



EMERALDS

| | |
|-----------------------|---|
| Project Title | Extreme-scale Urban Mobility Data Analytics as a Service |
| Project Acronym | EMERALDS |
| Grant Agreement No. | 101093051 |
| Start Date of Project | 2023-01-01 |
| Duration of Project | 36 months |
| Project Website | https://emeralds-horizon.eu/ |

D4.1 – Mobility Data Analytics and Learning Services 1st Version

| | |
|------------------------------|--|
| Work Package | WP 4, Extreme Scale Mobility Data Analytics & Learning |
| Lead Author (Org) | Anita Graser (AIT) |
| Contributing Author(s) (Org) | Anahid Jalali (AIT), George Theodoropoulos (UPRC), Ioannis Athanasopoulos (UPRC), Andreas Patakis (UPRC), Yannis Kontoulis (UPRC), Nikolaos Koutroumanis (UPRC), Christos Doulkeridis (UPRC), Nikos Pelekis (UPRC), Yannis Theodoridis (UPRC), Daniel Calvo (ATOS), Ignacio Elicegui (ATOS), Yerhard Lalangui (ATOS), Charlotte Fléchon (PTV), Christian Hinkelbein (PTV), Mattia Pretti (SIS), Luca Paone (SIS), Claudio Vicari (SIS), Stathis Antoniou (INLE) |
| Due Date | 31.03.2024 |
| Date | 29.03.2024 |
| Version | V1.0 |

Dissemination Level

- PU: Public
 SEN: Sensitive, only for members of the consortium (including the Commission)

Versioning and contribution history

| Version | Date | Author | Notes &/or Reason |
|---------|------------|--|---|
| 0.1 | 02/06/2023 | Anita Graser | TOC |
| 0.2 | 23/02/2024 | Anita Graser, Anahid Jalali, Georgios Theodoropoulos, Ioannis Athanasopoulos, Andreas Patakis, Yannis Kontoulis, Nikolaos Koutroumanis, Yannis Theodoridis, Christos Doulkeridis, Daniel Calvo, Ignacio Elicegui, Yerhard Lalangui, Charlotte Fléchon, Christian Hinkelbein, Mattia Pretti, Luca Paone, Claudio Vicari, Stathis Antoniou | Draft for internal review |
| 0.3 | 08/03/2024 | Anita Graser, Anahid Jalali, Georgios Theodoropoulos, Michael Madianos, Yerhard Lalangui, Mattia Pretti, Stathis Antoniou | First review comments addressed |
| 0.4 | 18/03/2024 | Anita Graser, Anahid Jalali, Georgios Theodoropoulos, Michael Madianos, Yerhard Lalangui, Mattia Pretti, Stathis Antoniou | Second review comments addressed |
| 0.5 | 22/03/2024 | Anita Graser | Addressed Scientific and Technical Manager Comments |
| 1.0 | 25/03/2024 | Anita Graser | Final draft for submission |

Quality Control (includes peer & quality reviewing)

| Version | Date | Name (Organisation) | Role & Scope |
|---------|------------|--|---|
| 0.2 | 05/03/2024 | Bahare Salehi (ULB), Ting Gao (TUD), Theivapraksham Hari (TUD), Foivos Galatoulas (INLE) | First Round Internal Review Comments |
| 0.3 | 12/03/2024 | Bahare Salehi (ULB), Ting Gao (TUD), Theivapraksham Hari (TUD), Foivos Galatoulas (INLE) | Second Round Internal Review Comments |
| 0.4 | 20/03/2024 | Yannis Theodoridis (UPRC) | Scientific and Technical Manager Review |
| 1.0 | 29/03/2024 | Foivos Galatoulas (INLE) | Final review by Coordinator |



**Funded by
the European Union**

This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101093051



Disclaimer

EMERALDS - This project has received funding from the Horizon Europe R&I programme under the GA No. 101093051. The information in this document reflects only the author’s views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided “as is” without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.

Copyright message

©EMERALDS Consortium. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation, or both. Reproduction is authorized provided the source is acknowledged.

Table of Contents

| | | |
|-------|---|----|
| 1 | Introduction..... | 10 |
| 1.1 | Purpose and Scope of the Document | 10 |
| 1.2 | Relation to Work Packages, Deliverables and Activities | 12 |
| 1.3 | Contribution to WP4 and Project Objectives..... | 14 |
| 1.4 | Structure of the Deliverable..... | 15 |
| 2 | Extreme Scale Mobility Data Analytics at the Edge/Fog/Cloud continuum | 16 |
| 2.1 | Probabilistic Trip Chaining | 17 |
| 2.1.1 | Brief Survey of the State-of-the-Art | 17 |
| 2.1.2 | Overview and Description | 18 |
| 2.1.3 | Preliminary Evaluation | 19 |
| 2.1.4 | Next Steps | 19 |
| 2.2 | Dropoff / Destination Prediction | 20 |
| 2.2.1 | Brief Survey of the State-of-the-Art | 20 |
| 2.2.2 | Overview and Description | 21 |
| 2.2.3 | Preliminary Evaluation | 23 |
| 2.2.4 | Next Steps | 25 |
| 2.3 | Trajectory Data Analysis | 26 |
| 2.3.1 | Brief Survey of the State-of-the-Art | 26 |
| 2.3.2 | Overview and Description | 27 |
| 2.3.3 | Preliminary Evaluation | 27 |
| 2.3.4 | Next Steps | 29 |
| 2.4 | Real-Time Extreme Scale Map Matching | 29 |
| 2.4.1 | Brief Survey of the State-of-the-Art | 30 |
| 2.4.2 | Overview and Description | 31 |
| 2.4.3 | Preliminary Evaluation | 33 |
| 2.4.4 | Next Steps | 34 |
| 3 | Active & Federated Learning over Mobility Data | 35 |
| 3.1 | Traffic State / Flow Forecasting | 36 |
| 3.1.1 | Brief Survey of the State-of-the-Art | 36 |
| 3.1.2 | Overview and Description | 37 |
| 3.1.3 | Preliminary Evaluation | 38 |
| 3.1.4 | Next Steps | 41 |
| 3.2 | Parking Garage Occupancy Prediction..... | 42 |
| 3.2.1 | Brief Survey of the State-of-the-Art..... | 42 |



- 3.2.2 Overview and Description 43
- 3.2.3 Preliminary Evaluation 44
- 3.2.4 Next Steps 51
- 3.3 Crowd Density Prediction 51
- 3.3.1 Brief Survey of the State-of-the-Art 52
- 3.3.2 Overview and Description 52
- 3.3.3 Preliminary Evaluation 53
- 3.3.4 Next Steps 59
- 3.4 Active Learning & XAI for Crowd Prediction 59
- 3.4.1 Brief Survey of the State-of-the-Art 59
- 3.4.2 Overview and Description 60
- 3.4.3 Preliminary Evaluation 62
- 3.4.4 Next Steps 68
- 4 Mobility AI-as-a-Service 69
- 4.1 Brief Survey of the State-of-the-Art 69
- 4.2 MAIaaS Platform Overview 71
- 4.3 MAIaaS Platform Components 74
- 4.3.1 Data Management 75
- 4.3.2 ML Models Development Framework 77
- 4.3.3 ML Models Deployment (Production) Framework 79
- 4.3.4 Federated Learning Module 81
- 4.4 Preliminary Evaluation 83
- 4.5 Next Steps 86
- 5 Conclusions and Next Steps 88
- Annex 89
- References 95

List of Figures

- FIGURE 1-1: D4.1 SOFTWARE COMPONENTS / EMERALDS AND THEIR ROLE IN MOBILITY DATA ANALYTICS AS A SERVICE.. 11
- FIGURE 1-2: PROCESS DIAGRAM SHOWING THE RELATIONSHIP BETWEEN THE DIFFERENT PHASES OF CRISP-DM..... 11
- FIGURE 1-3: THE MOBILITY DATA SCIENCE PIPELINE..... 12
- FIGURE 1-4: EMERALDS SERVICES POSITIONING ACROSS DATA PIPELINES AS PRESENTED IN D2.1 13
- FIGURE 1-5: EMERALDS TOOLSET IMPLEMENTATION PLAN AS PRESENTED IN D2.1 13
- FIGURE 2-1: DISTRIBUTION OF TRIPS PER HOUR OF DAY (LEFT) & HEATMAP OF ENTRY STOPS (RIGHT)..... 21
- FIGURE 2-2: DISTRIBUTION OF TRIPS PER EXIT STOP ID..... 22
- FIGURE 2-3: ACCURACY OF MODELS FOR DIFFERENT CLASS BALANCING THRESHOLD VALUES..... 25
- FIGURE 2-4: TRAJECTORY DATA AND TRAVEL TIME ANALYSIS RESULTS IN A JUPYTER NOTEBOOK 28
- FIGURE 2-5: SCREENSHOT OF THE ANALYSIS MODEL UNDER DEVELOPMENT FOR UC3 USING TRAJECTOOLS ALGORITHMS . 28
- FIGURE 2-6: RAW FCD DATA 30
- FIGURE 2-7: EXAMPLE OF A SINGLE DEVICE ID RAW FCD 31
- FIGURE 2-8: EXAMPLE OF ONE DEVICE CANDIDATE POINTS PATHS..... 31
- FIGURE 2-9: BENCHMARK NETWORK OF THE CITY OF ROME 33
- FIGURE 3-1: THE FRAMEWORK OF GRAPH MULTI-ATTENTION NETWORK 38
- FIGURE 3-2: SENSOR DISTRIBUTION IN NDW (LEFT) AND PEMS-BAY (RIGHT) DATASETS. 39
- FIGURE 3-3: DISTRIBUTION OF SPEED AND INTER-NODE CORRELATIONS ON PEMS-BAY AND NDW DATASETS..... 39
- FIGURE 3-4: DECOMPOSITION OF RANDOM SENSOR’S MEASUREMENTS FOR BOTH DATASETS..... 40
- FIGURE 3-5: VISUALIZATION OF THE HISTORICAL FORECAST AND HOW THE MODEL RE-TRAINING FUNCTIONS..... 44
- FIGURE 3-6. VISUALIZATION OF THE PRE-PROCESSED INPUT DATA FOR PARKING OCCUPANCY PREDICTION MODELS..... 44
- FIGURE 3-7: ILLUSTRATION OF THE TARGET VARIABLE DISTRIBUTION FOR ALL THREE PARKING GARAGES. 45
- FIGURE 3-8: NAIVE BASELINE 1 FOR PARKING GARAGE OCCUPANCY PREDICTION. 45
- FIGURE 3-9: NAIVE BASELINE 2 FOR PARKING GARAGE OCCUPANCY PREDICTION..... 45
- FIGURE 3-10: RESIDUAL ANALYSIS OF PARKLAAN XGBOOST MODEL. 46
- FIGURE 3-11: RESIDUAL ANALYSIS OF STRAND XGBOOST MODEL. 47
- FIGURE 3-12: XGBOOST FAILURE EXAMPLE FOR KURHAUS..... 47
- FIGURE 3-13: XGBOOST FAILURE EXAMPLE FOR PARKLAAN. 48

| | |
|--|----|
| FIGURE 3-14: FORECAST MODEL FAILURE CASCADE EXAMPLE FOR ALL THREE PARKING GARAGES..... | 48 |
| FIGURE 3-15: PREDICTION VS ACTUAL CLASSIFICATION LABELS OF KURHAUS (ADABOOSTCLASSIFIER)..... | 50 |
| FIGURE 3-16: PREDICTION VS ACTUAL CLASSIFICATION LABELS OF PARKLAAN (ADABOOSTCLASSIFIER)..... | 50 |
| FIGURE 3-17: PREDICTION VS ACTUAL CLASSIFICATION LABELS OF STRAND (ADABOOSTCLASSIFIER)..... | 50 |
| FIGURE 3-18: VISUALIZATION OF THE PRE-PROCESSED CROWD DATA..... | 53 |
| FIGURE 3-19. CROWD DENSITY BASELINE, WHICH ASSUMES THAT THE EXPECTED CROWD DENSITY FOR THIS WEEK IS THE SAME AS THE PREVIOUS WEEK..... | 55 |
| FIGURE 3-20: COMPARISON OF THE BEST (SCHEVENINGEN VOLLEDIG) AND WORST (BEACH STADIUM) RESIDUAL ANALYSIS OF XGBOOST..... | 55 |
| FIGURE 3-21: EXAMPLE OF VERY LOW PREDICTION ERROR FOR “SCHEVENINGEN VOLLEDIG” AREA, INCLUDING INPUT FEATURES..... | 56 |
| FIGURE 3-22: EXAMPLE OF HIGH PREDICTION ERROR FOR “BEACH STADIUM” AREA, INCLUDING INPUT FEATURES..... | 56 |
| FIGURE 3-23: EXAMPLE OF MISCLASSIFICATION FOR TOEGANG KURHAUS..... | 58 |
| FIGURE 3-24: EXAMPLE OF FALSE NEGATIVES FOR TOEGANG ZEEKANT..... | 58 |
| FIGURE 3-25: EXPLAINABLE INTERACTIVE AL FRAMEWORK CONCEPT..... | 61 |
| FIGURE 3-26: SHAP VISUALIZATION FOR PARKING OCCUPANCY PREDICTION OF PARKLAAN..... | 63 |
| FIGURE 3-27: ILLUSTRATION OF THE GLOBAL SURROGATE DECISION TREE ON PARKLAAN ADABOOST CLASSIFIER..... | 64 |
| FIGURE 3-28. LIME’S OUTPUT ON ONE TRUE POSITIVE SAMPLE FOR PARKLAAN..... | 64 |
| FIGURE 3-29: LIME’S OUTPUT ON ONE MISCLASSIFIED (FN) SAMPLE FOR PARKLAAN IN LATER HOURS OF THE DAY..... | 65 |
| FIGURE 3-30: SHAP VISUALIZATION FOR CROWD DENSITY PREDICTION OF SCHEVENINGEN VOLLEDIG..... | 66 |
| FIGURE 3-31: ILLUSTRATION OF THE GLOBAL SURROGATE DECISION TREE ON TOEGANG KURHAUS XGBOOST CLASSIFIER..... | 67 |
| FIGURE 3-32: LIME’S OUTPUT ON ONE CORRECTLY CLASSIFIED (TP) SAMPLE FOR TOEGANG KURHAUS..... | 67 |
| FIGURE 4-1: MLOps COMBINE MACHINE LEARNING, APPLICATIONS DEVELOPMENT, AND IT OPERATIONS. SOURCE: NEAL ANALYTICS..... | 70 |
| FIGURE 4-2: MAIAAS PLATFORM ARCHITECTURE – LOGICAL VIEW..... | 73 |
| FIGURE 4-3: ML LIFECYCLE IMPLEMENTATION V1..... | 74 |
| FIGURE 4-4: EMERALDS MAIAAS MINIO UI..... | 76 |
| FIGURE 4-5: EMERALDS MAIAAS DATA MANAGEMENT INTERFACES OVERVIEW..... | 76 |
| FIGURE 4-6: EMERALDS MAIAAS DATAHUB EXAMPLE WITH UC1 DATASETS..... | 77 |
| FIGURE 4-7: EMERALDS MAIAAS KUBEFLOW UI..... | 78 |
| FIGURE 4-8: MODEL DEVELOPMENT OVERVIEW..... | 79 |
| FIGURE 4-9: MODEL DEPLOYMENT OVERVIEW..... | 81 |
| FIGURE 4-10: MAIAAS FEDERATED LEARNING MODULE – LOGICAL VIEW..... | 82 |
| FIGURE 4-11: EXAMPLE OF A KUBERNETES’ CLUSTER MONITORING DASHBOARD CREATED WITH GRAFANA FOR EMERALDS MAIAAS KPIS..... | 85 |
| FIGURE 4-12: EVIDENTLYAI MONITORING DASHBOARD EXAMPLE (FROM EVIDENTLYAI.COM)..... | 86 |

List of Tables

| | |
|---|----|
| TABLE 1: TERMINOLOGY..... | 7 |
| TABLE 2 - MATRIX OF ALIGNMENT..... | 8 |
| TABLE 3: OVERVIEW OF EMERALDS AND MAIAAS PLATFORM COMPONENTS DEVELOPED IN WP4..... | 10 |
| TABLE 4: EMERALDS’ RELATION TO EXTREME SCALE / BIG DATA VS..... | 15 |
| TABLE 5: PROBABILISTIC TRIP CHAINING KPIS..... | 19 |
| TABLE 6: STATISTICS OF TICKET VALIDATION DATASETS..... | 21 |
| TABLE 7: FEATURES USED IN DROPOFF PREDICTION..... | 23 |
| TABLE 8: DROPOFF / DESTINATION PREDICTION EVALUATION RESULTS..... | 24 |
| TABLE 9: DROPOFF / DESTINATION PREDICTION KPIS..... | 25 |
| TABLE 10: TRAJECTORY DATA ANALYSIS KPIS..... | 29 |
| TABLE 11: EXTREME SCALE MAP MATCHING KPIS..... | 34 |
| TABLE 12: STATISTICS OF PEMS-BAY AND NDW DATASETS..... | 39 |
| TABLE 13: PERFORMANCE COMPARISON OF DIFFERENT APPROACHES FOR TRAFFIC PREDICTION ON PEMS-BAY AND NDW DATASETS..... | 41 |
| TABLE 14: COMPUTATIONAL SUMMARY ON THE PEMS-BAY DATASET..... | 41 |
| TABLE 15: TRAFFIC STATE / FLOW FORECASTING KPIS..... | 41 |
| TABLE 16: FORECAST MODEL PERFORMANCE COMPARED TO THE NAIVE BASELINES ON ALL THREE PARKING GARAGES..... | 46 |
| TABLE 17: CLASSIFIER PERFORMANCE ON ALL THREE PARKING GARAGES..... | 49 |
| TABLE 18. CONFUSION MATRICES OF THE THREE PARKING MODELS..... | 50 |
| TABLE 19: PARKING GARAGE OCCUPANCY PREDICTION KPIS..... | 51 |
| TABLE 20: CROWD DENSITY FORECAST SUMMARY RESULTS..... | 54 |



| | |
|---|----|
| TABLE 21: CROWD DENSITY PREDICTION CLASSIFICATION RESULTS SUMMARY | 57 |
| TABLE 22: CONFUSION MATRICES OF TWO CLASSIFIERS | 57 |
| TABLE 23: CROWD DENSITY PREDICTION KPIS | 58 |
| TABLE 24: ACTIVE LEARNING & XAI FOR CROWD PREDICTION KPIS..... | 68 |
| TABLE 25: KUBEFLOW CLUSTER RESOURCES..... | 73 |
| TABLE 26: APPLICATION AND TOOLS CLUSTER RESOURCES | 74 |
| TABLE 27: NFS SERVER RESOURCES | 74 |
| TABLE 28: MOBILITY AI-AS-A-SERVICE PLATFORM PERFORMANCE KPIS..... | 84 |
| TABLE 29: MOBILITY AI-AS-A-SERVICE PLATFORM USAGE KPIS..... | 84 |
| TABLE 30: SUMMARY OF EMERALDS PRESENTED IN D4.1 | 88 |

Terminology

| Acronym | Description |
|---------|--|
| AI/ XAI | Artificial Intelligence/ Explainable AI |
| AL | Active Learning |
| API | Application Programming Interface |
| CC | Computing Continuum |
| CI/CD | Continuous Integration / Continuous Deployment |
| DevOps | Development and Operations |
| DoA | Description of Action |
| FCD | Floating Car Data |
| FL | Federated Learning |
| GIS | Geographic Information System |
| GPS | Global Positioning Systems |
| GPU | Graphics Processing Unit |
| KPI | Key Performance Indicator |
| MAaaS | Mobility Analytics as a Service |
| MA(P)E | Mean Absolute (Percentage) Error |
| MAIaaS | Mobility Artificial Intelligence as a Service |
| ML | Machine Learning |
| MLOps | Machine Learning Operations |
| (RR)MSE | (Relative Root) Mean Square Error |
| NN/GNN | Neural Networks / Graph Neural Networks |
| OGC | Open Geospatial Consortium |
| OS | Operating System |
| OSM | OpenStreetMap |
| PoC | Proof of Concept |
| RA | Reference Architecture |

| | |
|------|----------------------------|
| SotA | State-of-the-Art |
| TRL | Technology Readiness Level |
| UC | Use Case |
| VA | Visual Analytics |
| WP | Work Package |

Table 1: Terminology

GA Matrix of Alignment

Table 2 outlines the outputs of D4.1 mapped to the GA commitments as stated in the Description of Action (DoA) Annex 1 and Annex 2.

| GA Components Title (and type) | GA Component Outline | Document Chapter(s) | Justification |
|---|---|---------------------|---|
| Deliverable | | | |
| D4.1 Mobility Data Analytics and Learning Services v1 | <i>Software library containing early versions of the extreme Mobility Data Analytics and Learning Services with embedded privacy-preserving tools and ingesting heterogeneous data types from each tier of the computing continuum accompanied by report consolidating release notes and instructions on executing the developed methods. A report will document the advancement of AI models compared to state-of-the art models and methods.</i> | Chapter 1-5 | <p>This document describes the early versions of the extreme Mobility Data Analytics and Learning Services emeralds.</p> <p>Chapter 1 puts them into context of the project goals and architecture, whereas Chapter 5 summarises and declares next steps towards v2 (to be documented in D4.2).</p> <p>Chapters 2-4 document the advancements over the state-of the art and reference the source code repositories which include readmes with release notes and instructions on executing the software.</p> |
| Tasks | | | |
| T4.1 Extreme Scale Mobility Data Analytics at the Edge/Fog/Cloud continuum | <i>The goal of this task is to facilitate the implementation of an analytics framework that will be deployed in-situ/at the edge. Since the main goal of the project is to create an urban data- oriented MAaaS toolset that can support real-time data-driven decision making, the existence of a framework that can analyse data-streams and provide valuable insights fast, that is without the need for lengthy data transfers from the edge to the cloud and vice-versa, is of great importance. Data-privacy is also a topic that will be addressed, with the framework leveraging data locality to enable applications where no piece of</i> | Chapter 2 | Chapter 2 presents the emeralds developed in T4.1. These Emeralds perform state-of-the-art analytics operations in the Edge/Fog/Cloud Computing Continuum. |

| | | | |
|---|--|-----------|---|
| | <p>sensitive data leaves the place from where it is harvested. However, the ample compute that is offered by the modern cloud infrastructure will also be taken advantage of, but only when and where that is achievable without the privacy-preserving or the real-time capacity of the system.</p> | | |
| <p>T4.2 Active & Federated Learning over Mobility Data</p> | <p>The goal of this task is to develop ML & AI tools that improve mobility data mining and extraction of trustworthy information. Particularly, pattern & anomaly detection methods for distributed spatial timeseries and advanced visual analytics & active learning prototypes. Existing federated machine learning approaches will be adapted to deal with distributed spatiotemporal timeseries data. Existing explainable AI approaches will be extended to support spatiotemporal data analysis for mobility applications. Moreover, existing AML approaches will be extended to enable AML for mobility AI tasks, such as pattern and anomaly detection. To this end, suitable spatiotemporal visualization capabilities will be developed that enable users to determine correct labels for mobility applications.</p> | Chapter 3 | Chapter 3 presents the emeralds developed in T4.2. These emeralds implement state-of-the-art timeseries modelling techniques for mobility prediction tasks, including explainable and active learning techniques. |
| <p>T4.3 Mobility AI-as-a-Service</p> | <p>This task bridges between the developments of WP4 and WP2's by implementing an AI as a Service (AlaaS) platform tailored to the requirements of mobility use cases. It will follow the emerging MLOps paradigm to allow creating end-to-end pipelines and workflows for the automation and industrialisation of ML models, guaranteeing reproducibility, transparency and trustworthiness as required by the emerging "AI Act". The platform will include a catalogue of ML models according to the needs of WP5 use cases that will be developed in collaboration with T4.2 and which will be interoperable with the European AI on-demand platform. Leveraging the functionalities of the AlaaS platform, ML models will be served "As a Service" and their performance will be continuously monitored to trigger periodic re-training (i.e., continual learning) when deviations are detected in terms of accuracy due to domain shift. Models will be also available to be downloaded and executed locally as part of T4.1 edge analytics framework.</p> | Chapter 4 | Chapter 4 presents the Mobility AI as a Service (MAlaaS) platform developed in T4.3, including and breakdown per platform component. the T4.2 |

Table 2 - Matrix of Alignment

Executive Summary

The objective of this deliverable is to present the first version of the “Mobility Data Analytics and Learning Services” developed in EMERALDS. These analytics and learning tools and services are internally known as “emeralds”, as introduced in previous deliverables, such as D2.1. The emeralds releases are provided as software repositories with release notes and instructions on executing the developed methods. This report accompanies the emeralds releases, v1. The key aspects covered in this report include thorough descriptions of the emeralds and how they advance the state of the art, both with respect to the mobility data science field as well as the practical application in the use cases, reflected by the scientific and technical KPIs.

