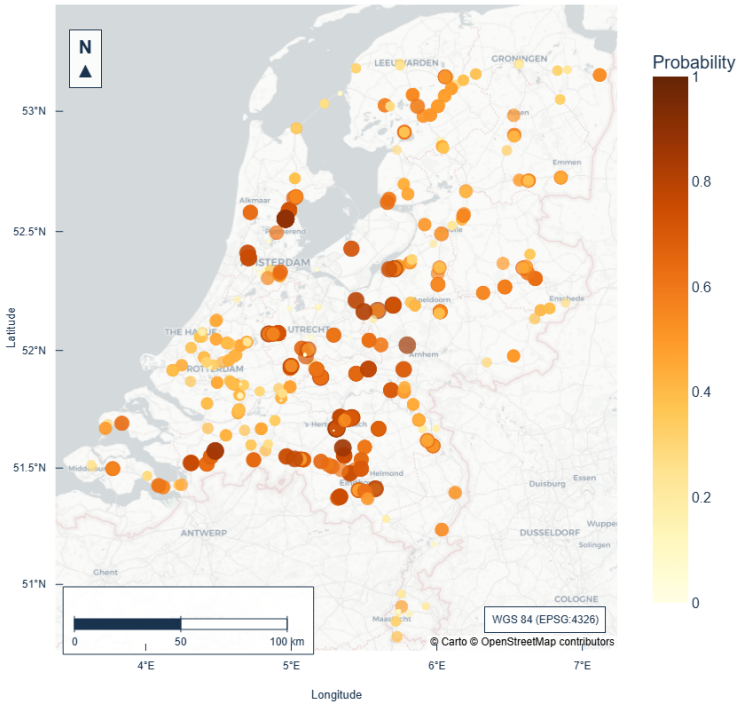

Figure S64: Probability heatmap for roadside fire predictions – GCN

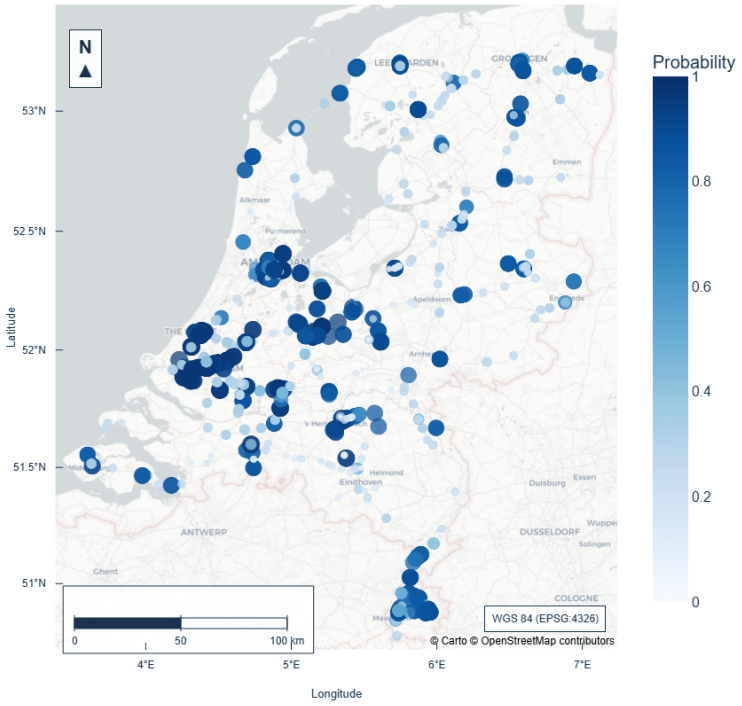

Figure S65: Probability heatmap for road ponding predictions – GCN

S5.6 Graph Attention Network

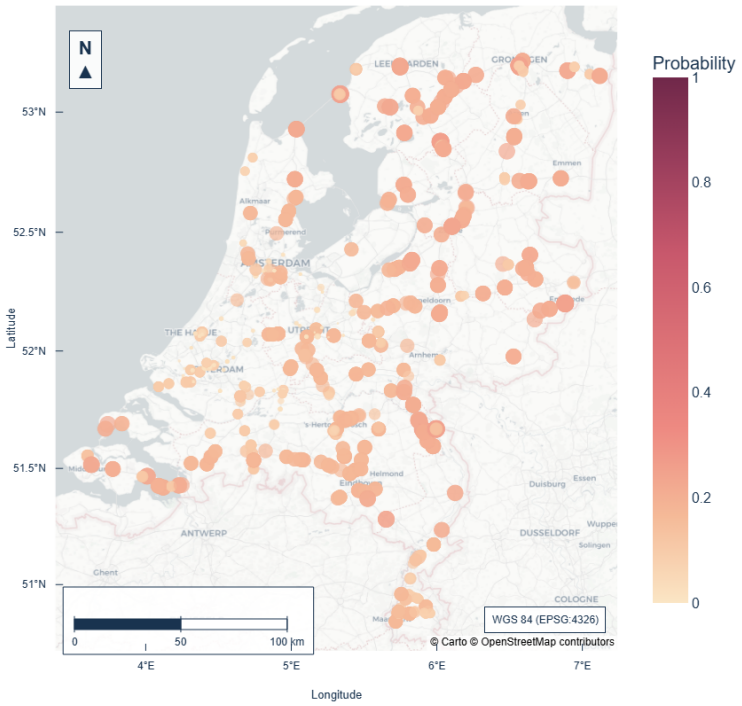

Figure S66: Probability heatmap for infrastructure damage predictions – GAT

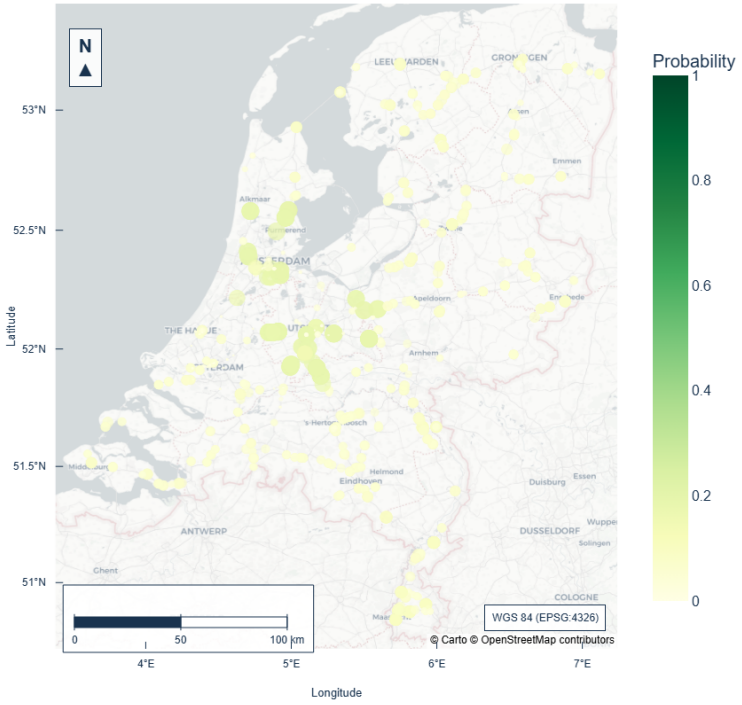

Figure S67: Probability heatmap for slippery pavement predictions – GAT

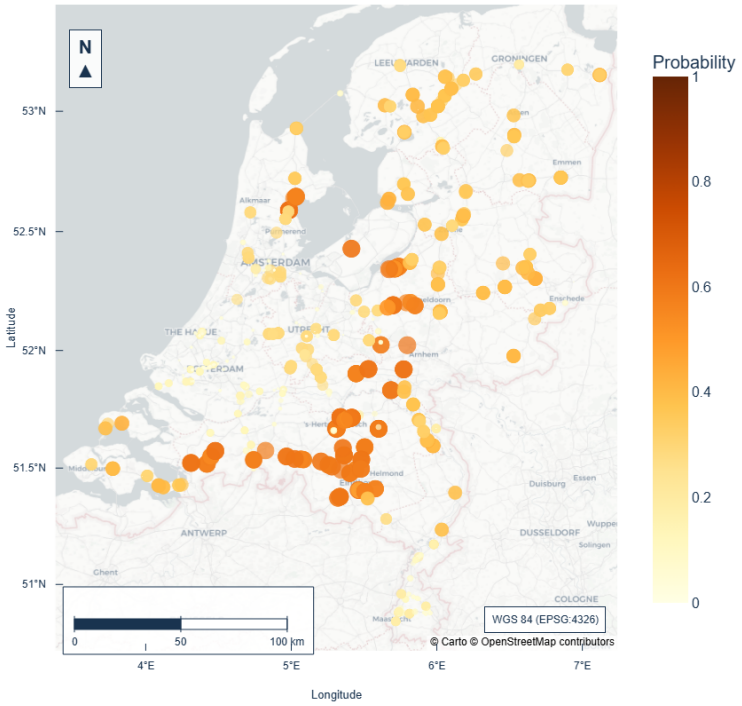

Figure S68: Probability heatmap for roadside fire predictions – GAT