EMERALDS’s vision is to design, develop and create an urban data-oriented Mobility Analytics as a Service (MAaaS) toolset, consisting of the proclaimed EMERALDS services, compiled in a proof-of-concept prototype, capable of exploiting the untapped potential of extreme urban mobility data. The toolset will enable the stakeholders of the urban mobility ecosystem to collect and manage ubiquitous spatiotemporal data of high-volume, high-velocity and of high-variety, analyse them both in online and offline settings, import them to real-time responsive AI/ML algorithms and visualise results in interactive dashboards, whilst implementing privacy preservation techniques at all data modalities and at all levels of a data workflow architecture. The toolset will offer advanced capabilities in data mining of large amounts and varieties of urban mobility data.

The emeralds developed in WP4 “Extreme Scale Mobility Data Analytics & Learning” include T4.1 emeralds that perform state-of-the-art analytics jobs in the Computing Continuum (i.e., at the Edge, the Fog, and the Cloud), T4.2 emeralds that enable active and federated learning over mobility data, and the T4.3 Mobility AI-as-a-Service platform. In total, this deliverable covers eight (8) emeralds, three (3) of which are also already leveraging the MAaaS platform.

In the process of developing the WP4 emeralds presented in this deliverable, multiple iterations have been performed between technical and use case partners to ensure proper business and data understanding, as necessary for data science developments according to industry standards (CRISP-DM). The key dataset insights are therefore included in the corresponding emeralds sections.

1 Introduction

1.1 Purpose and Scope of the Document

The purpose of D4.1 “Mobility Data Analytics and Learning Services 1st Version” is to introduce the software components (referred to as “emeralds”) that are being developed in WP4 “Extreme Scale Mobility Data Analytics & Learning”, as listed in Table 3. D4.1 provides a methodological overview, functional description of the developed software components, puts them in the context of the state of the art, and presents preliminary evaluation results, before laying out the next steps of development. The expected TRL of these first versions is around TRL3.

| Task / emerald | Maturity | Repository Link |
|--|------------------------|----------------------------------|
| Extreme Scale MDA at the CC (T4.1) | | |
| Probabilistic Trip Chaining ¹ | 1st version | Project's Github |
| Dropoff/Destination Prediction ² | 1st version | Project's Github |
| Monitoring and Forecasting Shared Mobility Demand ³ | To be reported in D4.2 | Under Construction |
| Trajectory Data Analysis ⁴ | 1st version | Project's Github |
| Real-Time Extreme Scale Map Matching | 1st version | Project's Github |
| Active & Federated Learning over Mobility Data (T4.2) | | |
| Traffic State / Flow Forecasting | 1st version | Project's Github |
| Parking Garage Occupancy Prediction ⁵ | 1st version | Project's Github |
| Crowd Density Prediction ⁶ | 1st version | Project's Github |
| Active Learning & XAI for Crowd Prediction ⁷ | 1st version | Project's Github |
| Active Learning for Risk Category Classification | To be reported in D4.2 | Under Construction |
| Federated Learning Models for Mobility Data | To be reported in D4.2 | Under Construction |
| Mobility AI-as-a-Service (T4.3) | | |
| Data Management ⁸ | 1st version | Project's Github |
| ML Models Development Framework ⁹ | 1st version | Project's Github |
| ML Models Deployment Framework ¹⁰ | 1st version | Project's Github |
| Federated Learning Module | 1st version | Project's Github |

Table 3: Overview of emeralds and MAIaaS platform components developed in WP4

The emeralds covered in this deliverable cover different mobility data analytics as a service aspects, as illustrated by Figure 1-1.

- The analytics emeralds address key mobility analysis challenges and datasets typically encountered in smart cities, including public transport ticketing data and vehicle trajectory data, including floating car data (FCD)

¹ Previously referred to as “Probabilistic Approach for Trip Chaining” in D2.1

² Previously referred to as “Trajectory/Route Forecasting and Origin/Destination Estimation” in D2.1

³ New emerald, not previously listed in D2.1

⁴ Previously referred to as “Trajectory Data / Travel Time Analysis” in D2.1

⁵ Previously referred to as “Parking garage occupancy forecasting” in D2.1

⁶ Previously referred to as “Crowd density forecasting” in D2.1

⁷ Previously referred to as “Active Learning & XAI for crowd/flow forecasting” in D2.1

⁸ “Combining Data Ingestion Interfaces” presented in D2.1 and a new “Data Repository”.

⁹ Previously referred to as “ML Experimentation Module” and “ML Training and Testing Module” in D2.1

¹⁰ Previously referred to as “ML Models (and Tools) Repo” and “ML Monitoring Tools” in D2.1

- The learning emeralds address essential mobility AI challenges [1] of destination prediction, traffic forecasting, and occupancy/crowd prediction, including explainability of model predictions.
- The Mobility AI-as-a-Service Platform (MAIaaS) provides essential infrastructure for the development of the analytics and learning emeralds, including interfaces and integrations with WP2 and WP3 components, as detailed in the following section.

Additionally, Figure 1-1 shows the connections to the use cases (UC1-3) and Early Adopter Demonstrators (EAD1-2).



Figure 1-1: D4.1 Software components / emeralds and their role in mobility data analytics as a service

The development of the ML emeralds follows the data science process consisting of iterative steps of business and data understanding, followed by data preparation and modelling and evaluation before eventual deployment of models, as laid out in the Cross-industry standard process for data mining CRISP-DM (Figure 1-2). Therefore, the respective emeralds' evaluation sections also include results of the business and data understanding stages.

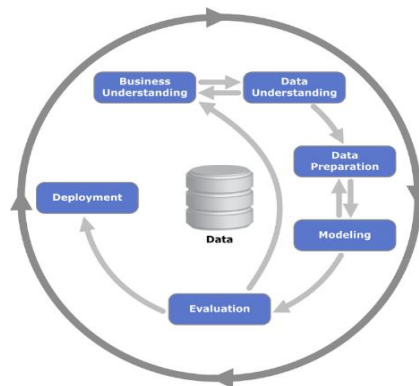


Figure 1-2: Process diagram showing the relationship between the different phases of CRISP-DM¹¹

¹¹ Image by Kenneth Jensen, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=24930610>

Important note on model selection and evaluation

One challenge evaluating machine learning solutions for mobility is that the **results are often very location- (city- or country-) specific** [2]. This is the case, since mobility behaviour (and resulting mobility patterns) are influenced by many factors, including social and geographic factors that differ from region to region. Hence, while one can look at different modelling approaches and different results presented in the literature, the available data (its quantity and quality) will be an important factor influencing what kind of models can be trained and the complexity of the observed mobility behaviour will influence the achievable prediction accuracy.

The KPIs of the emeralds presented here are a combination of scientific KPIs that aim to advance the scientific state of the art, as well as technical KPIs that aim to improve the analytical capabilities of the use case stakeholders. An earlier version of these KPIs has been presented in D2.1. In this deliverable, the KPIs have been further refined and synchronized with the use cases in WP5. Continued collaboration with WP5 will also be essential in the evaluation process of the upcoming 1st Assessment Cycle, which is described in the following section.

1.2 Relation to Work Packages, Deliverables and Activities

The emeralds presented in this deliverable focus on the Mobility Analytics step of the Mobility Data Science Pipeline [3], as shown in Figure 1-3, which aligns with the EMERALDS work package structure where WP3 provides Mobility Data Cleaning & Preprocessing, WP5 deals with Real World Applications and Data Collection, and WP2 provides the Data Management Infrastructures with Mobility Data Privacy being a key topic spanning all work packages.

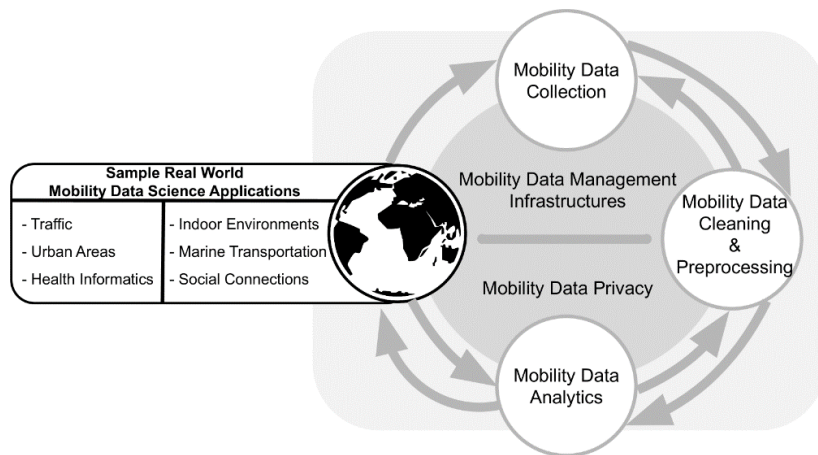


Figure 1-3: The Mobility Data Science Pipeline¹²

The emeralds comply with the EMERALDS reference architecture introduced in D2.1 “Reference Architecture” (Figure 1-4) and are validated and demonstrated through real-world data from selected representative use cases whose main context is outlined D5.1 “Use Cases Scoping Document”. To ingest heterogeneous data types from each tier of the computing continuum, the emeralds reported in this deliverable link to WP3/T3.1 emerald “Privacy-Preserving Data Ingestion”. Furthermore, the emeralds are designed to support end-to-end pipelines to facilitate the validation and demonstration as well as re-usability beyond the project scope.

¹² Image source: Mokbel et al. (2024) <https://arxiv.org/html/2307.05717v4>

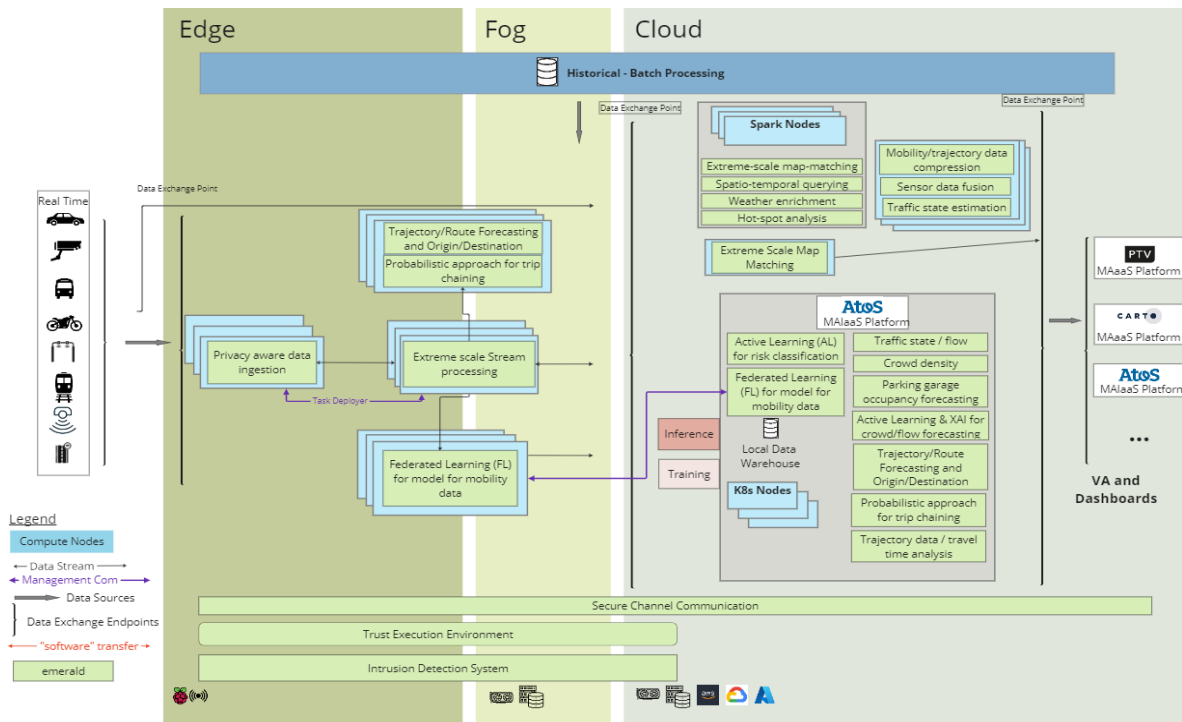


Figure 1-4: EMERALDS services positioning across Data Pipelines as presented in D2.1

The status of the emeralds reported in this deliverable is the result of the 1st implementation cycle, as shown in Figure 1-5 at M15. Therefore, the results of the 1st integration, 1st assessment cycle, and the 2nd set of cycles will be presented in the next deliverable D4.2. Throughout all upcoming cycles, the **collaboration framework** established within the project (with technical meetings and workshops conducted between technical partners and use case leaders) will be further intensified to enable iterative development and targeted innovations.

Tool Implementation Milestones

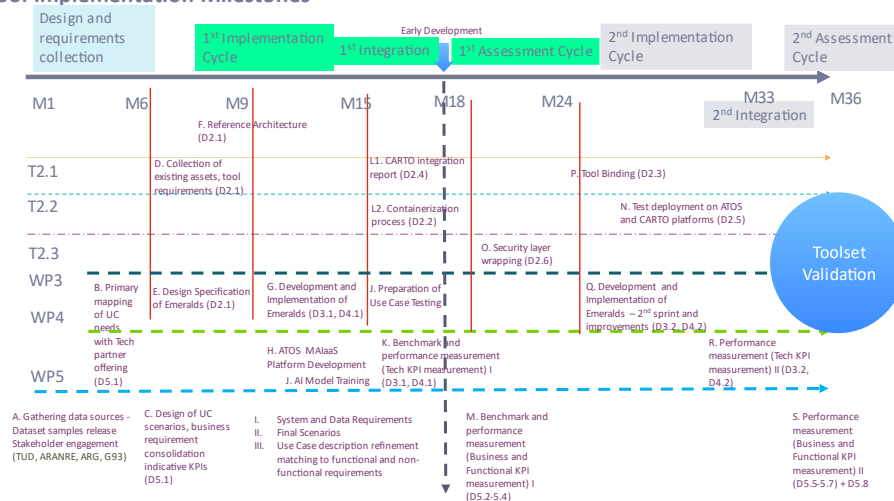


Figure 1-5: EMERALDS Toolset Implementation Plan as presented in D2.1

1.3 Contribution to WP4 and Project Objectives

This document is the key output of all tasks in WP4 in the first development year (project month 4 to 15). It contributes to the fulfilment of **Project Objective 3 (O3) “Develop mobility data analytics and AI/ML tools and services – MAaaS”** by developing location-aware analytics aimed at providing fast and accurate information along the CC.

The results of WP4 are part of the EMERALDS toolset, thus directly contributing to the achievement of **Project Objective 1 (O1) “Design a service-oriented reference architecture of a palette of services (‘emeralds’) for extreme scale urban mobility data analytics”**. The emeralds selected for development within WP4 and presented in the following chapters cover essential mobility data analytics and prediction tasks, that are commonly encountered in smart cities, as represented by the use case partners. Table 4 presents the key strengths of WP4 services in dealing with extreme data.

| | Volume | Velocity | Variety | Veracity | Visualization |
|--|---|---|--|--|-------------------------------------|
| Extreme Scale MDA at the CC (T4.1) | | | | | |
| Dropoff / Destination Prediction | Multiple large data streams | Real-time analytics: applicable to streaming data | Integration of various datasets: GPS tracks, weather, static data etc. | Handling of missing data & uncertainty | N/A |
| Monitoring and Forecasting Shared Mobility Demand | tba | tba | tba | tba | tba |
| Probabilistic Trip Chaining | Large historic data volumes | N/A | Integration of spatial timeseries from multiple sources | Handling of missing data & uncertainty | N/A |
| Trajectory Data Analysis | N/A | N/A | Support for a variety of different movement data formats | Supports the data quality assessment processes | Provides VA tools for movement data |
| Extreme Scale Map Matching | Large real-time streams and GBs of static street map data | GPS real-time feed rate to be 2k data points per second | Support for different street map data sources | Handling of interruptions / holes / and invalid data | N/A |
| Active & Federated Learning over Mobility Data (T4.2) | | | | | |
| Traffic State / Flow Forecasting | Large volumes of training data | Real-time analytics: applicable to streaming data | Integration of spatial timeseries from multiple sources | Handling of data quality issues | N/A |
| Parking Garage Occupancy Prediction | N/A | Real-time analytics: applicable to streaming data | Integration of spatial timeseries from multiple sources | Handling of missing data | N/A |
| Crowd Density Prediction | N/A | Real-time analytics: | Integration of spatial timeseries | Handling of missing data & uncertainty | N/A |

| | | | | | |
|---|-----|---|---|---|--------------------------------------|
| | | applicable to streaming data | from multiple sources | | |
| Active Learning & XAI for Crowd Prediction | N/A | N/A | Integration of spatial timeseries from multiple sources | Enables expert input to address potential data errors | Provides visualizations for AL & XAI |
| Active Learning for Risk Category Classification | N/A | N/A | Integration of spatial timeseries from multiple sources | Enables expert input to address potential data errors | Provides visualizations for AL & XAI |
| Federated Learning Models for Mobility Data | N/A | Real-time analytics: applicable to streaming data | Integration of spatial timeseries from multiple sources | Handling of missing data & uncertainty | N/A |

Mobility AI-as-a-Service (T4.3)

| | | | | | |
|--|---------------|---|-----|-----|---|
| Data Management | Large volumes | Real-time analytics: applicable to streaming data | N/A | N/A | N/A |
| ML Models Development Framework | Large volumes | Real-time analytics: applicable to streaming data | N/A | N/A | Provides development environment with visualization support |
| ML Models Deployment (Production) Framework | Large volumes | Real-time analytics: applicable to streaming data | N/A | N/A | N/A |
| Federated Learning Module | Large volumes | Real-time analytics: applicable to streaming data | N/A | N/A | N/A |

Table 4: Emeralds' relation to extreme scale / big data Vs

The results of WP4 also contribute to **Project Objective 4 (O4) "Demonstrate and measure the efficiencies of the novel Extreme Scale Analytics services through three pilot use cases"** by demonstrating the EMERALDS toolset's capabilities in extreme data analytics workflows in urban mobility scenarios across Europe.

1.4 Structure of the Deliverable

This deliverable is of type OTHER and, therefore, focuses on source code and object code for the emeralds developed in WP4. The source code and object code are provided in code repositories (i.e., GitHub repositories with well-documented README files- Annex 1). This document accompanies the source code and object code as well as provides an overview and the context for the emeralds' development in WP4. This deliverable is structured in three main chapters, each one corresponding to the WP4 tasks T4.1-T4.3. Inside each chapter, the developed emeralds are described and evaluated in dedicated sections. Finally, the document concludes with the conclusions and next steps, which are summarised in Section 5.

2 Extreme Scale Mobility Data Analytics at the Edge/Fog/Cloud continuum

This chapter presents the developed emeralds that perform state-of-the-art analytics operations at the Edge/Fog/Cloud Computing Continuum (CC) as part of the edge analytics framework. Pushing some of the computation that takes place when such operations are executed closer to where data is collected (in-situ) can significantly improve user experience and provide a more comprehensive value proposition overall, including concepts like enhanced user and data privacy (e.g., through data locality and data aggregation), improved response times etc. The emeralds reported in this chapter are the following:

1. **Probabilistic Trip Chaining:** This emerald utilizes public transport data in the form of ticket validation records to develop a probabilistic approach for chaining multiple trips taken by passengers that have used multiple transport means (for example, multiple buses) in sequence. This kind of analytics is essential to turn ticket validation records into reliable public transport demand information.
2. **Dropoff / Destination Prediction:** This emerald utilizes public transport data in the form of ticket validation records to forecast the drop-off stop for each individual passenger. The long-term goal of this offering is to accurately model the behaviour of public transport users and later utilize the model forecasts to assess travel time wasted and evaluate the quality of the transport system. This model helps to further improve the public transport demand information by filling in gaps.
3. **Trajectory Data Analysis:** This emerald offers essential data analytics capabilities for trajectory data to enable data understanding and modelling for analytics, as well as AI development. Examples of the analytics/processing jobs that are part of this emerald are, i) Outlier Detection, ii) Speed/Travel time computation, iii) Trajectory smoothing, iv) Visualization and much more.
4. **Real-Time Extreme Scale Map Matching:** This emerald implements a map matching offering that utilizes floating-car-data (FCD) to effectively estimate speeds on street network segments, thus turning the privacy-sensitive data of individual vehicles into anonymized traffic speed information.

The source code or object code of these emeralds can be found in the following repositories:

| Emerald | Repositories | Publications |
|--------------------------------------|---|--------------|
| 2.1 Probabilistic Trip Chaining | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/UC3-TripChaining_CrowdDensity/tree/main | |
| 2.2 Dropoff / Destination Prediction | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/Dropoff-Prediction/tree/main | |

| | | |
|---|--|---------------------------|
| 2.3 Trajectory Data Analysis | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/trajectools-ggis • https://github.com/emeralds-horizon/UC3-traveltime-analytics • https://github.com/emeralds-horizon/Cartoblog • https://github.com/emeralds-horizon/analytics-and-learning/tree/main/uc3-travel-time-analysis | ^{13, 14, 15, 16} |
| 2.4 Real-Time Extreme Scale Map Matching | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/WP4_Extreme_Scale_Map_Matching | |

2.1 Probabilistic Trip Chaining

This emerald utilizes public transport data in the form of ticket validation records to develop a probabilistic approach for chaining multiple trips taken by a single user. This way, this emerald provides a more robust way of computing drop-off stops of users that have used multiple transport means (for example, multiple buses) in sequence.

This emerald has different algorithmic components determining the entry stops, the intermediate exit stops and the final exit stops of each trip as each of these three case requires a different approach. The component determining the final exit stops performs engages from a similar principal as T4.1 emerald “Dropoff / Destination Prediction” but it uses a probabilistic rather than an ML approach. The probabilistic approach enables the inference of public transport passenger behaviour using expert domain knowledge (for example, by mirroring trips, as explained below). The remaining gaps in the dataset can then be filled by the ML approach or with the use of WP3 data imputation tools.

2.1.1 Brief Survey of the State-of-the-Art

Trip chaining is tackled with a variety of approaches in previous studies, also providing loose definitions on the notion of trip chains. This can be identified in the cases of aggregated trip chains, individuals trip chains, multi-mode chains and more. Herein, we adopt the definition [4], which examines the consecutive trips individuals make within a defined time period across different means of transport.

The works around trip chaining revolve around understanding individual travel patterns [5] in its different dimensions, for example how does trip chaining habits vary when considering different age groups or gender [6], different transportation modes or different periods (such as pre-pandemic vs post-pandemic) [7]. Many different modelling approaches have been used to tackle this problem. To name a few: optimization techniques [8], multivariate analysis [9], models from utility theory [10], k-means, SVM, and other ML algorithms [11].

One of the difficulties of trip chaining is that the **results of each work are hard to generalize as they are often very country- (or even city-) specific**. For instance, Schneider et al. [12] assess how much

¹³ <https://anitagraser.com/2024/01/12/trajectools-v2-in-the-works/>

¹⁴ Graser & Dragaschnig (under review) “Towards Low and No-Code Solutions for Movement Data Analytics” submitted to MDM2024.

¹⁵ <https://carto.com/blog/analyzing-mobility-hotspots-with-movingpandas>

¹⁶ <https://anitagraser.com/2023/11/28/analyzing-mobility-hotspots-with-movingpandas-carto/>

detour on bicycles are people willing to do within a chained trip. While these results have a direct impact on improving the transport network, they can only be applied to the city the data came from, in this case Delft. And this is because, as also mentioned in Schneider et al.'s paper, the results strongly differ depending on parameters that are city-specific. Similar examples can be found in Hensher & Reyes [13], where it is assessed if trip chaining is a barrier to public transport use, but again, its result concern Australia, and Lunke & Oystein [14], where the study explores how various factors influence the choice between public transport and car on daily trip chains for Oslo. Hence, while one can look at different modelling approaches and different results presented in the literature, the available data will usually be the guide of what can be deduced and how.

2.1.2 Overview and Description

An inherent difficulty in the works mentioned in the state of art is whether they are able to link and process relevant data sources to create a complete dataset. This is precisely the goal of this emerald: starting with the ticket validation files, use a series of algorithms and probabilistic approaches to create a complete dataset of chained trips.

This emerald introduces a new approach by estimating entry-exit combinations for a larger number of trips using a combination of probabilistic methods, new distance metrics calculations and algorithm optimization. The tool is validated with datasets originating from UC3 (T5.4, D1.4) and aligns with the objectives and functionalities described in D5.1.

The emerald's output will be used as an input by the "Crowd Density Prediction" emerald. Namely, the resulting dataset of chained trips will be used to predict the number of people by summing the number of validations per stop, per segment or per any other route-related variable. Crowd prediction will also take advantage of the data enrichment provided by emerald "Sensor Data Fusion". It will start as soon 'Sensor Data Fusion' can be applied to UC3 data and the chained trip dataset can be enriched.

To facilitate users in replicating the models' results, this component is offered as a set of Jupyter notebooks which can be used to create a validation file where missing entry and intermediate stops have been computed. More precisely:

- **_A_DataPrep** is the data preparation file. It creates Validation, GTFS and Vehicle events files by connecting different datasets and calculating new needed variables (for example new codes or clarified information related to events, validations, vehicles, schedules, or routes). These files are the base of subsequent analysis.
- **_B_EntryInterStops_Euclidian** uses the datasets created in step A and calculates the entry and intermediate stops for the validation data dated 25.09.2018 using Euclidian distance (done)
- **_B_EntryInterStops_Harvesine** uses the datasets created in step A and calculates the entry and intermediate stops for the validation data dated 25.09.2019 using the Haversine distance which is more precise than Euclidian distance as it takes into account the curvature of Earth's surfaces (done)
- **_B_EntryInterStops_Network** uses the datasets created in step A and calculates the entry and intermediate stops for the validation data dated 25.09.2019 using Network distance (work in progress)

The first step of the Trip Chaining algorithm is to determine the validation coordinates (col "val_coord" in blue) by finding the closest (in terms of time) "Door opens" event to the validation time (col "Time" in green) and extracting the corresponding coordinates of the event. Using the validation coordinates and the route information, the entry stop is then determined as the closest stop to the validation coordinates. The entry stop_id together with all entry-related information can be seen in blue.

Then, the second part of the algorithm determines the intermediate exit stop coordinates. It does so by looking at the route of the next entry stop and finding the stop with is closest to the entry stop. All intermediate exit-stop related information is shown in purple.

The resulting file contains all the stops of the trip except for the last exit stop (see missing datapoints of the last row in purple). As already mentioned, the missing final exit stop of this chained trip will be determined either probabilistically or using emerald “Dropoff / Destination Prediction”.

2.1.3 Preliminary Evaluation

The incorporation of this emerald in the data workflow of an end-user bears the potential to improve trip chaining in identifying more entry, intermediate, and exit stops. Currently >50MB are used to validate the existing algorithms’ performance. Its evaluation will be further reported in D5.6, where the tool will be integrated in the process of deriving public transport insights with additional use case data resources.

The following table summarizes all KPIs relevant to this emerald, including KPIs previously presented in D2.1 as well as two additional KPIs on percent point improvements for entry and exit stops. The achieved values will be determined in the 1st assessment cycle, after the ongoing validation of the model results has been completed, as detailed in the next steps section.

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|---|----------------|--------------|--|---|
| Successful identification of complete chained trips over current algorithm. ¹⁷ | 0 | Over 55% | Comparison with result on ground truth dataset | Tbd in the 1 st assessment cycle |
| Performance metrics, Ingestion throughput and input data size. | - | - | To be defined | Tbd in the 1 st assessment cycle |
| Percent point improvement over current algorithm’s entry stops estimation percentage rates. | 60 | Over 65% | Comparison with result on ground truth dataset | Tbd in the 1 st assessment cycle |
| Percent point improvement over current algorithm’s exit stops estimation percentage rates. | 40 | Over 45% | Comparison with result on ground truth dataset | Tbd in the 1 st assessment cycle |

Table 5: Probabilistic Trip Chaining KPIs

2.1.4 Next Steps

In the ongoing 1st integration cycle, we will complete our validation with respect to entry and intermediate stops and crosscheck the results with existing algorithms from UC3 partners. These results will be used as training data for T4.1 emerald “Dropoff / Destination Prediction”. In the following 1st assessment cycle, we will assess the KPIs listed in Table 5.

In the 2nd implementation cycle, the exit stops will be added (potentially using results of emerald “Dropoff / Destination Prediction”). The addition of exit stops will allow us to have the entire trip chaining information and produce a complete Validation file. This file will be enriched by the T3.3 emerald “Sensor Data Fusion” and used as an input for the T4.2 emerald “Crowd Density Prediction”.

¹⁷ Previously referred to as „Trip Chaining accuracy” in D2.1



2.2 Dropoff / Destination Prediction

This emerald utilizes public transport data in the form of ticket validation records to forecast the drop-off stop for each individual passenger. The long-term goal of this offering is to accurately model the behaviour of public transport users and later utilize the model forecasts to assess travel time wasted and evaluate the quality of the transport system.

Public transport (PT) is the cornerstone of any modern city. Large metropolitan areas with millions of inhabitants need to provide effective and ample public transport means to increase the living standards of the citizens and promote economic growth. Consequently, predicting the exit stops of public transport users is important for several reasons. To begin with, predicting the destination of passengers can help optimize the allocation of resources and improve the service quality of public transport systems, with the operators adjusting the supply of vehicles, seats, and staff to match the passenger flow and reduce waiting time, crowding, and operational cost. Moreover, it can help support the planning and management of urban transportation and land use, since planners and managers can analyse the travel patterns and preferences of users and design the transportation network accordingly. This can promote the integration of public transportation and urban development, and achieve the goals of sustainability, equity, and livability. As the literature shows, Machine Learning (ML) and Artificial Intelligence (AI) have proven to be the go-to concepts for forecasting tasks when historical data with correlated attributes can be used. Lastly, it is important to note that the usage of CC devices that are classified as Edge/Fog is an important part of this Emerald because such devices enable scalability, cost-effectiveness and easy integration.

2.2.1 Brief Survey of the State-of-the-Art

Costa et al. [15] work on predicting the destination of passengers in urban public transport systems using smart card data. The authors apply three different models (Top-K, NB, and J48) to different groups of travellers based on their age or economic conditions and compare their accuracy and performance. The paper uses data from Porto, Portugal, as a case study, and analyses more than 90 million trips recorded from two main transport providers in the city. The paper concludes that it is possible to predict the journey's destination based on the past with an average accuracy rate that varies from 20% to 65%, depending on the model and the group of travellers.

Zuniga et al. [16] propose a new methodology to estimate, update, and forecast the origin-destination (OD) matrices of passengers on a public transport corridor. The authors use historical data and real-time information from intelligent transportation systems, such as the fare system, to infer the exit stops of passengers. The methodology consists of two parts: an estimation algorithm based on a Bayesian approach and a prediction algorithm based on artificial neural networks. The authors test the methodology using data from the Metro of Valparaiso corridor in Chile and compare it with a static approach. The results show that the proposed methodology can improve the accuracy and performance of OD matrix estimation and prediction and support the optimization and control of public transport systems.

Zhao et al. [17] proposes a novel framework to predict the destination of passengers on urban public transport systems using smart card data. The authors argue that the destination of a passenger is influenced by multiple factors, such as the user's own travel preference, the crowd's travel preference, and the region's characteristics under certain spatiotemporal contexts. The framework, called MDLF, consists of two main components: a feature extraction component that extracts features from multiple and complementary views, and a prediction component that uses a recurrent neural network to calculate a moving trend score for each possible destination. The authors evaluate the framework using two real-world datasets (from Shenzhen & Shanghai) and compare it with existing Bayesian, Ensembling and Recurrent Neural Network based methods. The results show that MDLF can achieve higher accuracy and performance than the competitors, especially for occasional trips with strong

randomness and uncertainty. The paper contributes to the field of destination prediction by providing a data-driven and multi-view-based approach that can handle the complexity and diversity of urban public transport systems.

Zhang et al. [18] present a novel framework for predicting an individual’s next trip in public transportation systems, given their historical trip sequences and the time when the prediction is made. The paper addresses a problem that has not been well studied in the literature and is important for applications such as proactive travel recommendations and mobility management. The paper introduces a deep learning model called DeepTrip, which uses a trip sequence-to-sequence structure with an attention mechanism to capture the complex spatiotemporal and historical dependencies of individual mobility. The paper also develops a new method for representing continuous travel attributes, such as trip time and duration, using an overlapped embedding model that preserves both the categorical and numerical features of the data. The paper evaluates the performance of DeepTrip using trip data in urban rails and shows that it outperforms existing models by a significant margin. The paper also analyses the impact of different factors, such as prediction time, travel status, and data representation, on the prediction accuracy.

2.2.2 Overview and Description

We tackle the problem of predicting the drop-off stop of a public transport passenger’s trip. In such a setting, a wide array of data points can be made available depending on the ticketing platform that is put in place. Smart cards are a prime example of such a platform, supplying us with valuable information regarding the entry stop of the passenger, the time of ticket validation, the route etc.

This emerald utilizes ticket validation data from the city of Riga, supplied to us by Riga Traffic Company (Rigas Satiksme) assisted by our partner Grupa93 (G93) through EMERALDS use case #3 (UC3). The data studied in this reporting period (M4-M15) include validations from two individual days, the 7th and the 11th of September 2021. Table 6 presents a brief set of metrics regarding the dataset, with Figure 2-1 presenting the distribution of trips per time of day.

Riga Traffic Company – Ticket Validation Data (7 & 11 Sept 21)

| | |
|--------------------------------------|---|
| Number of validations: 161097 | Number of passengers (individual smart cards): 87835 |
| Number of routes: 73 | Number of individual entry/exit stops: 719/712 |

Table 6: Statistics of ticket validation datasets

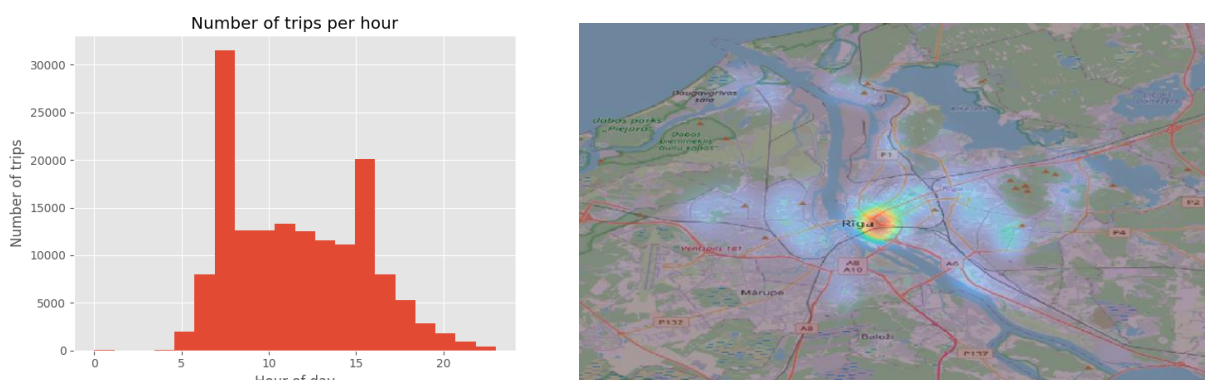


Figure 2-1: Distribution of Trips per hour of day (left) & Heatmap of entry stops (right)

Figure 2-1 (left) illustrates a clear picture with respect to the temporal trend that is followed on a daily basis, with ticket validations spiking when most people commute to and from work (around 7am and 4pm, respectively). Figure 2-1 (right) shows that the center of Riga is by far the most populated area where people validate tickets.

Based on the aforementioned analysis, we put in place a set of preprocessing steps that need to be executed for the dataset to be ready to be fed to our ML methods. These steps are:

1) **Feature Encoding:** Our dataset contains several useful features about each ticket validation. These features include the route ID, the validated ticket ID, the vehicle ID, etc. To enforce uniformity, we encode all these features with values between 0 and No. of classes-1. The following is an example of this transformation, where stops with ID “1”, “2” and “6” as swapped with a new encoded unique ID (1,2,2,6) -> (0,0,1,2).

2) **Per Route Stop Encoding:** In its initial state, the column named “exit_stop_id” contains the unique identifier of the exit stop of each validated ticket. That unique identifier is put in place to differentiate between all the available stops regardless of route. Essentially, if that column was left as is, the resulting model could predict that a passenger would exit at a stop that is not part of the route of the specified bus line, affecting the overall result. To avoid this, we swap to a per route unique identifier for each stop, essentially forcing the implemented model to only predict stops that are part of the specified route. To achieve this, we group all records based on the “route_id” column and then apply the same encoding routine as described in [step 1](#).

3) **Temporal Features:** In this step, we use the supplied timestamp of each validation to extract the following: i) hour of day (0-23), ii) minute of hour (0-59) and iii) second of minute (0-59).

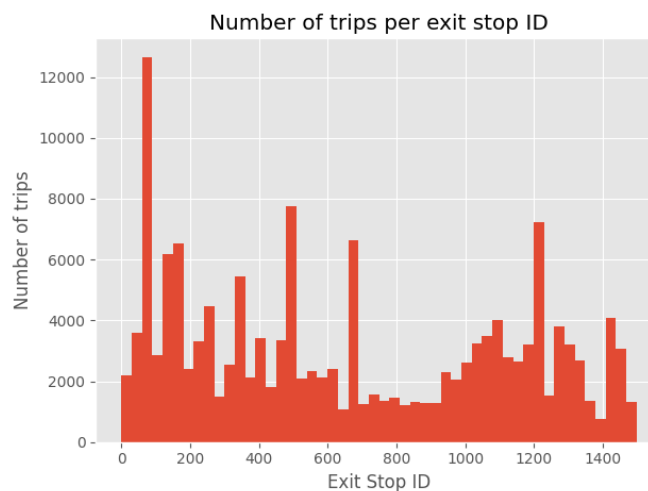


Figure 2-2: Distribution of Trips per Exit Stop ID

4) **Class Balancing:** This is the final and one of the most important steps of our preprocessing pipeline. As illustrated in Figure 2-2, some exit stops appear much more frequently than others, with extreme examples being the most frequent exit stop that appears more than 12000 times while 59 other stops appear just once. This skewedness can create a lot of problems for our model and thus, needs to be addressed. Our solution is a commonly used one that entails removing all the trips/validations that correspond to exit stop IDs that appear very infrequently (based on a threshold). However, given that the data sample that is available is not large (161K trips in 2 days) we need to be conservative with the threshold value we select, otherwise we risk discarding a large percentage of the overall dataset and possibly compromising our proposed model. That is why we define this value as a percentage, discarding the exit stops that appear less than X% of the number of appearances of the most frequent stop.

The full list of features is listed in Table 7.

| Feature Name | Description |
|--------------------|---|
| GarNr | Unique ID of bus |
| Route | Unique ID of route |
| minute | Minute of hour (0-59) |
| hour | Hour of Day (0-23) |
| second | Second of Hour (0-59) |
| ValidTolonaID | Unique ID of ticket |
| TrafficCompanyCode | Unique ID of the traffic company of the bus/route |
| Stop_id | Unique ID of entry stop |
| Stop_lon, Stop_lat | Coordinates of entry stop |
| Exit_stop_id | Unique ID of exit stop |

Table 7: Features used in Dropoff Prediction

The source code used in this emerald is hosted in the EMERALDS GitHub repository. Python3 is the programming language used, with packages like Pandas and Numpy providing all the needed functionality for the preprocessing.

2.2.3 Preliminary Evaluation

In our study, we evaluate 3 alternative ML methods/models, as follows:

1. **Random Forest** classifier (RF), an ML algorithm that combines the output of multiple decision trees to reach a single result, using bagging and feature randomness to reduce overfitting and bias [19].
2. **XGBoost** classifier (XGB), an ML algorithm that uses gradient boosting to train an ensemble of decision trees that can classify data into different classes [20].
3. **LightGBM** classifier (GBM), an ML algorithm that uses gradient boosting to train an ensemble of decision trees that can classify data into different classes, using novel techniques such as Gradient-based One Side Sampling and Exclusive Feature Bundling to improve the efficiency and accuracy of the model [21].

These ML methods have been selected because they all employ ensembling, essentially training multiple classifiers (hundreds) and combining them to create a more robust and comprehensive overall model. Additionally, these models have been proven to be very resource-efficient and data-efficient, meaning that they can be highly effective even if input data and computation resources are limited. This nicely fits our use case, since obtaining large amounts of historical ticket validation data is hard for most European cities while the available infrastructure in most use cases is rarely comprised by high-power GPU clusters that would be needed to efficiently train much larger Deep Learning models for example.

It is also important to note that this emerald also contributes to the overall project KPI of performing processing and analytics in situ, as its main goal is to be efficient enough to be applicable to low-resource environments.

The models are implemented, optimized, and evaluated using Python3. For RF, the scikit-learn implementation is used whereas native packages that carry the respective name are used for XGBoost and LightGBM. Regarding hardware, 3 types of compute devices are used in this evaluation, 1) an

Apple M2 Max SoC with a max TDP of around 60 Watts, 2) an Intel i7 based NUC with a max TDP of around 40 Watts and 3) a Raspberry Pi 4 with 8GB of RAM and a max TDP of less than 10 Watts. All of these devices can be classified as Edge/Fog, because of their energy consumption being less than 100 Watts, with the Raspberry Pi being an Edge-first device with minimal energy consumption and very low cost.

Table 8 presents a set of metrics for each Device-Model pair for a fixed Class Balancing Percentage (5%). Additionally, all models have been trained and evaluated under an **80/20** train/test protocol. The models have been tuned using exhaustive grid search, with their best combinations being the ones shown here. The following metrics have been selected because they directly address KPIs that were included in D2.1 and are essential for this Task/Work Package:

- 1) **Fit time**: The time needed for each model to be trained. Since the size of the training dataset is greatly affected by both the class balancing factor and the train split percentage (80%), we note that the time presented here corresponds to a training dataset size of approximately **75000 records**.
- 2) **Inference time**: The time needed for each model to provide a prediction for a single ticket validation record.
- 3) **Model Size**: The approximate size of each model in memory.
- 4) **Accuracy**: The average accuracy of each model (the percentage of exit stops that the model has accurately predicted over its test set).

| Device type | Device | Model | Fit time [s] | Inf. Time / Record [μs] | Model Size [MB] | Accuracy % |
|-------------|----------|-------|----------------------|-------------------------|-----------------|------------|
| Edge / fog | M2 Max | XGB | 11.67 | 13.02 | 20.80 | 30.1 |
| | | GBM | 35.10 | 18.88 | 18.28 | 30.3 |
| | | RF | 6.73 | 29.96 | 4582.0 | 32.4 |
| | i7 NUC | XGB | 15.12 | 17.41 | 20.78 | 29.4 |
| | | GBM | 8.11 | 32.76 | 18.28 | 29.8 |
| | | RF | 13.09 | 81.76 | 4575.5 | 32.5 |
| Edge | RPi4 8GB | XGB | 158.98 | 114.43 | 20.83 | 29.8 |
| | | GBM | 61.67 | 621.14 | 18.29 | 30.3 |
| | | RF | Out of Memory | | | |

Table 8: Dropoff / Destination prediction evaluation results

The key takeaways (in order of importance) of Table 8 are:

- 1) The Random Forest model is orders of magnitude larger than XGBoost and LightGBM in terms of size in memory. This introduces problems when the underlying infrastructure cannot support its size, which is what happened with Raspberry Pi 4. Even though the device we used includes 8 GB of RAM, with the inflated model and the respective data not fitting into its memory, it caused a crash. Hence, there are no performance figures for RF in RPi.
- 2) In all cases, accuracy hovers around 30% which has to be interpreted in the context of multi-label prediction, meaning that we successfully predict the exit stop 3 out of 10 times, keeping in mind that we have to pick the correct exit stop out of 10s of available stops per route (37 in the worst case scenario). While RF is better at 32.5%, that <10% improvement comes at the cost of size, as previously mentioned.