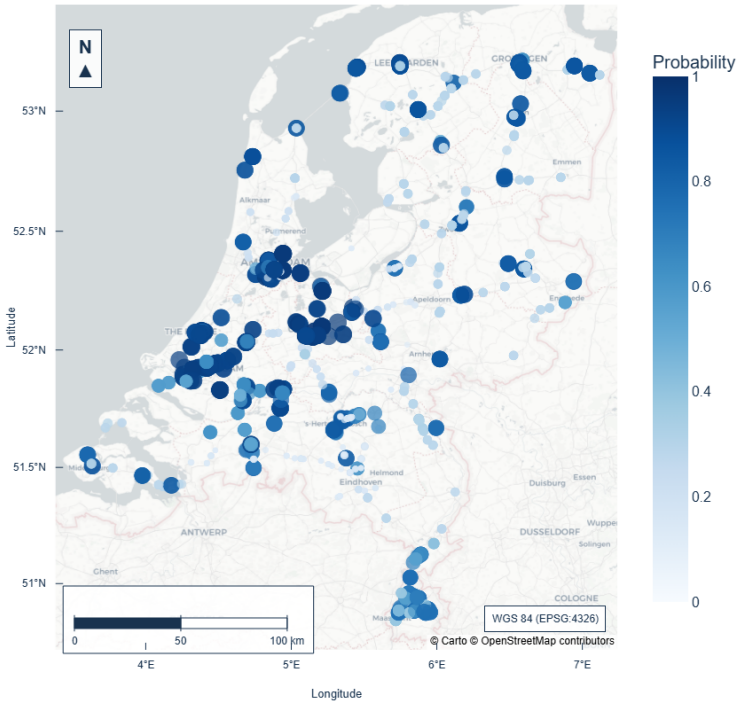

Figure S69: Probability heatmap for road ponding predictions – GAT

S5.7 Graph Convolution Network–Bidirectional Long-Short Term Memory Network

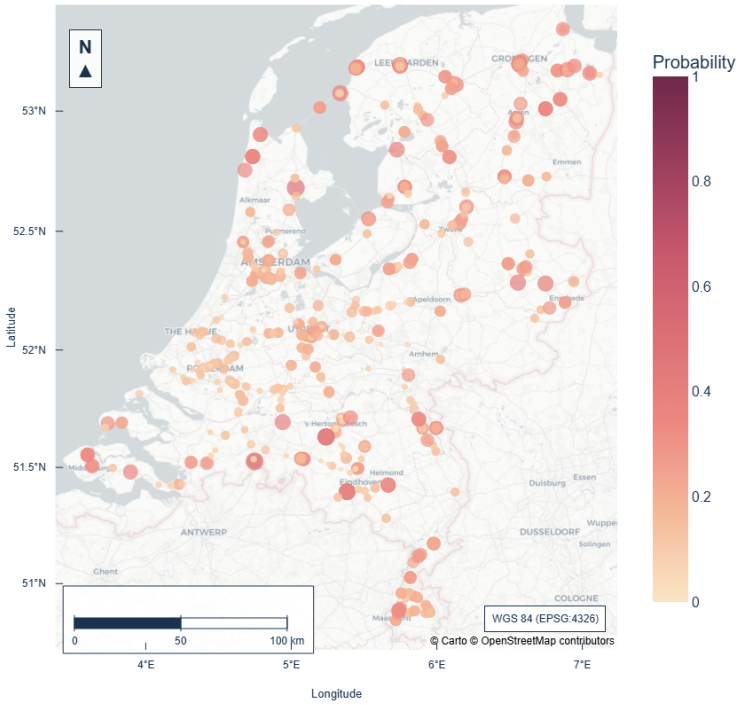

Figure S70: Probability heatmap for infrastructure damage predictions – GCN-BLSTM

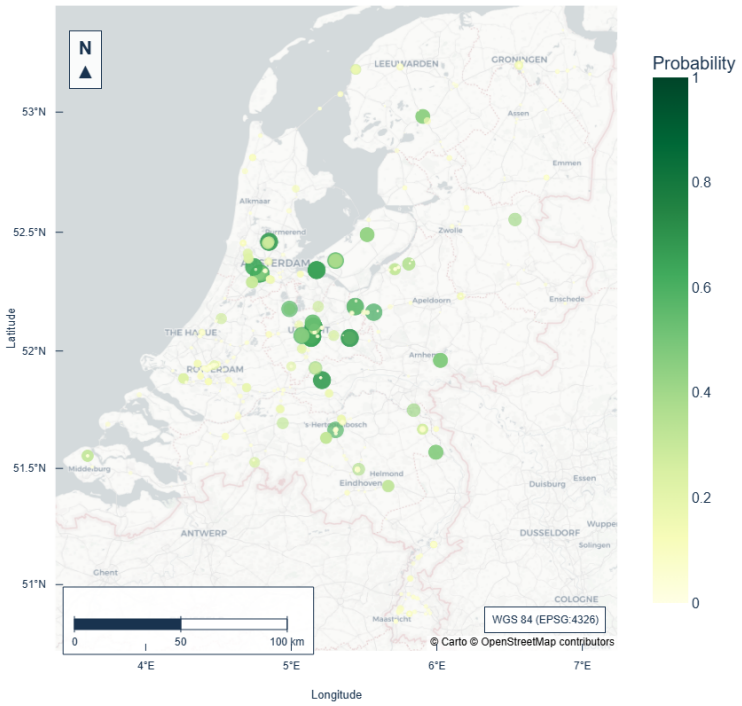

Figure S71: Probability heatmap for slippery pavement predictions – GCN-BLSTM

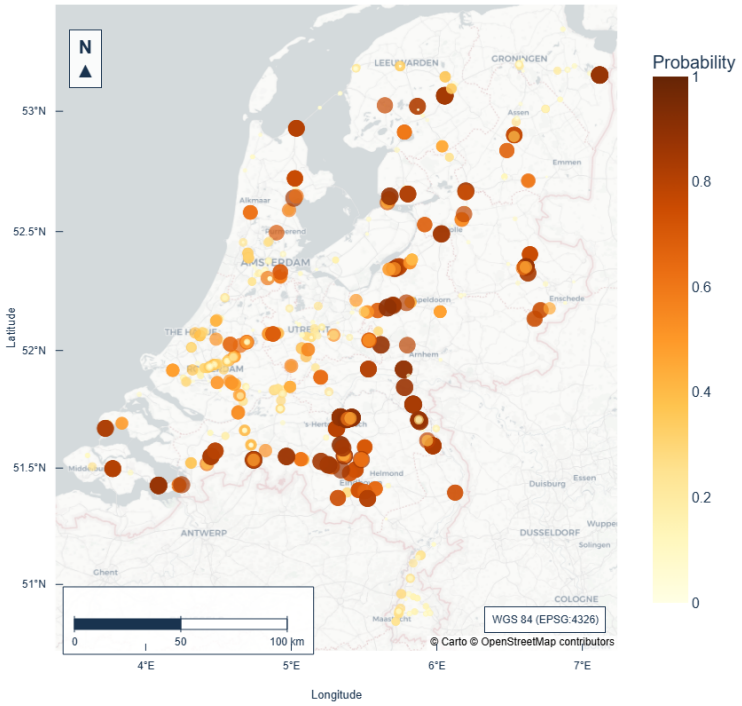

Figure S72: Probability heatmap for roadside fire predictions – GCN-BLSTM

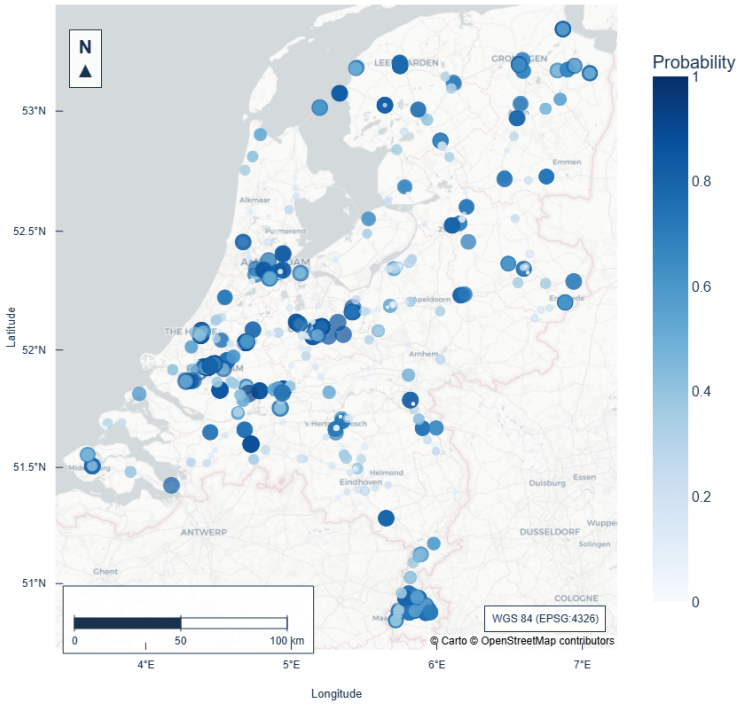