- 3) While both M2 Max and i7 NUC provide excellent results in terms of performance in fitting and inference given their computational capabilities and cost, the very energy efficient and cheap Raspberry Pi 4 is also highly capable, being able to output approximately 9,000 predictions per second using the XGB model. This means that even such a low spec device can support real-time forecasting for large-scale workloads.
- 4) GBM is faster than XGB at fitting, while XGB is faster than GBM at inference, except for the M2 Max where the underlying codebase is not optimized for ARM.

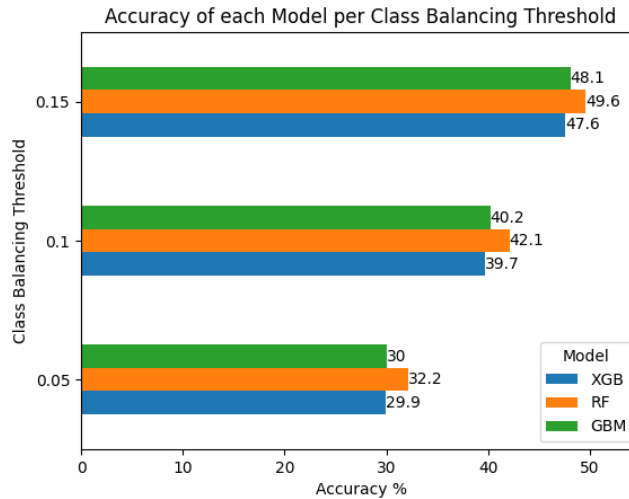


Figure 2-3: Accuracy of Models for different class balancing threshold values

Figure 2-3 shows the effect that the two variables, M and N, regarding class balancing has to the accuracy of each model. For Larger values, all models produce higher quality results, something that is to be expected since the resulting dataset becomes more homogeneous as this value increases.

The following table summarizes the KPIs relevant to this emerald. The “Real time forecasting” KPI previously described in D2.1 has been further refined to measure the performance with respect to fit and inference times. The achieved values will be determined in the 1st assessment cycle, after the baseline values have been established, as detailed in the next steps section.

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|--|----------------|---|--|-----------------------------|
| Prediction accuracy/ Performance | SotA | 20% speed up with the same or better accuracy | Error metrics / Fit & inference times | N/A |
| Number of object/s that forecasting has been applied to | N/A | 10,000 objects per run | Automated tracking and logging of forecasting applications | 6,000 objects in single run |

Table 9: Dropoff / Destination prediction KPIs

2.2.4 Next Steps

During the ongoing 1st integration and the 1st assessment cycle that will follow it, we will focus on two different aspects:

1. The first one is the comparison with State-of-the-Art. Due to our methodology being tailored for the Edge/CC, it is not as straightforward identifying competitors with a similar focus.

However, we plan to evaluate our offerings with respect to overall SotA in destination prediction and showcase the efficiency and quality of our approach while considering metrics like size of mode, inference times, etc.

2. The second one is related to the experimentation with large datasets. To evaluate our approach more effectively, we need to incorporate larger datasets that span multiple days, months etc. This way, our methodology will be tested in a more extensive real-world scenario that allows for more detailed training that will in turn create more comprehensive and complete models.

During the 2nd implementation, integration and assessment cycle, new features will be engineered and added when such datasets are available using external data sources like weather or event information etc. (exploiting the respective emeralds developed in WP3).

2.3 Trajectory Data Analysis

This emerald offers essential data analytics capabilities for trajectory data to enable data understanding and modelling for analytics as well as AI development. Examples of the analytics/processing jobs that are part of this emerald are i) Outlier Detection, ii) Speed/Travel time computation, iii) Trajectory smoothing, iv) Visualization and much more.

These emerald leverages and contributes to one of the leading open-source Python libraries for trajectory data analytics: MovingPandas¹⁸ which represents a background asset from partners' previous works. MovingPandas focuses on rapid prototyping, data exploration, and extreme flexibility, providing flexible interoperability with a large variety of spatial data formats.

2.3.1 Brief Survey of the State-of-the-Art

In recent years, multiple research groups have been working on scientific software libraries for movement data analytics in Python [22,23,24], R [25], and other languages [26]. However, these libraries are still rather fragmented (no single library covers all use cases) and their usage requires significant programming skills. This makes them difficult to learn and limits their use, particularly in interdisciplinary settings. Furthermore, the visualization capabilities of these libraries are limited [27] which presents a further challenge since the mobility and “transportation domain is characterized by both complex data and complex problems, which calls for visual analytics approaches” [28].

Geographic information systems (GIS) provide strong capabilities for analysing and visualizing spatial data and are therefore often used to work with movement data and related geographic context data in the mobility and transport domain. GIS-based no-code solutions such as the ArcGIS Model Builder¹⁹, the CARTO Workflow design tool²⁰ or the QGIS Model Designer²¹ are popular ways to create reproducible workflows for spatial data analysis, for example for street network quality evaluation [29], seismic microzonation analysis [30] or landslide prediction [31]. However, so far, these no-code solutions do not extend to spatiotemporal movement data. Other widely accessible visual analytics systems' capabilities are also very limited with respect to spatiotemporal data. Therefore, to achieve “wider and better understanding of the environmental, economic, and societal processes, their interrelations, and effects”, “researchers should not only strive for advancing the research but also take the responsibility for transferring the operational knowledge to practitioners and casual analysts.” [32]

¹⁸ <https://movingpandas.org>

¹⁹ <https://pro.arcgis.com/en/pro-app/latest/help/analysis/geoprocessing/modelbuilder/what-is-modelbuilder-htm>

²⁰ <https://carto.com/workflows>

²¹ https://docs.qgis.org/3.28/en/docs/user_manual/processing/modeler.html

The integration of movement data analysis libraries into GIS-based no-code solutions is hindered by a mismatch between movement data models for trajectories and classic GIS data models following the OGC Simple Features standard [33]. Simple Features defines point, line, and polygon features that combine geometry and arbitrary attributes, but the temporal dimension is not standardized. Mobility data specific standards, such as OGC Moving Features [34], have not seen wide adoption yet [35].

2.3.2 Overview and Description

The following steps have been taken to support the needs of the UCs and bring trajectory-specific analytics capabilities to the EADs:

- Development of a **novel rapid prototyping / no-code solution for trajectory data analysis** to provide transport analysts with a no-code environment for prototyping and to facilitate co-creation/co-production between data scientists and domain experts. This solution, called “**Trajectools**”, is being realized as a plugin for the **open-source geographic information system QGIS** (Figure 2-5) which brings MovingPandas functionality to the QGIS Processing Toolbox and Model Designer (as shown in Figure 2-5). The development of this plugin has been started with a focus on the algorithms needed for UC3 and further algorithms will be added throughout the project.
- The **Carto Platform (EAD2)** provides a no-code workflow design tool similar to the QGIS Model designer. Preliminary investigations into integration opportunities with Carto have however revealed significant engineering challenges, including the fact that Python libraries (such as MovingPandas) cannot be used directly in the workflow design tool. However, a demonstration of trajectory analysis with MovingPandas and hotspot analysis and visualization in Carto has been developed in a Carto data science notebook²².
- Development of a **trajectory data analytics-ready environment** for the **MLOps platform**. This novel trajectory analysis environment has already been used to develop trajectory analysis notebooks for data understanding and public transport travel time analysis in UC3 (Figure 2-4). This environment provides the following features and will continue to be refined throughout the project:
 - Trajectory analytics: travel time and speed calculation, trajectory generalization, outlier detection, stop detection, trajectory splitting, overlay operations (using the open-source MovingPandas library)
 - Public transport schedule analytics: routes, travel times and speeds (using the open-source `gtfs_functions`²³ library)
 - Static and interactive trajectory visualizations for notebook environments and data apps (as shown in Figure 2-4)

2.3.3 Preliminary Evaluation

Individual algorithms of the trajectory analysis toolbox (in MovingPandas and/or Trajectools) can be evaluated with regards to their performance in terms of execution time, as previously reported in D2.1. In this regard, we can report the following improvements:

- **Performance measurement of the trajectory cleaning:** Where the previous code took 7.5 minutes for 100 trajectories (of the Porto Taxi dataset from Kaggle), it now takes only 3.5

²² <https://docs.carto.com/data-and-analysis/carto+-python>

²³ https://github.com/Bondify/gtfs_functions

seconds. And 34k trajectories can be processed in 16 minutes. The new OutlierCleaner has been implemented in: <https://github.com/movingpandas/movingpandas/issues/333>.

With regards to usability improvements and time from first analysis idea to first results, we can report the following improvements:

- The trajectory / travel time analysis for public transport trajectories which used to require writing of custom Python code is now feasible in the novel no-code environment. For example, Figure 2-4 shows Interactive trajectory data visualization in a Jupyter notebook, consisting of a spatial / map view and a plot comparing observed public transport speeds (computed from trajectories) and scheduled speeds (based on the public transport schedule GTFS) while Figure 2-5 shows the analysis workflow model implemented in the no-code platform.

```
In [50]: tt_plot = by_hour_obs.hvplot(title=f'Speed {SEGMENT_ID}', y='mean_speed_kmh', x='t_', label='Observed speed', ylim=(0,50), f
map_plot = SEG.hvplot(alpha=0.2, geo=True, color='yellow', line_width=20, tiles='Cartolight', frame_height=300, frame_width=
map_plot = map_plot * clipped.hvplot(title=f'Vehicle trajectories {SEGMENT_ID} (n={len(clipped.trajectories)}), tiles=None)
map_plot + tt_plot
```

Out[50]:

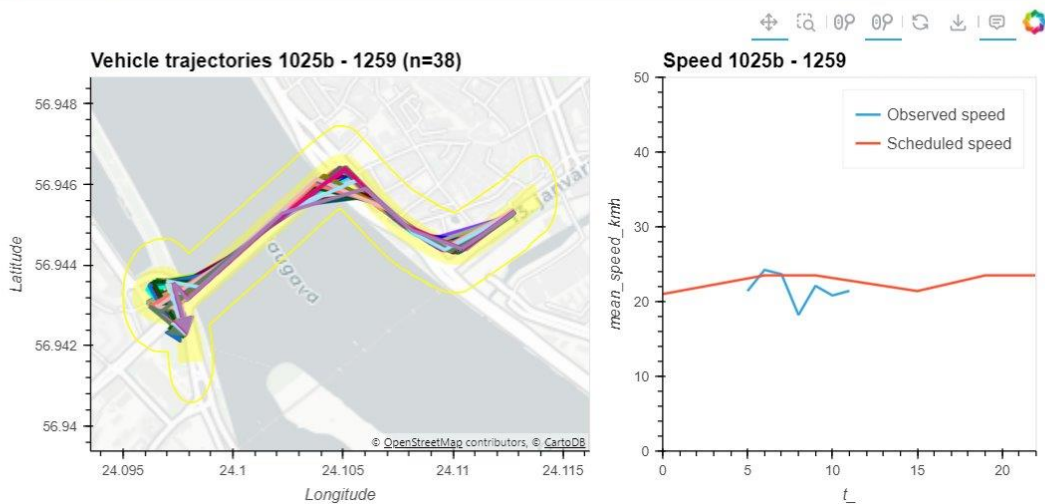


Figure 2-4: Trajectory data and travel time analysis results in a Jupyter notebook

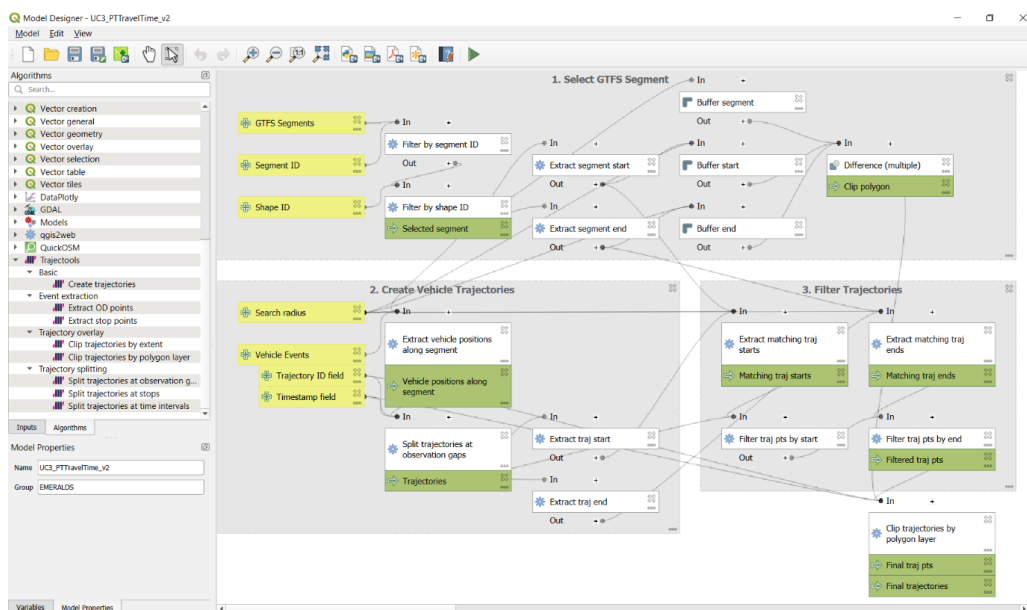


Figure 2-5: Screenshot of the analysis model under development for UC3 using Trajectools algorithms.

The following table summarizes the KPIs relevant to this emerald. They have been revised from the KPIs presented in D2.1 to better represent the improvement in analytical capabilities and to synchronize with the use cases in WP5. The first KPI has already been successfully achieved. The achieved value for the second KPI will be determined in the 1st assessment cycle, as detailed in the next steps section.

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|--|----------------|--------------|---|---|
| Interoperable analytics tool | 0 | >= 2 | Integration in multiple frameworks | 2 (MLOps platform & QGIS plugin) |
| Share of public transport network (segments) analysed | 0 | 20% | Application of the tool to UC3 PT network | Tbd in the 1 st assessment cycle |

Table 10: Trajectory Data Analysis KPIs

2.3.4 Next Steps

The conceptual work for no-code solutions for mobility data analysis developed in this emerald as well as the Trajectools plugin implemented in QGIS Processing can serve as a prototype for how to add mobility support in other GIS-based no-code environments, such as Carto’s workflow designer.

In the ongoing 1st integration and the 1st assessment cycle that will follow, we will apply the tool to analyse the UC3 PT network.

In the 2nd implementation cycle, will focus on the following aspects:

1. More integration work is ongoing to achieve feature parity between the no-code solution (Trajectools) and the underlying movement analysis library (MovingPandas) by adding trajectory cleaning, resampling, smoothing, and aggregation.
2. Simultaneously, feedback from use case partners who use Trajectools may lead to the development of new algorithms for MovingPandas.

2.4 Real-Time Extreme Scale Map Matching

This emerald implements a map matching offering that utilizes floating-car-data (FCD) to effectively estimate speeds on street network segments, thus turning the privacy-sensitive data of individual vehicles into anonymized traffic speed information.

Research interest in traffic state estimation using Floating Car Data (FCD) has surged, driven by the growing availability of vehicle trajectory data. This surge can be attributed to the proliferation of GPS-equipped vehicles and fleets across diverse sectors, such as taxi services and public transportation, rental companies. Through the process of map-matching, raw GPS data (Figure 2-6) from vehicle trajectories is linked to specific segments within the road network. This linkage facilitates the estimation of key traffic parameters such as speed, travel time, and congestion levels along designated road segments.

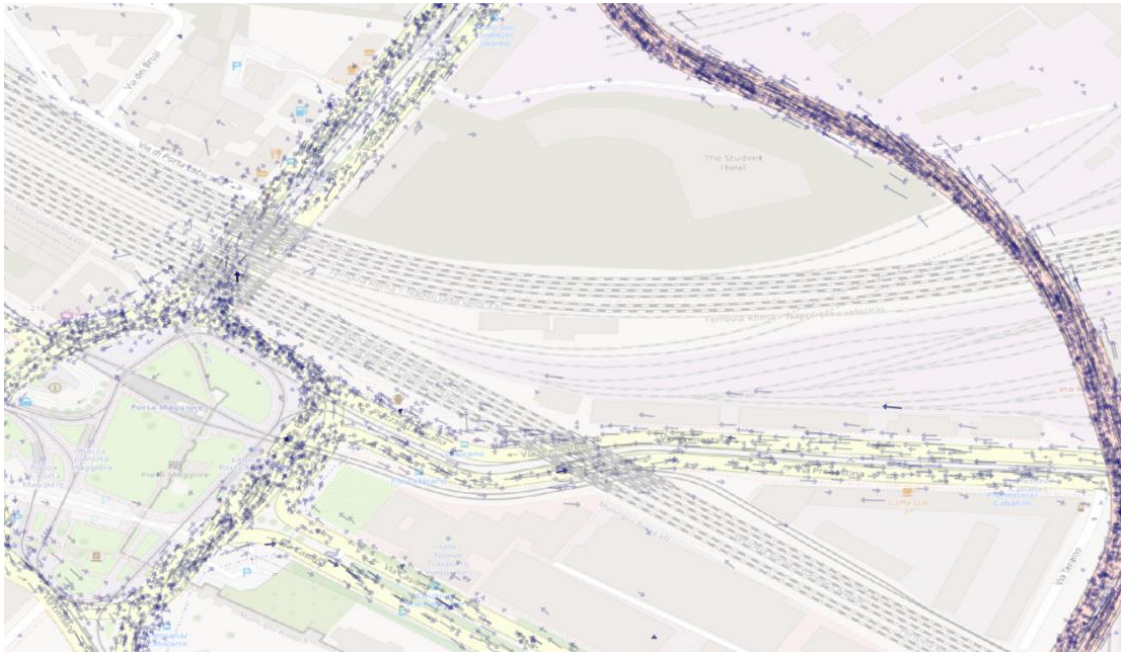


Figure 2-6: Raw FCD data

2.4.1 Brief Survey of the State-of-the-Art

Many academic studies focused on the topic of map matching, including Zhao et al. [36] and Altintasi et al. [37]. The most popular SotA approaches build on the concept of Hidden-Markov-Models, as discussed in detail in the SotA review of the WP3 “Extreme Scale Map Matching” emerald presented in D3.1.

This emerald distinguishes itself from the WP3 “Extreme Scale Map Matching” emerald by its focus on effective real-time and extreme scale street network segment speed estimation rather than highly accurate path reconstruction for individual moving object trajectories.

The most notable open-source map-matching tools are:

1. **Valhalla:** A flexible routing engine with map matching capabilities
 - valhalla.github.io
2. **Open Source Routing Machine (OSRM):** High-performance routing engine for shortest paths in road networks
 - [Project OSRM on GitHub](https://github.com/Project-OSRM)
3. **GraphHopper:** Route planning library and server using OpenStreetMap data
 - [graphhopper/map-matching at master · graphhopper/graphhopper · GitHub](https://github.com/graphhopper/graphhopper)
4. **Barefoot:** Java library for real-time map matching with OpenStreetMap
 - [Barefoot GitHub](https://github.com/Barefoot)
5. **pgMapMatch:** Python library using PostgreSQL PostGIS extension
 - [GitHub - amillb/pgMapMatch: map-matching of GPS traces](https://github.com/amillb/pgMapMatch)

The limits of the above tools are mainly related to:

- Lack of capability to read generic street network data standards because they are based on OpenStreetMap.
- Performance and scalability when dealing with big data (e.g., high number of single GPS points coming from connected vehicles).

- The map: a set of files representing the map on which to work.
- FCD data: a set of files representing the raw FCD data to be read.

The module emits as output:

- Data exposed through an OpenAPI interface²⁴.

The executables are distributed as Docker images and require to be properly configured. A Kubernetes²⁵ installation on a single node is enough to run it properly. The first version of the tool does not support horizontal scalability, next versions will.

In the next versions, there will be the need for a stream processing system, the choice will probably fall on Kafka²⁶ compatible.

The system also needs to read local data, so some sort of persistent volume is needed. E.g., using an NFS server along with Kubernetes managed mounts could be a good solution.

Summing up, for the current version of the tool there is the need for:

- A Kubernetes installation, that could be contained in a single node.
- The capability to reach a Docker registry where PTV can put its Docker images.
- A Kubernetes Ingress service, e.g., a NGINX server installed on the k8s cluster.
- A persistent storage area.

The module requires inputs as static files. This must be made available to the running pod, either as a locally mounted folder, or as a shared folder. In either case the system will be able to use them.

- The map: the map files will be provided as a collection of files. The folders must have the following structure:
 - *got*, containing a single folder
 - *emeralds*, containing a sequence of folders like:
 - *<xyz>*, i.e., a 3-letter named folder containing a single file like:
 - *<filename>.pbf.gz*
- FCD data: the FCD data to be used to feed the tool must be provided as a collection of compressed CSV files in a folder called FCD.

The file must be a CSV, the fields must be separated with “,”. Most fields are reserved for future versions usage, depending on the quality of the raw data that will be available.

The current mandatory fields are:

- Latitude: the latitude, expressed in decimal format with the ‘.’ as decimal separator. Minimum 5 digits after the decimal point
- Longitude: the longitude, expressed in decimal format with the ‘.’ as decimal separator. Minimum 5 digits after the decimal point

²⁴ <https://www.openapis.org/>

²⁵ <https://kubernetes.io/>

²⁶ <https://kafka.apache.org/>

- Time: the timestamp expressed as milliseconds since unix epoch (accuracy better than a second)
- Provider_trace_id: a string unique to the trip of a vehicle. Preferably a fixed length base64 encoded hash

The output is provided by the module as an OpenAPI compatible output. The specifications are provided in the OpenAPI format by properly accessing the URL of the service whenever installed.

The module generally makes use of the OpenLR standard to refer to streets. Details on this methodology can be found on the website of the OpenLR association²⁷, more than all from the whitepaper²⁸.

The Kubernetes Ingress must be properly configured to be able to reach the APIs.

2.4.3 Preliminary Evaluation

The current developments and testing activities rely on a benchmark network representing a portion of the Rome city centre with 5,612 nodes and 9,559 links, and a GPS dataset of 0.5 million datapoints for a single day.



Figure 2-9: Benchmark network of the city of Rome

- The **data processing latency** is given by the processing time plus a time window that there might be to allow for FCD data coming in not ordered in time, plus a polling interval (<1 min). At the current stage, this cannot be tested due to the use of pre-processed dataset only. The design goal for the data processing latency is to be less than 1 minute.
- At this stage, the **memory usage** has been of ~12GB (Map ~100MB, every node; trajectories in progress ~10GB, total partitioned over all nodes; Estimator 0.5GB, every node).

²⁷ <https://www.openlr-association.com/>

²⁸ https://www.openlr-association.com/fileadmin/user_upload/openlr-whitepaper_v1.5.pdf

- The **ingestion rate** will be tested once the real time stream for the York showcase and the associated infrastructure are available.

The following table summarizes the KPIs relevant to this emerald, as previously described in D2.1. The baseline values will be established using the open source tools listed in the SotA section above.

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|--------------------------------|----------------|--------------------------------|--|---|
| Data Processing Latency | SotA | < 1 minute | Processing time + time window to allow FCD data coming in not ordered in time + polling interval | Tbd in the 2 nd assessment cycle |
| Memory Usage | SotA | 64GB / 16 CPU | Utilization of the Random Access Memory | 12GB / 1 CPU ²⁹ |
| Ingestion rate | SotA | 2k datapoints /s ³⁰ | | Tbd in the 2 nd assessment cycle |

Table 11: Extreme Scale Map Matching KPIs

2.4.4 Next Steps

In the 2nd implementation cycle, we will focus on the following aspects:

1. To be able to ingest data from more than one publicly or commercially available FCD data provider, means of communication have to be properly understood. Some providers could send data in batches, some other in a streaming fashion, even more suited to this work's needs.
2. To extend the emerald to scale horizontally and make it "extreme-scale", so that even if the amount and ratio of ingested data increase, or the size of the study area increase, the application can still work with the expected latency by installing more copies of it, working in parallel.

²⁹ benchmark network of 5.612 nodes and 9.559 links, and a GPS dataset of 0.5 M datapoints for a single day

³⁰ for the showcase and able to scale up to 1M datapoints/s

3 Active & Federated Learning over Mobility Data

This chapter presents the developed emeralds that employ state-of-the-art timeseries modelling techniques for traffic flow, parking occupancy and visitor data to provide a final crowd prediction system. Furthermore, XAI and active learning techniques are leveraged to enhance the performance and transparency of the model predictions to ensure better decision making in urban management systems. The emeralds presented in this deliverable include three ML models and one emerald developing XAI:

1. **Traffic State / Flow Forecasting:** This emerald utilizes historic traffic flow data from various sensors to forecast traffic state and flow patterns in different regions. This kind of predictive model is essential to predict traffic conditions and thus enables the development of efficient traffic management strategies.
2. **Parking Garage Occupancy Prediction:** This emerald utilizes historic parking occupancy timeseries, weather, and event information to predict the future availability of parking spaces in specific locations. This kind of predictive model is key to developing effective parking management systems and can provide input to downstream models, such as crowd density prediction.
3. **Crowd Density Prediction:** This emerald utilizes crowd data and auxiliary data sources to forecast the crowdedness in specific areas. This kind of predictive model is essential to accurately predict crowd patterns and thus enables the development of efficient crowd management systems.
4. **Active Learning & XAI for Crowd Prediction:** This emerald leverages explainable AI (XAI) and active learning (AL) techniques to create a transparent and interpretable ML pipeline. This empowers data experts to provide insights and improvements to the predictive models, facilitating better decision-making in urban planning and management systems.

The source code or object code of these emeralds can be found in the following repositories:

| Emeralds | Repositories | Publications |
|--|--|--------------|
| Traffic State / Flow Forecasting | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/WP4_Traffic_state_forecasting | |
| Parking Garage Occupancy Prediction | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/analytics-and-learning/tree/main/uc1-parking-model | |
| Crowd Density Prediction | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/analytics-and-learning/tree/main/uc1-crowd-model | |
| Active Learning & XAI for Crowd Prediction | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/analytics-and-learning/tree/main/uc1-crowd-model • https://github.com/emeralds-horizon/analytics-and-learning/tree/main/uc1-parking-model | 31 |

³¹ Jalali, Graser & Heistrachre (2023) Towards eXplainable AI for Mobility Data Science. <https://arxiv.org/abs/2307.08461>

3.1 Traffic State / Flow Forecasting

This emerald utilizes historic traffic flow data from various sensors to forecast traffic state and flow patterns in different regions. The goal of this emerald is to accurately predict traffic conditions and enable efficient traffic management strategies.

The traffic forecasting problem represents a spatiotemporal timeseries prediction problem, where the input comprises a set of traffic conditions (e.g., flow, average speed etc.) represented in one or more timeseries, and the output is a forecast of future conditions.

Problem studied: given a network of N detectors that observe C traffic conditions at T historical time steps, denoted as $X = \{X_t \in R^{N \times C} | t = t_0 - T + 1, \dots, t_0\} \in R^{T \times N \times C}$, where t_0 is the current timestep, we aim to predict the traffic conditions for all detectors in the next Q time steps, $\hat{Y} = \{X_t \in R^{N \times C} | t = t_0 + 1, \dots, Q\} \in R^{Q \times N \times C}$.

The purpose of this module is to provide accurate short-term forecasts of future traffic state conditions, based on historical observations, essential for the application of smart cities—particularly in traffic management and urban planning. In the upcoming section, we address the problem of forecasting average traffic speed without leveraging traffic flow information. This section provides:

- A brief overview of state-of-the-art models for traffic state forecasting
- Introduction of the implemented forecasting model
- First results on models' benchmarking and evaluation for various forecasting horizons (i.e., 15, 30, 45, 60 minutes)

3.1.1 Brief Survey of the State-of-the-Art

Traffic forecasting models are generally classified into two main categories: univariate models, which utilize a single timeseries from an individual sensor, or multivariate models, which inherently assumes interdependencies among timeseries from multiple sensors across the network [38]. Today, the prevailing trend involves employing multivariate models to capture complex spatiotemporal relationships between traffic data. Data-driven traffic forecasting methodologies can be broadly categorized into three groups: (1) statistical methods, (2) traditional machine learning, and (3) deep learning.

Statistical methods, including history average (HA), vector auto-regression (VAR), and auto-regressive integrated moving average (ARIMA), are especially suitable for smaller datasets, benefiting from a clear and simplified computational structure compared to more advanced machine learning approaches. However, these methods often require satisfying the stationarity assumption for each timeseries. Compared to statistical methods, **Machine Learning (ML)** models can offer more robust generalization capabilities, learn complex non-linear correlations effectively, and process high-dimensional data. ML approaches can be segmented into three categories: feature-based methods, Gaussian process models, and state-space models [39]. **Deep Learning (DL)** models have become increasingly popular in recent years, and research has demonstrated their strong applicability for timeseries forecasting. In the past decade, the rapid development of architectures has led to the emergence of hybrid networks based on Convolutional Neural Networks (CNN) [40], and Recurrent Neural Networks (RNN) [41]. For timeseries applications, LSTM is preferred over conventional RNNs due to its ability to selectively remember long-range dependencies, addressing the vanishing gradient problem. Hybrid networks like ConvLSTM [42] and PredRNN [43] have been increasingly applied to predictive learning of urban spatiotemporal data and have shown significant advantages.

To address the limitation of conventional methods in learning directly from non-Euclidean data within urban systems, recent breakthroughs have emerged through deep learning techniques, notably **Graph Neural Networks (GNN)** [44]. Expanding upon this, Kipf and Welling [45] introduced the Graph

convolution network (GCN) as an evolution of the CNN model, designed to perform convolution on structured graphical input. Additionally, to jointly capture the spatiotemporal features without using an RNN cell, the seminal work of Yu, Yin and Zhu [46] proposed the Spatiotemporal Graph Convolutional Network (STGCN).

Following the introduction of the Transformer [47,48], **attention-based** methods in traffic forecasting were designed to capture long-term dependencies. Wu et al. [49] proposed integrating an attention mechanism within a LSTM architecture for improving performance for short-term traffic speed forecasting. In an alternate approach, Guo et al. [50] proposed an Attention-based Spatial-Temporal Graph Convolutional Network (ASTGCN), employing both graph-based and standard convolution with attention mechanisms for capturing the spatiotemporal traffic trends.

To improve the reliability of long-term multivariate forecasting, Zheng et al. [51] proposed the Graph Multi-Attention Network (GMAN). GMAN integrates attention mechanisms into an encoder-decoder architecture, using multiple spatiotemporal attention blocks. A key innovation of GMAN is the transform attention mechanism, which is an attention layer between the encoder and decoder. Other approaches address the highly dynamic spatial correlations by introducing adaptive graph learning methods. DGCRN Li et al. [52] proposed a hyper-network to adaptively generate a dynamic adjacency matrix. To better capture temporal correlations, D2STGNN (Decoupled Dynamic Spatial-Temporal Graph Neural Network) [53] incorporates a temporal residual decomposition method along with a dynamic graph learning module based on a self-attention mechanism.

3.1.2 Overview and Description

In this section we provide a brief overview of the implemented models.

In Graph Multi-Attention Network (GMAN) approaches, the road network is interpreted as a weighted directed graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. \mathcal{V} is a set of $N = |\mathcal{V}|$ vertices representing the traffic sensors on the road network; \mathcal{E} is a set of edges representing the connectivity among vertices; and $\mathcal{A} \in \mathbb{R}^{N \times N}$ is the weighted adjacency matrix, where \mathcal{A}_{v_i, v_j} represents the proximity (measured by the road network distance) between vertex v_i and v_j .

GMAN adapts an encoder-decoder architecture, where both the encoder and the decoder consist of multiple spatiotemporal attention blocks. The encoder encodes the input traffic features and the decoder predicts the output sequence. Between the encoder and the decoder, a transform attention layer is applied to convert the encoded traffic features to generate the sequence representations of future time steps as the input of the decoder. The transform attention mechanism models the direct relationships between historical and future time steps that helps to alleviate the error propagation problem among prediction time steps.

Moreover, GMAN stacks multiple spatiotemporal attention blocks within the encoder and decoder, forming a deep network capable of extracting the traffic trends in large sensor networks. Within each spatiotemporal block, the spatial attention mechanism dynamically updates the pair-wise correlation between vertices with respect to the current time step, while the temporal attention mechanism dynamically weights the impact of previous and future time steps on the current step. Notably, the input data must be first transformed via a spatiotemporal embedding process to incorporate the time-dependent graph data into the attention mechanisms. Figure 3-1 illustrates the architecture of GMAN.

For benchmarking purposes, we employ a straightforward statistical model for univariate traffic forecasting, excluding cross-dependencies among the network's sensors. Various approaches address timeseries with multiple seasonal patterns, such as seasonality decomposition, the Trigonometric, Box-Cox, ARMA, Trend, Seasonal (TBATS) model, Meta's Prophet model, or the incorporation of Fourier terms as exogenous variables [54].

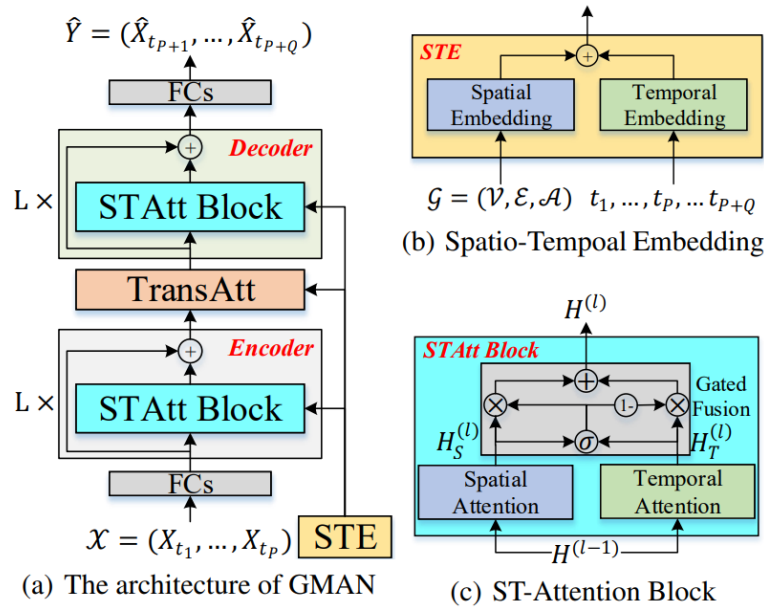


Figure 3-1: The framework of Graph multi-attention network

We opt for decomposing our series into trend, multiple seasonal components, and residuals using the Seasonal-Trend decomposition using Loess (MSTL) approach [55]. In this approach, each timeseries is decomposed into the following components:

$$X_t = \sum_k^n \widehat{S}_t^k + \widehat{T}_t + \widehat{R}_t$$

Here, X_t represents the observation at time t , and $\widehat{S}_t^k, \widehat{T}_t, \widehat{R}_t$ refer to the k^{th} seasonal component, trend, and residual, respectively. n denotes the number of distinct seasonal components.

Following this decomposition, we implement a simple ARIMA (1,0,1) model to the seasonally adjusted timeseries.

Regarding the codebase, all the code that is used in this emerald is hosted at our project's GitHub repository.

3.1.3 Preliminary Evaluation

We conduct experiments on two real-world large-scale datasets:

- **PEMS-BAY**: A public average traffic speed dataset, collected by California Transportation Agencies (CalTrans), comprises data from 325 sensors in the Bay Area ranging from January 1, 2017, to June 30, 2017, with measurements recorded every 5 minutes. Speed is measured in miles per hour and sensor locations, along with other metadata, are recorded in a separate metadata file. The road network's adjacency matrix is also provided.
- **NDW (UC2)**: A private average traffic speed/flow dataset, collected by National Road Traffic Data Portal, comprises data from 206 sensors in the Rotterdam area, Netherlands, ranging from April 25, 2022, to May 7, 2023, with measurements recorded every minute. Speed is measured in kilometres per hour, and traffic flow corresponds to the number of recorded cars per sensor. Sensor locations, along with other metadata, are documented in a separate metadata file. The road network's adjacency matrix is also provided.

To reduce computational load, we resample the NDW dataset to 5-minute intervals and retain a subset of the original data. Detailed statistics for the two datasets are presented in Table 12, while the distributions of sensors are visualized in Figure 3-2.

| Datasets | Samples | Sensors | Dates | Sample rate | Flow |
|----------|---------|---------|---------------------|-------------|------|
| PeMS-BAY | 52116 | 325 | 1/1/2017-30/6/2017 | 5 Min | No |
| NDW | 51841 | 206 | 1/11/2022-30/4/2023 | 5 Min | No |

Table 12: Statistics of PeMS-BAY and NDW datasets



Figure 3-2: Sensor distribution in NDW (left) and PeMS-BAY (right) datasets.

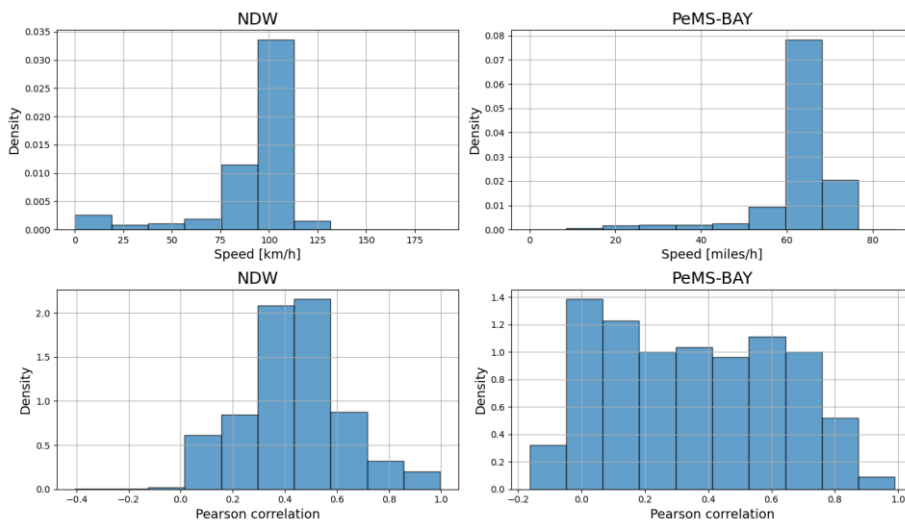


Figure 3-3: Distribution of speed and inter-node correlations on PeMS-BAY and NDW datasets.

Statistical analysis of the two datasets reveals distinct characteristics, as depicted in Figure 3-3. By calculating the Pearson correlation coefficients between all nodes, we can observe the explicit spatial correlations in NDW while those in PEMS-BAY are distinctly weaker. The velocity distribution of the PEMS-BAY dataset exhibits greater monotony, and it is more intensively close to the free-flow velocity, which means simpler traffic conditions and lower correlations between nodes. Velocity distributions in NDW tend to be polarized with more missing values marked as zero. Consequently, it is natural for methods to achieve significant performance on PEMS-BAY dataset. For the next steps, we apply linear interpolation techniques to estimate all missing values within timeseries.

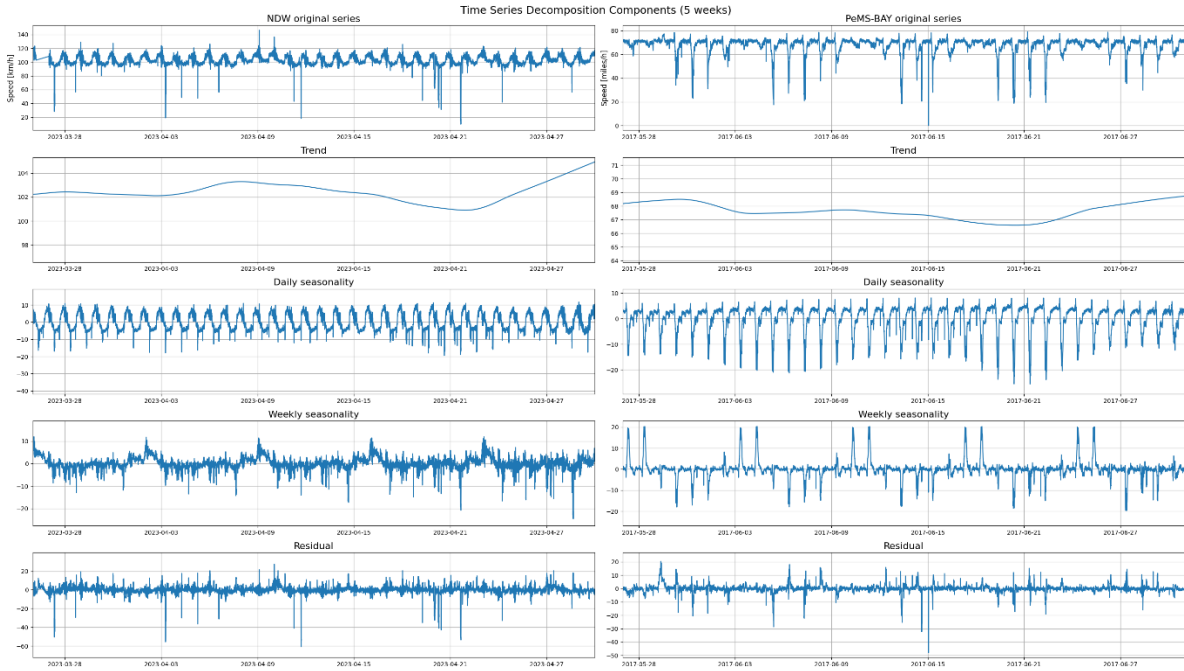


Figure 3-4: Decomposition of random sensor's measurements for both datasets.

The timeseries decomposition results for a random sensor, as depicted in Figure 3-4, suggest that the NDW dataset exhibits weaker weekly behaviour compared to the PEMS-BAY dataset. However, both datasets display a relatively stable trend component and robust intra-day peak components.

The applicability of the traffic state forecasting module is assessed based on performance and the computational cost for each model. As part of this process, we keep 70% of our datasets for training our models, 20% for validation and 10% for testing purposes. The experiments are run on an AMD EPYC 7742 processor (3.4 GHz), an NVIDIA A100 GPU and an Intel i7 NUC.

For GMAN, we use the default settings from their original proposal. The performances of all methods are evaluated by three commonly used metrics in traffic prediction, namely (i) Mean Absolute Error (MAE), which is a basic metric to reflect the actual situation of the prediction accuracy, (ii) Root Mean Squared Error (RMSE), which is more sensitive to abnormal value, and (iii) Mean Absolute Percentage Error (MAPE), which can eliminate the influence of data unit to some extent, defined as follows:

$$MAE(y, \hat{y}) = \frac{1}{M} \sum_{i=0}^M |y_i - \hat{y}_i|$$

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{M} \sum_{i=0}^M (y_i - \hat{y}_i)^2}$$

$$MAPE(y, \hat{y}) = \frac{1}{M} \sum_{i=0}^M \frac{|y_i - \hat{y}_i|}{y_i}$$

where y denotes the ground truth, \hat{y} denotes the predicted values and M represents the number of observed samples.