Figure S73: Probability heatmap for road ponding predictions – GCN-BLSTM

S5.8 Graph Attention Network–Bidirectional Long-Short Term Memory Network

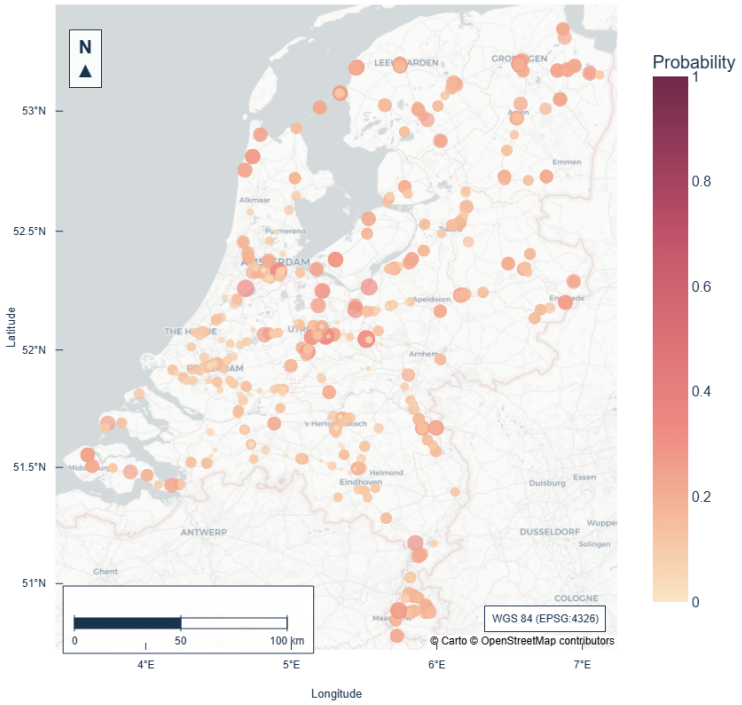

Figure S74: Probability heatmap for infrastructure damage predictions - GAT-BLSTM