Table 13 shows the comparison of different methods for 15 minutes (3 steps), 30 minutes (6 steps), 45 minutes (9 steps), and 1 hour (12 steps) ahead predictions on two datasets.

| | | 15 min | | | 30 min | | | 45 min | | | 60 min | | |
|----------|------------|--------|------|-------|--------|-------|-------|--------|-------|-------|--------|-------|-------|
| | | MAE | RMSE | MAPE | MAE | RMSE | MAPE | MAE | RMSE | MAPE | MAE | RMSE | MAPE |
| PeMS-BAY | MSTL-ARIMA | 2.78 | 4.80 | 0.061 | 3.07 | 5.40 | 0.067 | 3.24 | 5.75 | 0.072 | 3.37 | 6.00 | 0.075 |
| | GMAN | 1.35 | 2.94 | 0.029 | 1.64 | 3.81 | 0.038 | 1.78 | 4.20 | 0.042 | 1.87 | 4.40 | 0.044 |
| NDW | MSTL-ARIMA | 5.51 | 9.37 | 0.084 | 6.19 | 10.70 | 0.096 | 6.51 | 11.33 | 0.101 | 6.56 | 11.37 | 0.100 |
| | GMAN | 3.60 | 8.05 | 0.059 | 4.05 | 9.22 | 0.068 | 4.39 | 10.00 | 0.075 | 4.65 | 10.55 | 0.080 |

Table 13: Performance comparison of different approaches for traffic prediction on PEMS-BAY and NDW datasets

Table 14 shows the comparison of the training/inference time for each model on the PEMS-BAY dataset. The GMAN training required 35 epochs, due to the early-stop mechanism.

| Model | Training time [min] | Inference time [min] | Device |
|------------|---------------------|----------------------|--------|
| MSTL-ARIMA | 7 | 6.5 | CPU |
| GMAN | 10/epoch | 0.5 | GPU |

Table 14: Computational summary on the PEMS-BAY dataset

Despite its lower performance, the MSTL-ARIMA method demands notably fewer resources, including energy, processing capacity, and storage, rendering it better suited for decentralized approaches, where GPU enabled systems are not available or suitable. Conversely, the GMAN approach attains superior accuracy and offers better scalability for larger datasets. However, it relies on a centralized infrastructure with sufficient computing resources (GPU).

To address extreme scale data, centralized approaches appear more prominent, given their ability to parallelize computations on GPUs. Due to the rapid data sampling rate (every 5 minutes), CPU-based applications struggle to scale without preprocessing techniques, such as resampling. The latency of this module comprises both the inference time and the data ingestion time. The design objective for forecasting latency is to remain below the time required for updating our forecasts (e.g., 5 minutes or 10 minutes). The above conclusions underscore the absence of a one-size-fits-all solution that excels in every aspect. Selecting the appropriate method entails aligning it with our performance requirements and hardware capacity. It is also evident that focus needs to be put on evaluating and expanding on methods that can be comparable with respect to accuracy, instead of focusing on just reducing energy and computational needs.

The following table summarizes the KPIs relevant to this emerald, expanding the prediction accuracy metrics described in D2.1 with additional performance metrics.

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|-----------------------------------|----------------|---|---------------------------------------|--------------------|
| Prediction accuracy / Performance | SotA | 20% speed up with the same or better accuracy | Error metrics / Fit & inference times | No improvement yet |

Table 15: Traffic State / Flow Forecasting KPIs

3.1.4 Next Steps

In the ongoing 1st integration cycle, our primary objective is to integrate the functionality of the methods presented in this section with other EMERALDS components and pipelines. These



integrations can leverage the results for tasks such as traffic state estimation to fill in missing values, thereby contributing to the development of an adaptive and resilient cumulative platform.

In the 2nd implementation and integration cycles, we will focus on further development of our models, involving the inclusion of additional traffic information, such as traffic flow, in the forecasting module to enhance forecasting performance. Efforts can also be directed towards reducing the computing resource demands of the implemented models. Additionally, we aim to explore more methods that may better align with urban traffic behaviour and the specific forecasting task. For instance, certain deep learning methods may excel in shorter forecasting horizons.

3.2 Parking Garage Occupancy Prediction

This emerald utilizes historic parking occupancy timeseries, weather, and event information to predict the future availability of parking spaces in specific garages. The goal is to precisely forecast high-occupancy parking demand patterns and thus provide input to increase the effectiveness of parking management systems.

The parking garage occupancy prediction problem presents a spatiotemporal timeseries prediction problem, where the input comprises of a set of past occupancy observations and environmental factors (e.g., weather and events) represented in multiple timeseries, and the output is a prediction of future occupancy values.

Problem studied: given a set of N parking garages that report C occupancy conditions at T historical time steps, denoted as $X = \{X_t \in R^{N \times C} | t = t_0 - T + 1, \dots, t_0\} \in R^{T \times N \times C}$, where t_0 is the current timestep, we aim to predict the occupancy conditions for all parking garages in the next Q time steps, $\hat{Y} = \{X_t \in R^{N \times C} | t = t_0 + 1, \dots, Q\} \in R^{Q \times N \times C}$.

These predictions can be modelled as either regression or classification tasks. As part of the data science process iterations, we explored diverse modelling approaches. Based on use case domain expert feedback, the correct prediction of high-occupancy events has higher priority than the prediction of lower-occupancy situations. We have therefore implemented both regression and classification models. This section provides:

- A brief overview of state-of-the-art models for parking garage occupancy prediction
- Introduction of the implemented prediction models
- First results on models' benchmarking and evaluation

Furthermore, these models serve as a basis for further advanced work as part of the T4.2 emerald "Active Learning & XAI for Crowd Prediction".

3.2.1 Brief Survey of the State-of-the-Art

Parking occupancy prediction has emerged as a critical aspect of urban mobility management and planning, driven by the escalating challenges posed by growing motor vehicle numbers and limited parking resources. Recent research has focused on developing advanced predictive models to optimize parking resource utilization, enhance traffic conditions, and facilitate efficient coordination of multiple parking facilities at various scales. Several key trends and methodologies are evident from the reviewed literature. Firstly, the adoption of hybrid models combining different recurrent neural network (RNN) [56] architectures, such as gated recurrent unit (GRU) [57] and long short-term memory (LSTM) [58], has gained traction. These hybrid models leverage the strengths of both architectures to improve prediction accuracy and efficiency while considering multiple factors like occupancy, weather conditions, and holidays. Secondly, studies have explored the performance of various forecasting methods across different types and scales of parking lots. Support vector machine (SVM) [59] has emerged as a stable and accurate predictor for diverse parking scenarios, particularly

for commercial, mixed functional, and large-scale parking lots. Thirdly, there is a growing emphasis on spatial-temporal prediction models that consider the similarities between parking lots in terms of functionality and occupancy patterns. These models integrate graph convolution networks with LSTM and temporal pattern attention mechanisms to assign appropriate weights to spatial features, resulting in significant improvements in prediction accuracy [60].

Furthermore, researchers have proposed innovative approaches that integrate timeseries decomposition, meta-learning, and deep learning techniques to enhance feature engineering, model building, and generalizability [61]. These approaches have demonstrated superior performance in terms of prediction accuracy, model adaptation speed, and robustness across different forecasting intervals and parking lot types. Moreover, the convergence of the Internet of Things (IoT) and Artificial Intelligence (AI) has opened new avenues for predicting parking availability by leveraging sensory data from urban environments [62]. Machine learning models based on neural networks and random forests [63] have shown remarkable performance in predicting parking occupancy, outperforming traditional forecasting methods. Finally, the impact of external factors, such as anti-pandemic policies [63] on parking behaviour and occupancy prediction [64], has been investigated. Novel policy-aware temporal convolutional network (P-TCN) [63] models have been developed to accurately identify the effects of policy interventions and improve prediction reliability in dynamic urban environments.

3.2.2 Overview and Description

In this section we provide a brief overview of the implemented models. The following models were used:

- Forecasting / regression: naïve seasonal baselines, XGBoost, Prophet, Ridge regression, Decision Tree, and univariate ARIMA
- Classification: AdaBoost, GaussianProcess, RandomForest, K-nearest Neighbor with K=3, and a Fully Connected Dense Neural Network (DNN) classifier

The focus of the modelling stage is on determining the optimal combination of variables, including historical occupancy data, weather conditions, events, and other relevant factors, to refine the predictive capability of the model. All models are trained using historical parking garage occupancy timeseries, as well as event and weather timeseries (including hourly and daily observations of temperature, precipitation rate, precipitation chance, wind speed and direction, and cloud cover).

To address the geographic overflow-effect between parking garages (i.e., users first try to park in the most conveniently located garage and only once that one is filled, they search for spots in other garages) we enhanced the model by encoding the occupancy of adjacent parking garages as features.

The experiments are implemented using Python and the Darts library, a specialized timeseries forecasting library. Figure 3-5 illustrates the workings of the Darts historical forecast function, which is used to evaluate the performance of the prediction models which are continuously re-trained to simulate how models would continue learning over time.

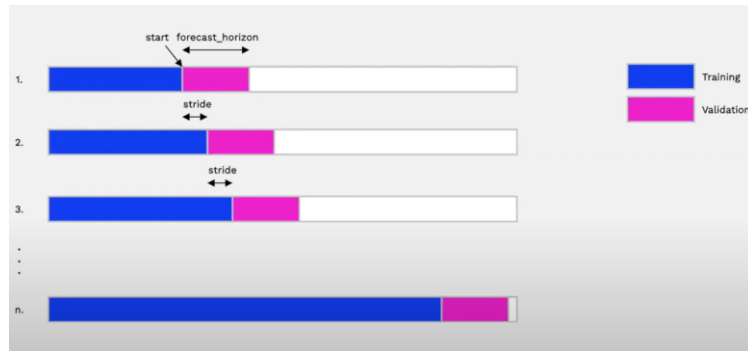


Figure 3-5: Visualization of the historical forecast and how the model re-training functions³²

3.2.3 Preliminary Evaluation

We conduct experiments on real-world data from use case 1 (UC1) comprising parking data, daily and hourly weather data, and calendar data. Initially, the parking data consisted of approximately 99,000 observations. In data cleansing step (based on feedback from UC1 data experts), we refined the dataset to 98,737 observations spanning from May 2022 to the end of October 2022. The weather dataset includes hourly and daily observations of temperature, precipitation rate, precipitation chance, wind speed and direction, and cloud cover. Additionally, the calendar dataset provides information on national holidays in the Netherlands.

To facilitate analysis, we concatenated all three datasets for the period from May 2022 to the end of October 2022. Subsequently, we resampled the data based on hourly maximum values. We then calculated the occupancy percentage for each parking facility and performed min-max normalization on each parameter to prepare the data for modelling purposes (see Figure 3-6).

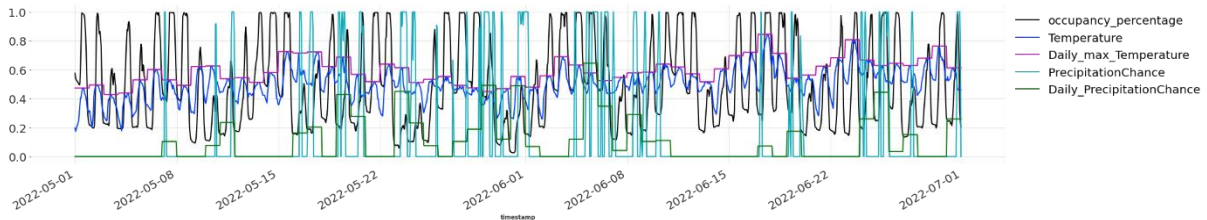


Figure 3-6. Visualization of the pre-processed input data for parking occupancy prediction models

Examining the distribution of the target variable, we observed characteristics of both bi-modality and skewness (see Figure 3-7). Furthermore, following min-max data normalization, a significant proportion of values cluster closely around zero, while occasional anomalous occurrences, such as rare instances of high occupancy rates, are evident in the data. To ensure robust assessment of model performance under these conditions, we opted for evaluation metrics such as R^2 , RMSE, precision, and recall.

³² Image source: „Darts for Timeseries Forecasting“ by Julien Herzen & Francesco Lässig
<https://www.youtube.com/watch?v=Kf6b5falvOM>

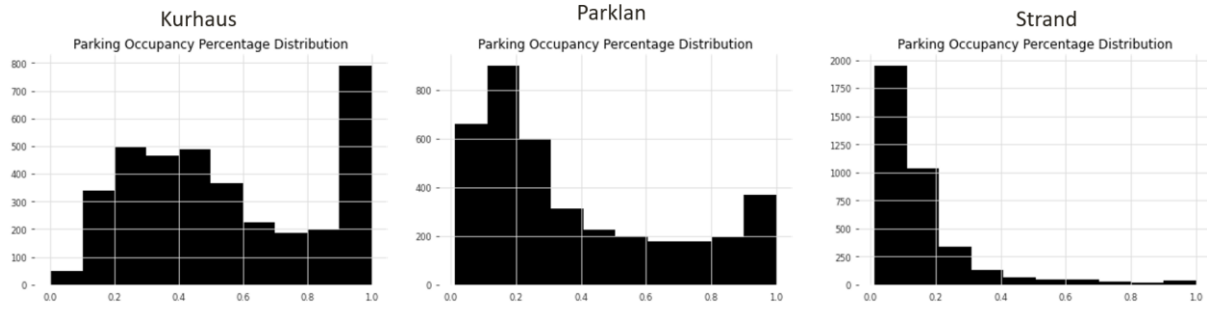


Figure 3-7: Illustration of the target variable distribution for all three parking garages.

The initial models are trained on May to July 20th, 2022. The remaining time period until the end of October 2022 is used for evaluation (using Darts’ historical forecast, as shown in Figure 3-5). Employing the historical forecast function, we predict a 7-day horizon and subsequently retrained the model for continuous weekly predictions.

To evaluate our models' performance, we established two naïve seasonal baselines:

- Baseline 1 predicts that the current week's occupancy will mirror that of the previous week (Figure 3-8).
- Baseline 2 assumes that the average occupancy from the prior month will be replicated in the current month (Figure 3-9).

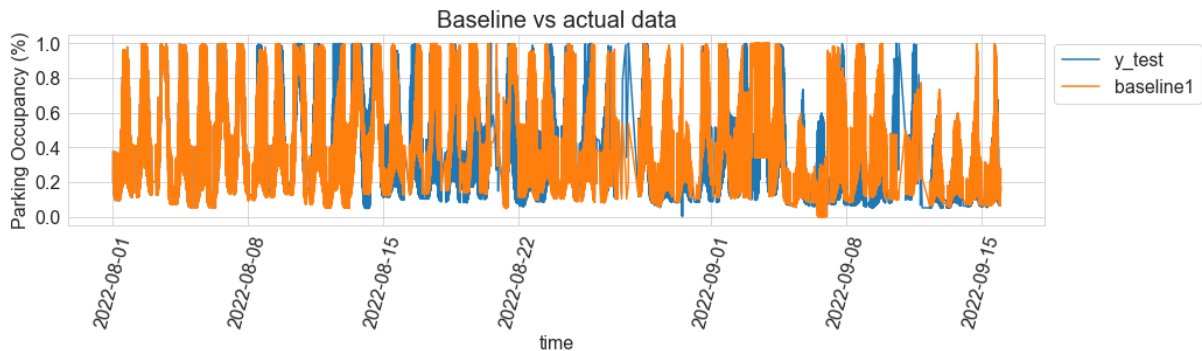


Figure 3-8: Naive baseline 1 for parking garage occupancy prediction.

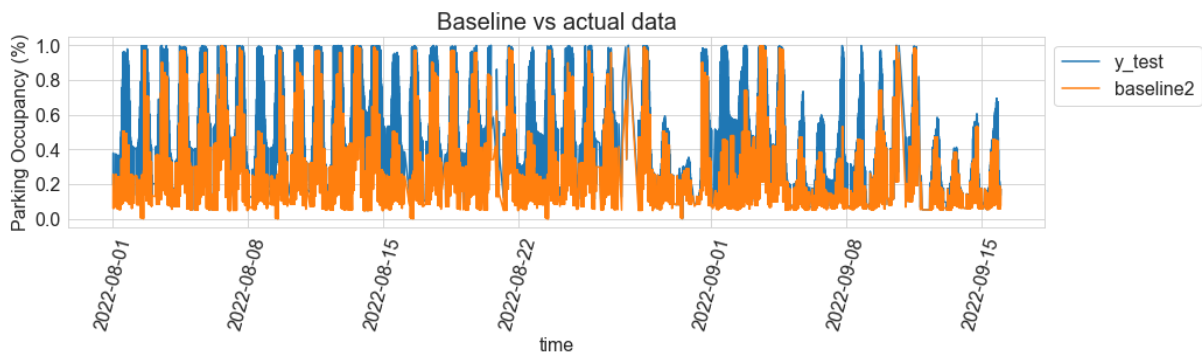


Figure 3-9: Naive Baseline 2 for parking garage occupancy prediction.

Results of these baselines are included in Table 16 which lists the RMSE, R^2 , precision, and recall scores for the test dataset, covering the period from mid-July 2022 to the end of October 2022. Critical high-occupancy events (used to determine precision and recall) are defined as hours with $\geq 90\%$ occupancy. Since high-occupancy events are of particular priority, the recall values are the most important metrics for comparing models.

3.2.3.1 Forecast / Regression models

Based on our preliminary analysis on the average prediction results, the XGBoost model exhibits promising performance in comparison to other regressors, including Facebook Prophet, Ridge regression, Decision Tree, and univariate ARIMA. The results of our best-performing models are summarized in Table 16. To account for instances where the parking garage reached full capacity, we incorporated penalties in the evaluation (by increasing the prediction error by 0.2 for cases where the model failed to correctly predict over 90% occupancy). In this table, we only consider the best performing penalized models.

The average XGBoost recall is 73.52% while the average baselines 1 and 2 recall are 79.93% and 55.49%, respectively. These results underline that naïve baselines (particularly baseline 1) are hard to beat through ML for these kinds of prediction tasks, particularly if the available training data is as limited as in UC1. Note that Prophet outperforms XGBoost on R^2 and RMSE but fails in precision and recall since it performs worse at detecting high-occupancy events.

| Parking | Model | R2 | RMSE | Precision | Recall |
|----------|------------|---------------|---------------|---------------|---------------|
| Kurhaus | Baseline 1 | 0.3258 | 0.2150 | 75.63% | 77.66% |
| | Baseline 2 | 0.0633 | 0.2534 | 89.81% | 64.02% |
| | XGBoost | 0.1878 | 0.2530 | 78.28% | 75.58% |
| | Prophet | 0.5653 | 0.1925 | 89.50% | 53.60% |
| Parklaan | Baseline 1 | 0.6108 | 0.1718 | 70.56% | 71.21% |
| | Baseline 2 | 0.4041 | 0.2126 | 95.14% | 52.46% |
| | XGBoost | 0.7981 | 0.1212 | 76.17% | 79.30% |
| | Prophet | 0.6625 | 0.1569 | 89.50% | 53.60% |
| Strand | Baseline 1 | 0.2008 | 0.1619 | 66.07% | 63.93% |
| | Baseline 2 | 0.0642 | 0.1822 | 49.18% | 50.00% |
| | XGBoost | 0.7058 | 0.0739 | 74.74% | 65.67% |
| | Prophet | 0.3694 | 0.1021 | 49.60% | 50.00% |

Table 16: Forecast model performance compared to the Naive Baselines on all three parking garages

We conducted further analysis on the model residuals for all parking garages, including the problematic weeks where the car park reached full capacity, but the model failed to predict higher occupancy. The XGBoost model for Parklaan has a good performance compared to the baselines, as shown in Figure 3-10. Few hours were incorrectly predicted as full (low-cost misprediction), and a few hours were incorrectly predicted as low-occupancy (high-cost mispredictions).

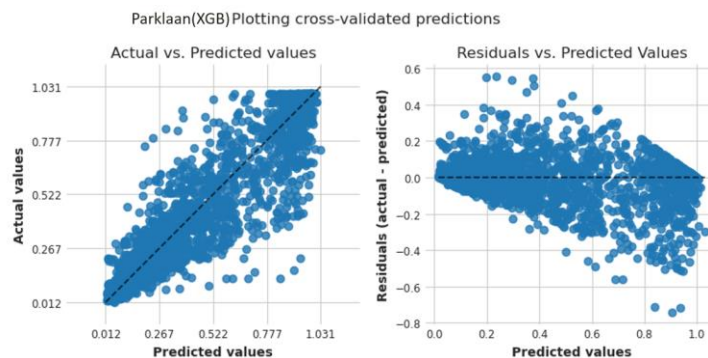


Figure 3-10: Residual Analysis of Parklaan XGBoost model.

The XGBoost model for Strand mistakenly predicts lower occupancy when the parking is indeed at its high occupancy rate, as shown in Figure 3-14. This is because this parking garage has mostly a low occupancy percentage since it has a large capacity but very few visitors and it only receives visitors when the other two adjacent garages are already at their full occupancy.

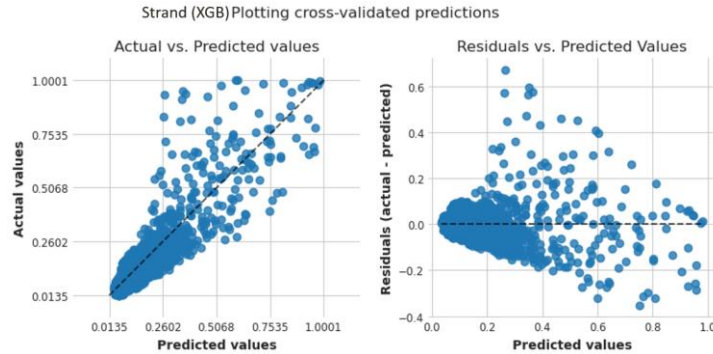


Figure 3-11: Residual Analysis of Strand XGBoost model.

We further inspected the problematic weeks, where the model could not correctly predict the occupancy rate of the car park. This is depicted in the following figures. Figure 3-12 shows a problematic high cost predicted week, regardless of the influential feature values such as higher number of the day of the week and temperature, the model failed to correctly predict the occupancy rate over 90% for Kurhaus.

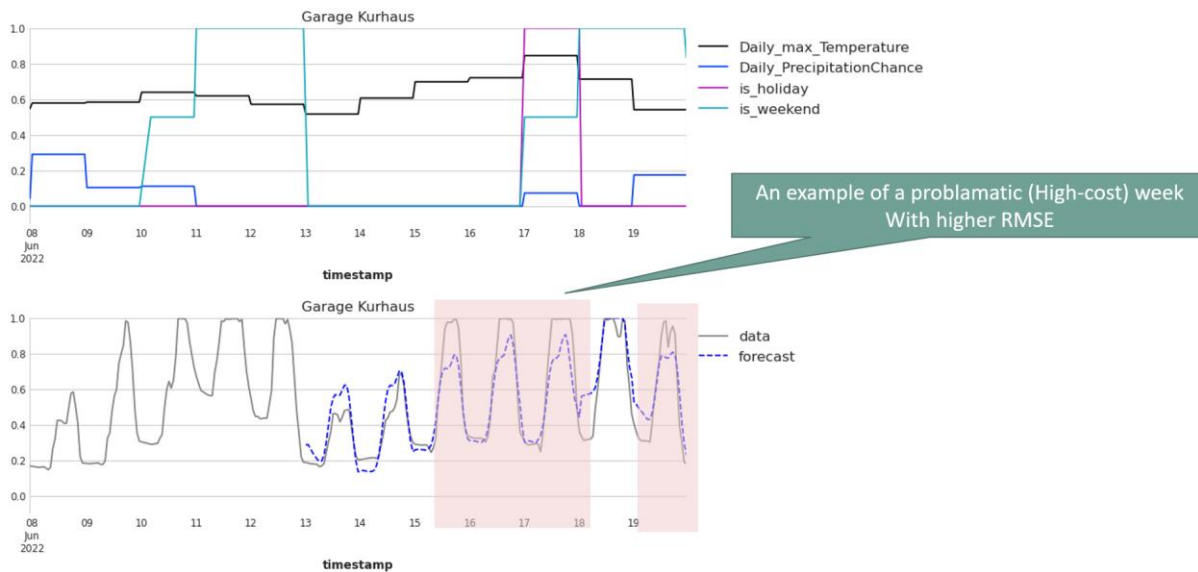


Figure 3-12: XGBoost failure example for Kurhaus

Figure 3-13 shows a problematic high cost predicted week due to expecting fewer visitors on a Wednesday and Thursday.