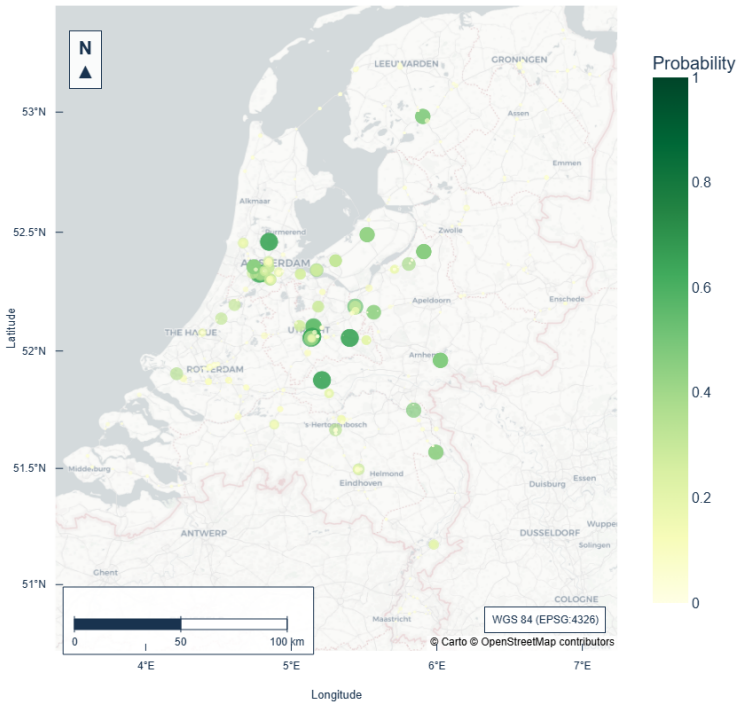

Figure S75: Probability heatmap for slippery pavement predictions – GAT-BLSTM

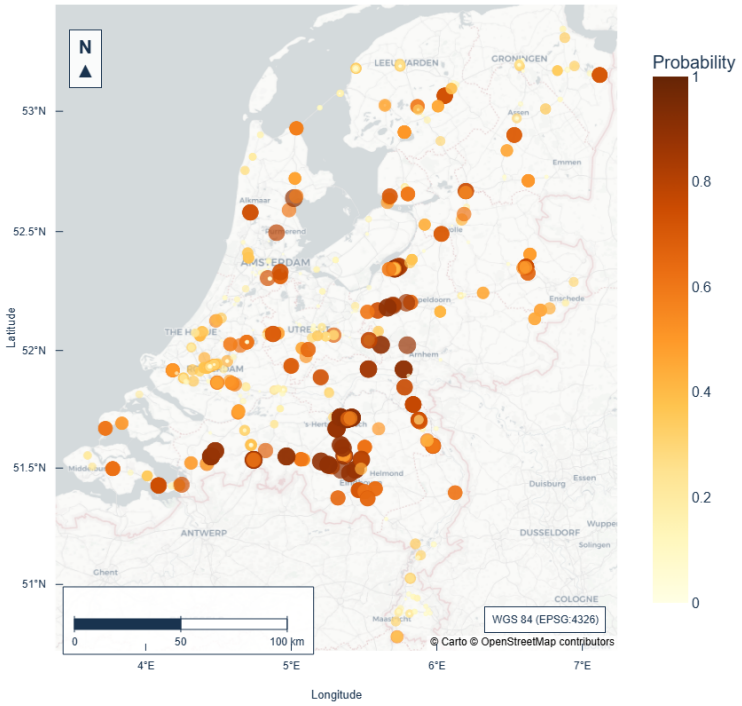

Figure S76: Probability heatmap for roadside fire predictions – GAT-BLSTM

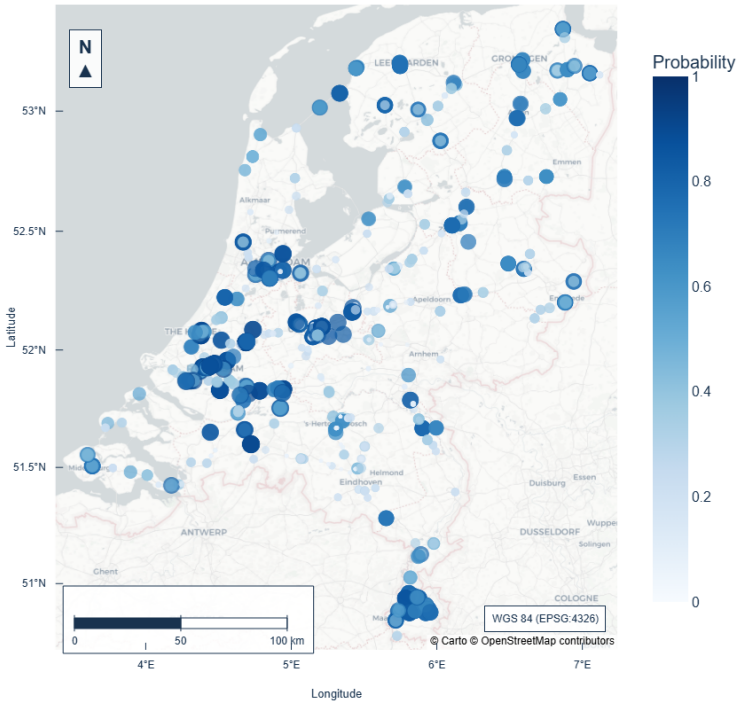

Figure S77: Probability heatmap for road ponding predictions – GAT-BLSTM

References

[1] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[2] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[3] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1412.6980.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[5] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems*, vol. 29, 2016, pp. 1019–1027.

[6] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012, ISSN: 1532-4435.

[7] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," ser. NIPS'12, Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 2951–2959.

[8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. DOI: 10.1023/A:1010933404324.

[9] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[10] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. DOI: 10.1111/j.1467-9868.2005.00503.x.

[11] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds., Heidelberg: Physica-Verlag HD, 2010, pp. 177–186. DOI: 10.1007/978-3-7908-2604-3_16.

[12] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[13] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002, Nonlinear Methods and Data Mining, ISSN: 0167-9473. DOI: https://doi.org/10.1016/S0167-9473(01)00065-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167947301000652.