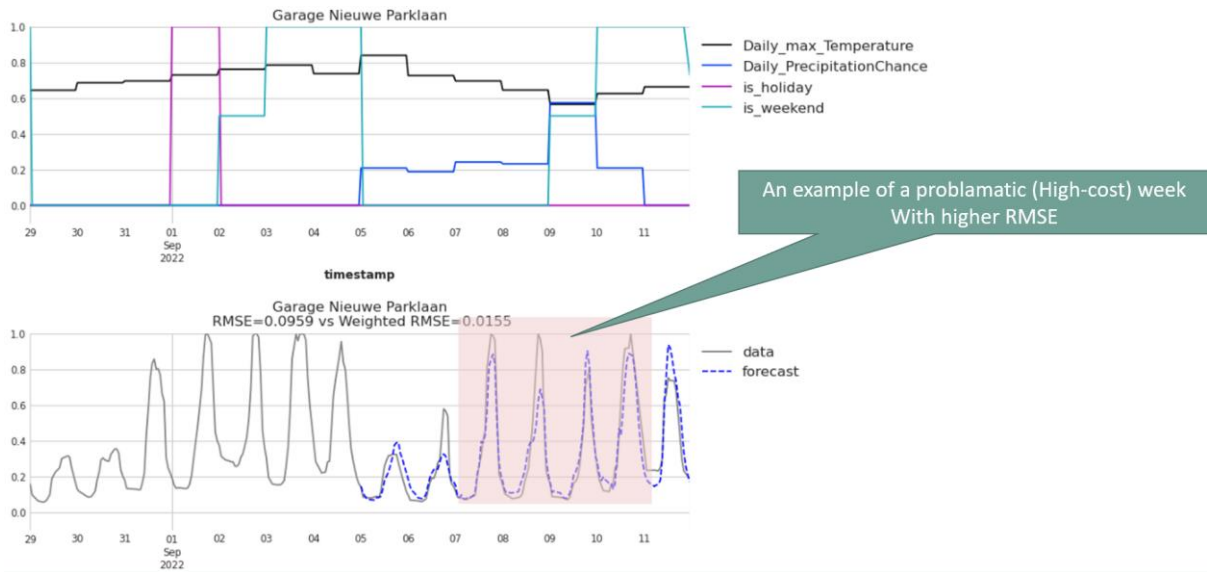


Figure 3-13: XGBoost failure example for Parklaan.

Figure 3-14 illustrates how a failure to predict high-occupancy in Kurhaus leads to underestimation of occupancy in Parklaan and consequently in Strand.

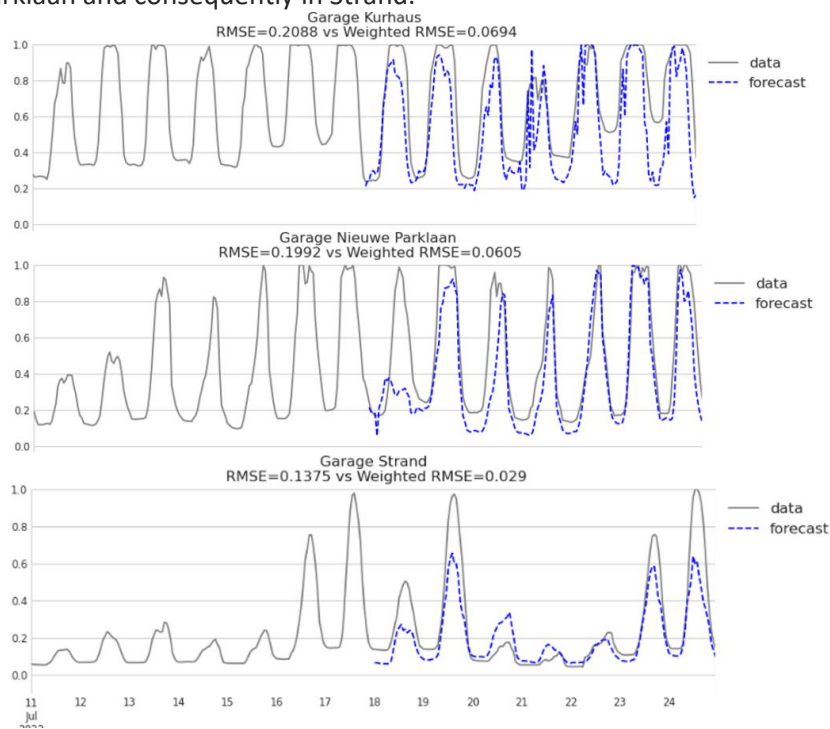


Figure 3-14: Forecast model failure cascade example for all three parking garages

3.2.3.2 Classification Models

We expanded our experiments by transforming the regression task into a classification task, with the objective of predicting which hours the parking will be either full or not full for the following week. This change in modelling approaches to classification was motivated by the use case need to focus on high-occupancy events when the parking garages reach maximum capacity. Moreover, classification models are widely explored in XAI research, and therefore more techniques are available to justify

their outcomes. To achieve this, we set a threshold of over 90% occupancy to define full occupancy. This threshold was discussed with our data providers.

Employing the same preprocessing steps as previously described, we trained a variety of classifiers, including AdaBoost, XGBoost, GaussianProcess, RandomForest, K-nearest Neighbor with looking only at three of the closest neighbours (K=3), and a Fully Connected Dense Neural Network (DNN) classifier. Across all three parking locations, the AdaBoost classifier demonstrated robust performance in capturing occupancy patterns, achieving an **average recall of 89.86%**. Specifically, it achieved a recall of 89.94% for Kurhaus, 85.07% for Parklaan, and 94.59% for Strand. Our findings underscore the importance of employing an ensemble of forecasting and classification models to improve prediction reliability and facilitate informed decision-making in resource planning and crowd management endeavours. The results are summarized in Table 17 and show that AdaBoost with highest recall proved to be the best model for capturing the occupancy behaviour.

| Parking garage | Model | F1 | Precision | Recall |
|----------------|---------------------------|---------------|---------------|---------------|
| Kurhaus | AdaBoostClassifier | 87.14% | 89.97% | 89.94% |
| | XGBoostClassifier | 82.83% | 81.24% | 84.80% |
| | GaussianProcessClassifier | 86.32% | 84.06% | 89.33% |
| | RandomForestClassifier | 87.00% | 80.63% | 89.56% |
| | KNeighborsClassifier | 80.63% | 80.66% | 80.61% |
| | DNN | 45.20% | 41.25% | 50.00% |
| Parklaan | AdaBoostClassifier | 83.92% | 82.86% | 85.07% |
| | XGBoostClassifier | 79.05% | 81.32% | 77.15% |
| | GaussianProcessClassifier | 82.57% | 83.14% | 82.02% |
| | RandomForestClassifier | 80.90% | 85.43% | 77.60% |
| | KNeighborsClassifier | 80.97% | 83.63% | 78.76% |
| | DNN | 47.74% | 45.67% | 50% |
| Strand | AdaBoostClassifier | 89.44% | 85.38% | 94.59% |
| | XGBoostClassifier | 85.19% | 89.86% | 81.52% |
| | GaussianProcessClassifier | 90.95% | 89.94% | 92.02% |
| | RandomForestClassifier | 84.27% | 92.15% | 78.21% |
| | KNeighborsClassifier | 77.45% | 89.78% | 71.01% |
| | DNN | 49.80% | 49.62% | 49.98% |

Table 17: Classifier performance on all three parking garages

We conducted a thorough examination of the reliability of AdaBoost by delving into the prediction results based on the confusion matrix generated by this classifier (see Table 18). Given the highly imbalanced nature of the data, with a prevalence of not-full observations over full observations, we observed that the model effectively distinguishes between the two cases. However, most of the False Negatives were observed with a one-hour delay, likely influenced by the previous hour's occupancy values of the parking itself and its adjacent parking. This is illustrated in Figure 3-15-Figure 3-17. Furthermore, we analysed the model confidence and observed that the confidence levels for many False Negative samples were notably low, indicating that these samples were situated close to the model's decision boundary and represented non-trivial cases. Leveraging Explainable AI (XAI), we can

delve deeper into the reasons behind these misclassifications (see T4.2 “Active Learning & XAI for Crowd Prediction” emerald).

| | Kurhaus Predicted | | Parklaan Predicted | | Strand Predicted | |
|------------|-------------------|--------|--------------------|--------|------------------|--------|
| True Label | < 90% | >= 90% | < 90% | >= 90% | < 90% | >= 90% |
| < 90% | 1889 | 74 | 2185 | 73 | 2446 | 7 |
| >= 90% | 61 | 448 | 57 | 157 | 2 | 17 |

Table 18. Confusion matrices of the three parking models.

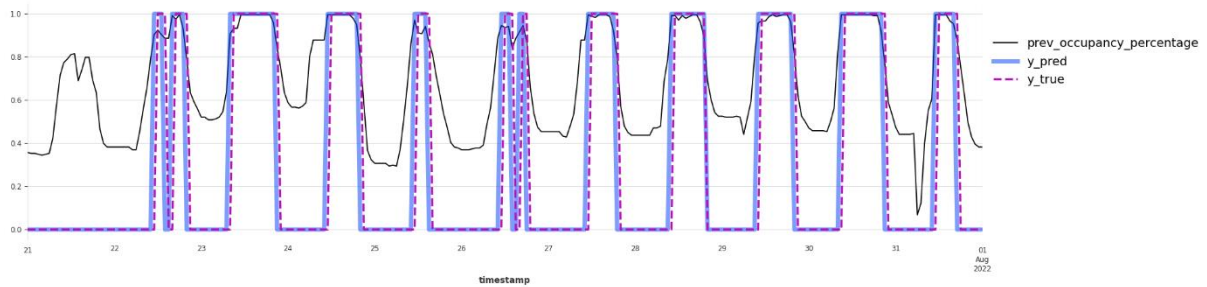


Figure 3-15: Prediction vs Actual classification labels of Kurhaus (AdaBoostClassifier).

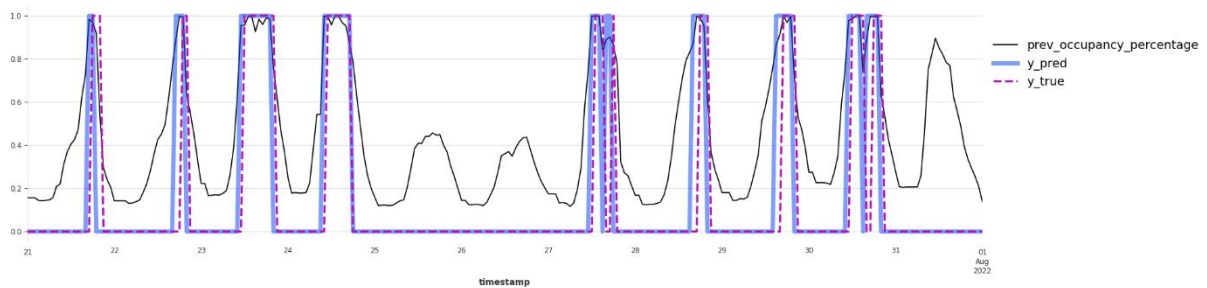


Figure 3-16: Prediction vs Actual classification labels of Parklaan (AdaBoostClassifier).

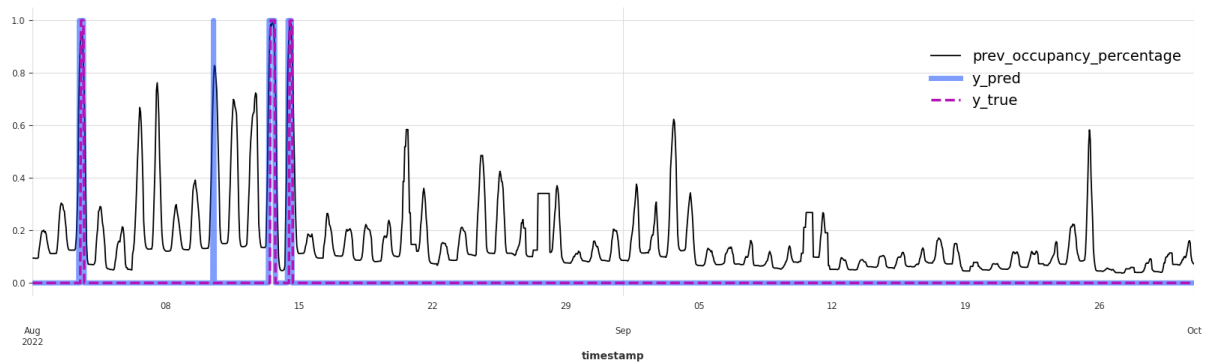


Figure 3-17: Prediction vs Actual classification labels of Strand (AdaBoostClassifier).

3.2.3.3 Summary

Since high-occupancy events are of priority for UC1, the recall values are the most important metrics for comparing models. The average baseline 1 recall is 79.93%. For regression, the average XGBoost recall is 73.52%. And for classification, the average AdaBoost recall is 89.86% (12.4% better than baseline and 22.2% better than XGBoost). We expect that XGBoost’s performance will further improve with respect to the naïve baseline as more training data becomes available.

While the classification models provide better high-occupancy predictions, the regression models are still valuable to estimate occupancy in lower-occupancy scenarios. These lower-occupancy scenarios may provide valuable information for other prediction models, such as the following T4.2 “Crowd Density Prediction” emerald since, for example, increasing parking occupancy is expected to correlate with increasing crowd density.

The following table summarizes the KPIs relevant to this emerald, as previously presented in D2.1. The KPI has already been successfully achieved.

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|----------------------------|--------------------------|---|-----------------------|----------------|
| Prediction accuracy | Naïve seasonal baselines | 10% improvement with regards to high-occupancy events | Precision & recall | 12.4% |

Table 19: Parking Garage Occupancy Prediction KPIs.

3.2.4 Next Steps

In the ongoing 1st integration and the assessment cycle that follows, the outputs of this emerald will be used as input for the T4.2 emerald “Crowd Density Prediction”.

In the 2nd implementation, integration, and assessment cycle, we aim to integrate Temporal Convolutional Networks (TCNs) and Long Short-Term Memory (LSTM) models, which have demonstrated robustness in capturing occupancy patterns based on our state-of-the-art analysis. However, it's worth noting that these models typically require a more extensive dataset to effectively capture fully occupied phenomena and the conditions under which they occur. Given that we currently only have data from one summer period, it may not suffice for training these complex models adequately. Therefore, our focus will be on acquiring additional data spanning multiple summer seasons or years to enhance the model's capability to capture and predict parking occupancy accurately under diverse conditions. This expansion of the dataset will enable us to leverage the full potential of TCNs and LSTMs for more accurate and reliable occupancy predictions.

Further improvements are expected to be gained through the XAI & AL process in the T4.2 emerald “Active Learning & XAI for Crowd Prediction”.

3.3 Crowd Density Prediction

This emerald utilizes historical crowd data and auxiliary data sources to predict crowd levels in specific areas. The goal is to accurately forecast crowd density patterns and evaluate the effectiveness of crowd management systems.

The crowd density prediction problem presents a spatiotemporal timeseries prediction problem, where the input comprises of a set of past crowd density observations and environmental factors (e.g., weather and events) represented in multiple timeseries, and the output is a prediction of future crowd density values.

Problem studied: given a set of N areas with C crowd density values at T historical time steps, denoted as $X = \{X_t \in R^{N \times C} | t = t_0 - T + 1, \dots, t_0\} \in R^{T \times N \times C}$, where t_0 is the current timestep, we aim to predict the crowd density for all areas in the next Q time steps, $\hat{Y} = \{X_t \in R^{N \times C} | t = t_0 + 1, \dots, Q\} \in R^{Q \times N \times C}$.

These predictions can be modelled as either regression or classification tasks. As part of the data science process iterations, we explored diverse modelling approaches. Based on use case domain expert feedback, the correct prediction of high-density events has higher priority than the prediction

of lower-density situations. We have therefore implemented both regression and classification models. This section provides:

- A brief overview of state-of-the-art models for crowd density prediction
- Introduction of the implemented prediction models
- First results on models' benchmarking and evaluation

Furthermore, these models serve as a basis for further advanced work as part of the T4.2 emerald "Active Learning & XAI for Crowd Prediction".

3.3.1 Brief Survey of the State-of-the-Art

Predicting crowd density is a vital task for urban planning, transportation management, and public safety. Accurate forecasts enable proactive resource allocation, traffic regulation, and crowd management strategies. They inform decision-making in emergencies, enhancing public health and safety measures. Thus, crowd density prediction optimizes urban infrastructure, resource utilization, and overall urban liveability. By anticipating crowded areas and potential congestion hotspots, authorities can proactively deploy resources, enhance public safety measures, and optimize urban infrastructure.

A notable emphasis on leveraging advanced AI methodologies, particularly deep learning, for predicting crowd density and traffic patterns exists in the current studies and related work. Different temporal models and timeseries data types are used for this purpose, and this state-of-the-art research underscores the critical role of advanced AI techniques in crowd density and flow prediction for applications ranging from urban planning to public safety monitoring; For instance, Jiang et al. [65], introduces DeepCrowd, employing Convolutional LSTM and attention mechanisms to forecast crowd density and flow within urban settings using a comprehensive human mobility dataset. Using trajectory data, Fu et al. [66], proposes a spatial-temporal convolutional model for crowd density prediction utilizing mobile-phone signalling data, adaptable to irregularly shaped areas. Addressing surveillance videos, Minoura et al. [67], presents patch-based density forecasting networks for crowd density forecasting, showcasing superior performance compared to existing approaches.

Some other works focuses on computer vision domain for crowd forecasting. For example, in the work of Zhao et al. [68], a model combining image processing and support vector regression is introduced to analyse crowd stability in public areas by predicting crowd density. Chen et al. [69], devise a time-dependent visiting trip planning framework, integrating deep learning for crowd density prediction with efficient trip planning. Ding et al. [70], enhance crowd counting accuracy and density map generation through an encoder-decoder CNN, surpassing current methodologies. Tackling overcrowding in Mumbai Suburban Railways, Sundaram et al. [71], proposes a GPS-based system for crowd prediction and monitoring.

Other work such as Wang et al. [72], introduce a simple crowd counting and localization network (SCALNet), demonstrating superior performance in crowd localization and counting tasks. Finally, Bhuiyan et al. [73], enhances crowd analysis for pilgrimages using a fully convolutional neural network, achieving high accuracy in crowd density classification.

3.3.2 Overview and Description

In this section we provide a brief overview of the implemented models. The following models were used:

- Forecasting / regression: naïve seasonal baselines, XGBoost, Prophet, Ridge regression, Decision Tree, and univariate ARIMA
- Classification: AdaBoost, GaussianProcess, RandomForest, K-nearest Neighbor with K=3, and a Fully Connected Dense Neural Network (DNN) classifier

The focus of the modelling stage is on determining the optimal combination of variables, including historical crowd density data, weather conditions, events, and other relevant factors, to refine the predictive capability of the model. All models are trained using historical crowd density timeseries, as well as event and weather timeseries (including hourly and daily observations of temperature, precipitation rate, precipitation chance, wind speed and direction, and cloud cover).

This spatial timeseries prediction problem distinguishes itself from the previously introduced T4.2 “Parking Occupancy Prediction” emerald, since the target value garage prediction has a hard maximum capacity limit while the crowd density doesn’t have such a natural limit. Furthermore, while garage models included the geographic overflow effect, the crowd density prediction models for individual areas do not include this effect since the relationship between the areas is less deterministic.

The experiments are implemented using Python and the Darts library, a specialized timeseries forecasting library and the evaluation is carried out using Darts historical forecast function, as illustrated by Figure 3-5.

3.3.3 Preliminary Evaluation

We conduct experiments on real-world data from use case 1 (UC1) comprising visitor data (“Resono” data), daily and hourly weather data, and calendar data. Initially, the visitor data consisted of approximately 380,000 observations over 17 areas (1 area describing the whole area of interest “Scheveningen volledig” and 16 subareas) in the time span from May 2021 to the end of November 2023. The weather dataset DT5 as recorded in D1.4 (the same as the one used for the parking data) includes hourly and daily observations of temperature, precipitation rate, precipitation chance, wind speed and direction, and cloud cover. Additionally, the calendar dataset provides information on national holidays in the Netherlands. Our goal is to predict the crowd density in the UC1 Scheveningen region over a 7-day horizon, providing hourly forecasts of expected visitor numbers.

To prepare the data, we merged all three datasets covering the period from May 2021 to the end of November 2023. We prioritize determining the most effective combination of variables, such as historical density data, weather conditions, events, and other relevant factors, to refine the model’s predictive performance. The data was pre-resampled on an hourly basis by the data providers, using the maximum visitor count as the reference. Subsequently, we computed the crowd density for each area by considering the surface area (m²) of the area and the hourly visitor count. Following this, we applied min-max normalization to each parameter to prepare the data for modelling purposes (see Figure 3-18), which only plots one month of this period for better readability.

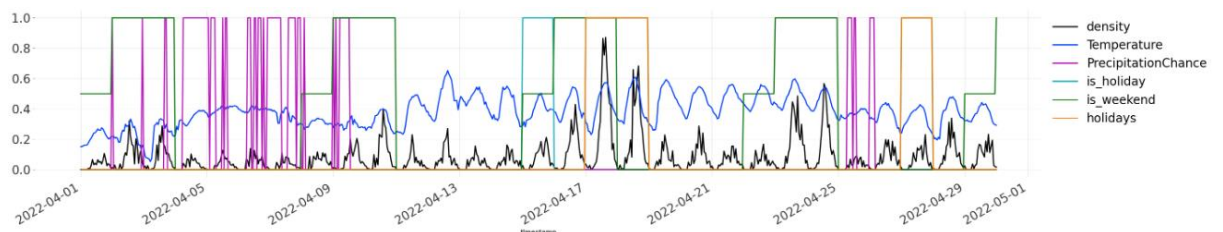


Figure 3-18: Visualization of the pre-processed crowd data.

Like the distribution of parking data, the distribution of the target variable “hourly visitor density” exhibits skewness. Furthermore, after min-max normalization, a considerable portion of values converge near zero, with sporadic anomalies like rare spikes (when the density is above 1 person per m²) in occupancy rates.

The initial models are trained on May 2021 to July 20th, 2022. The remaining time period until the end of October 2023 is used for evaluation (using Darts’ historical forecast, as shown in Figure 3-5).

Employing the historical forecast function, we predict a 7-day horizon and subsequently retrained the model for continuous weekly predictions.

To evaluate model performance, we selected the same evaluation metrics as in the “Parking Garage Occupancy Prediction” emerald (R^2 , RMSE, precision, and recall). These metrics provide a holistic assessment of model efficacy, accommodating the varied data distribution and the presence of outliers.

3.3.3.1 Forecast / Regression Models

The outcomes of our top-performing model XGBoost are summarized in Table 20. To evaluate our models' performance, we established a naïve seasonal baseline that predicts that the current week's occupancy will mirror that of the previous week (see an example for “Scheveningen volledig” in Figure 3-19). We see that our XGBoost model outperforms this baseline by an increase of 75.94% in “Scheveningen volledig” which is the most important area for the use case (UC1). However, the average performance of our model over all (sub)areas ($\overline{R^2}=0.0551$ and $\overline{RMSE}=0.0976$) is underperforming the Baseline ($\overline{R^2}=0.0855$ and $\overline{RMSE}=0.0687$) since XGBoost tends towards higher densities.

| Area | XGBoost | | Baseline | |
|------------------------------|---------------|---------------|---------------|---------------|
| | R^2 | RMSE | R^2 | RMSE |
| Scheveningen volledig | 0.5588 | 0.0786 | 0.3176 | 0.0822 |
| Boulevard Midden | 0.3821 | 0.0725 | 0.2513 | 0.0601 |
| Toegang Kurhaus | 0.4381 | 0.0774 | 0.4855 | 0.0575 |
| OV - Strandweg | -0.1967 | 0.0657 | 0.0329 | 0.0590 |
| Strand Centraal | 0.2681 | 0.1172 | 0.2533 | 0.0744 |
| Boulevard Zuid | 0.1666 | 0.0899 | 0.0833 | 0.0729 |
| Strand Zuid | 0.1780 | 0.0870 | 0.1382 | 0.0753 |
| De Pier | 0.1385 | 0.1025 | 0.1728 | 0.0575 |
| Toegang Scheveningseslag | 0.3143 | 0.0880 | 0.3715 | 0.0931 |
| Toegang Zeekant | 0.1586 | 0.1010 | 0.2620 | 0.0678 |
| OV - Kurhaus | 0.5081 | 0.0848 | 0.5004 | 0.07425 |
| Strand Noord | 0.1710 | 0.0921 | 0.1549 | 0.0614 |
| Toegang Zwarte Pad | -0.7569 | 0.1462 | -0.5402 | 0.0635 |
| Toegang Gevers Deynootweg | -0.0159 | 0.1130 | 0.1113 | 0.0959 |
| Boulevard Noord | 0.1488 | 0.1070 | 0.0711 | 0.0727 |
| OV - Zwarte Pad | -0.6645 | 0.1148 | -0.4213 | 0.0692 |
| Beach Stadium | -0.8595 | 0.1219 | -0.7898 | 0.0314 |

Table 20: Crowd density forecast summary results.

We conducted further analysis on the model residuals, as depicted in Figure 3-20, for the two areas Scheveningen volledig (best prediction performance) and Beach Stadium (worst prediction performance), including the problematic weeks where the density was much higher than what the model predicted. Analysing the errors for the whole areas, we also noticed that the model could predict the number of visitors in “Scheveningen volledig” area and areas close to “Kurhaus” better

than other areas. This is mostly due to the model's inability in predicting the peaks when the areas have a higher density.

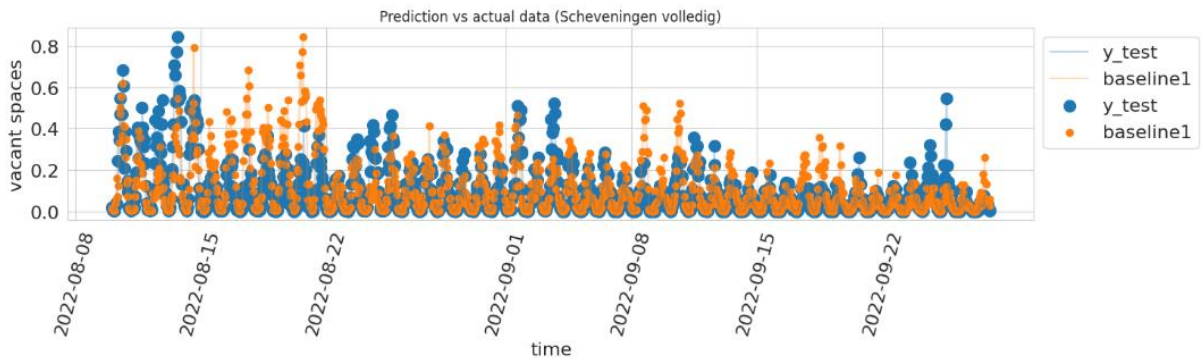


Figure 3-19. Crowd density baseline, which assumes that the expected crowd density for this week is the same as the previous week.

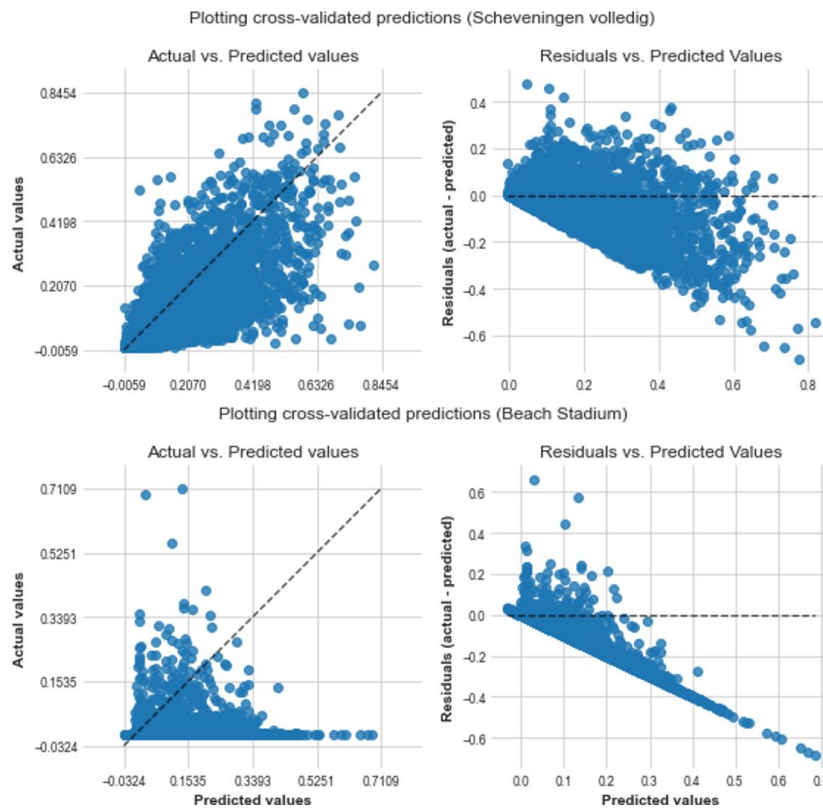


Figure 3-20: Comparison of the best (Scheveningen volledig) and worst (Beach Stadium) Residual Analysis of XGBoost.

We can compare the weekly prediction of the Scheveningen Volledig where the model successfully predicts the number of visitors close to the true values (See Figure 3-21). Only looking at the weekly prediction of the Beach Station we notice that the high prediction error is due to the model predicting higher visits when there were close to zero crowd available (See Figure 3-22).

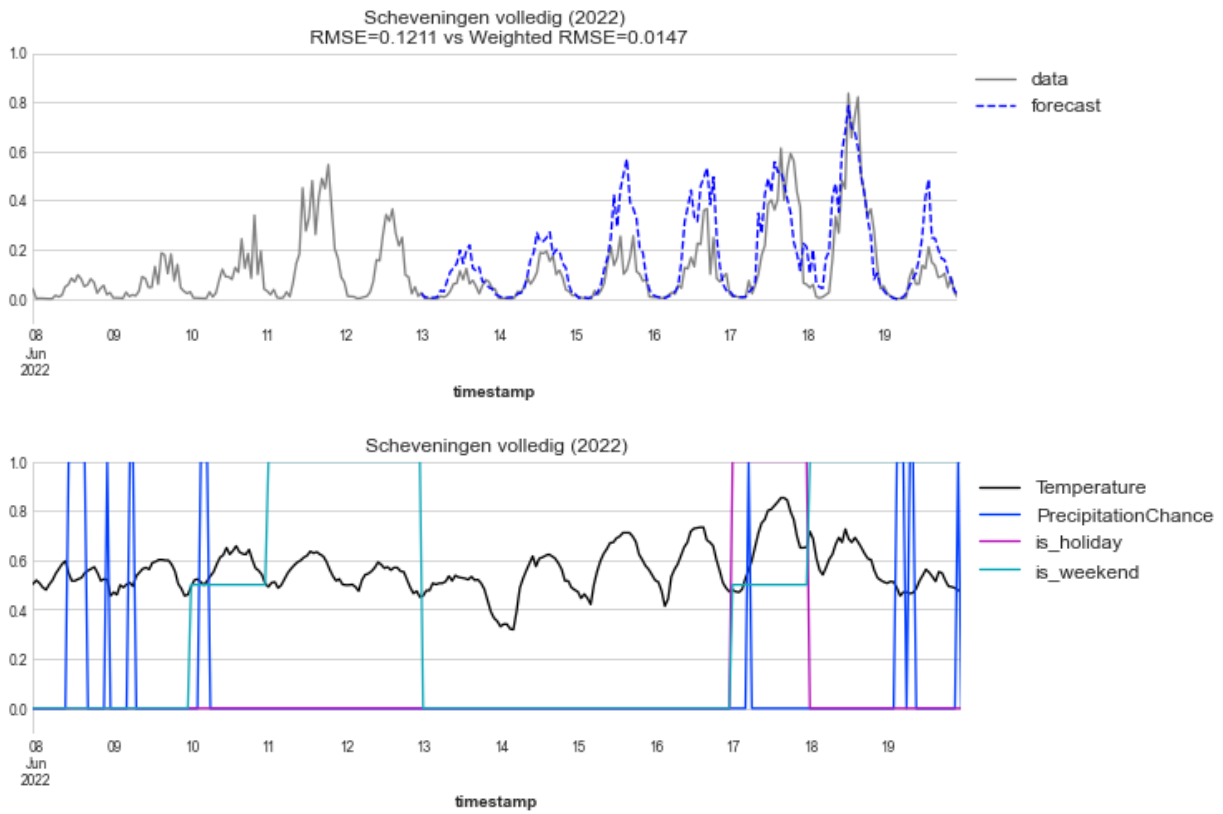


Figure 3-21: Example of very low prediction error for “Scheveningen volledig” area, including input features.

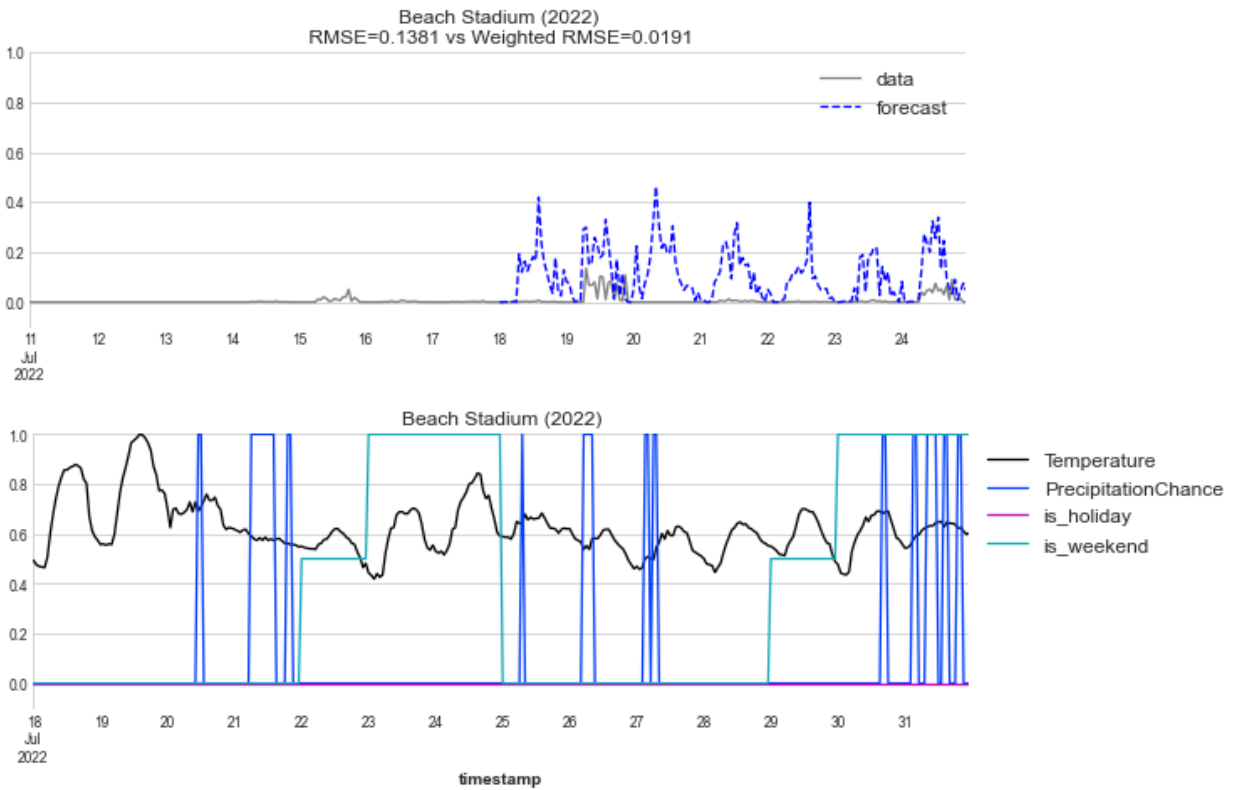


Figure 3-22: Example of high prediction error for “Beach Stadium” area, including input features.

3.3.3.2 Classification Models

Similar to the “Parking Garage Occupancy Prediction” emerald, we extended our experiments by transitioning the regression task into a classification task, aiming to predict which hours the crowd density would reach capacity or remain below capacity for the upcoming week. After receiving the input about critical density from our data providers, we only noticed that there are only two areas, where at some point in time the density of the crowd per surface m^2 increases to more than 1 person per m^2 . This is when considering a critical crowd as population/density more than 2 people per m^2 . For the sake of detecting a higher dense crowd, we created a binary class from the calculated density in way that when the density is equal or larger than 1 person per m^2 we consider a high dense population for that hour in the targeted area. Following the same preprocessing steps outlined previously, we trained multiple classifiers, including AdaBoost, XGBoost, GaussianProcess, RandomForest, K-nearest Neighbour with K=3, and a Fully Connected Dense Neural Network (DNN) classifier. Table 21 shows the results of this classification task for the crowd density data. For this classification task, unlike the parking occupancy prediction, the XGBoost model outperforms the other classifiers.

| Area | Model | F1 | Precision | Recall |
|-----------------|---------------------------|---------------|---------------|---------------|
| Toegang Kurhaus | AdaBoosClassifier | 82.46% | 85.07% | 80.28% |
| | XGBoostClassifier | 81.10% | 79.90% | 82.43% |
| | GaussianProcessClassifier | 82.98% | 86.53% | 80.18% |
| | RandomForestClassifier | 82.89% | 84.83% | 81.20% |
| | KNeighborsClassifier | 82.43% | 86.63% | 79.24% |
| | DNN | 47.88% | 45.93% | 50% |
| Toegang Zeekant | AdaBoosClassifier | 69.97% | 87.93% | 63.98% |
| | XGBoostClassifier | 70.96% | 74.28% | 68.52% |
| | GaussianProcessClassifier | 73.03% | 88.57% | 66.79% |
| | RandomForestClassifier | 70.89% | 82.53% | 65.70% |
| | KNeighborsClassifier | 68.38% | 81.38% | 63.37% |
| | DNN | 48.92% | 47.88% | 50% |

Table 21: Crowd density prediction classification results summary

We conducted a comprehensive analysis of the reliability of XGBoost by examining the prediction outcomes through the confusion matrix generated by this classifier (refer to Table 22). Similar to the previous subsection, given the imbalanced distribution of the data, where class “not critically dense” observations significantly outnumber the class “critically dense” observations, we observed XGBoost’s effectiveness in distinguishing between these cases. For both areas, Toegang Kurhaus and Zeekant, having Kurhaus a better modelled area.

| True Label | Kurhaus Predicted | | Zeekant Predicted | |
|------------|-------------------|------------|-------------------|------------|
| | < 1p / m2 | >= 1p / m2 | < 1p / m2 | >= 1p / m2 |
| < 1p / m2 | 2210 | 88 | 2351 | 39 |
| >= 1p / m2 | 64 | 139 | 65 | 41 |

Table 22: Confusion Matrices of two classifiers

Figure 3-23 and Figure 3-24 depict examples of False Negative predictions for Kurhaus and Zeekant, respectively. Using XAI we can investigate more about the reason behind these misclassifications. However, a quick examination of model confidence, we noticed that for half of the False Negatives (FNs), the model was very confident, and the other half the model was very unsure.

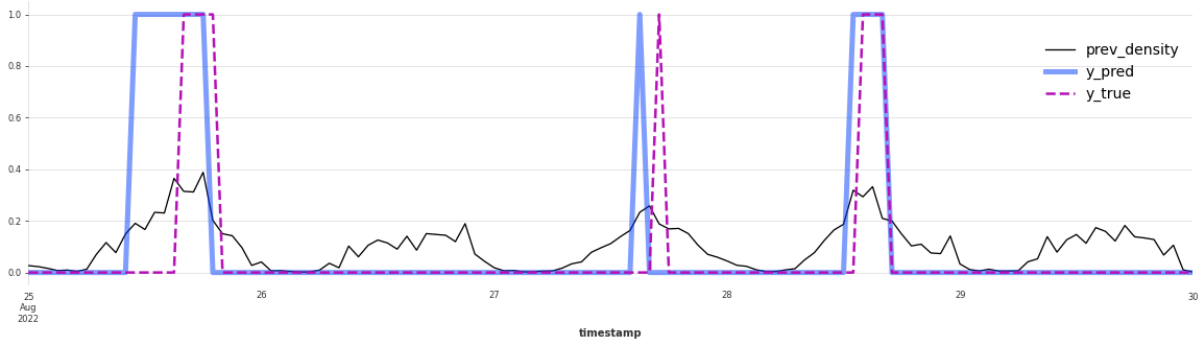


Figure 3-23: Example of misclassification for Toegang Kurhaus.

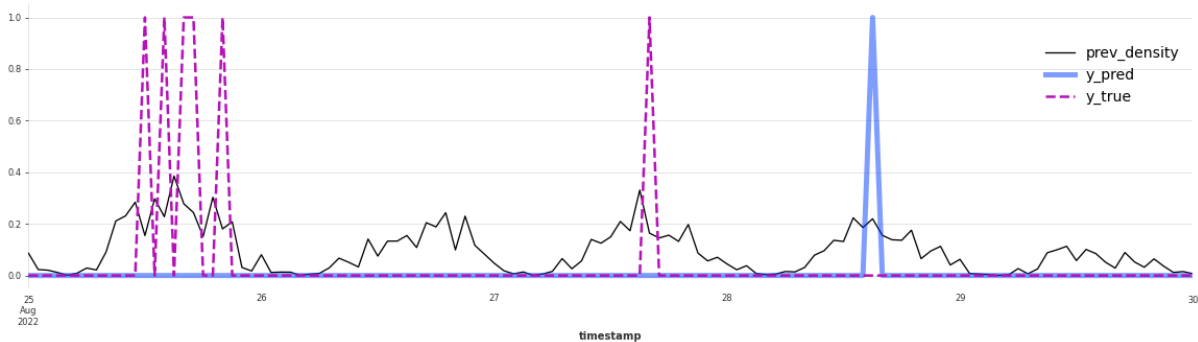


Figure 3-24: Example of False Negatives for Toegang Zeekant.

3.3.3.3 Summary

The following table summarizes the KPIs relevant to this emerald, as previously presented in D2.1. It is worth noting that the precision and recall values presented in the classification section above are based on preliminary assumptions from our use case partners about potentially critical crowdedness values. Therefore, the definition of critical crowdedness needs to be further refined in collaboration with our data providers and use case partners to appropriately reflect the real-world situation. A key issue is the spatiotemporal resolution of the crowd data. The areas vary significantly in shape and size and the crowd values are only available as hourly visitor counts per area, which are insufficient to determine the crowd density at a specific point in time. The refined definition will then be used to determine the precision and recall in future deliverables.

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|---------------------|--------------------------|---|-----------------------|---|
| Prediction accuracy | Naïve seasonal baselines | 10% improvement with regards to high-density events | Precision & recall | Tbd in the 1 st assessment cycle |

Table 23: Crowd Density Prediction KPIs

3.3.4 Next Steps

In the ongoing 1st integration and the 1st assessment cycle that follows, we aim to integrate the outputs of the T4.2 emerald “Parking Garage Occupancy Prediction” as further input to this model.

In the 2nd implementation cycle, we plan to incorporate Temporal Convolutional Networks (TCNs) and Long Short-Term Memory (LSTM) models into our framework, leveraging their proven effectiveness in capturing crowd density patterns, as indicated by our SotA analysis. However, the same challenge as the previous emerald, it is important to acknowledge that these models typically require extensive datasets to adequately capture fully occupied scenarios and the associated conditions. Despite the dataset covering 2.5 years, it may not suffice for such analyses, especially considering the impact of COVID-19 lockdowns in 2021, and partly in 2022, which influenced public event attendance patterns. Therefore, our primary focus will be on acquiring additional data spanning multiple summer seasons or years to augment the model's capacity to accurately predict crowd density under various conditions. This expansion of the dataset will empower us to fully harness the capabilities of TCNs and LSTMs, facilitating more precise and reliable crowd density predictions.

Moreover, we will intensify our efforts on preprocessing tasks to assign weights to areas experiencing higher density, enhancing the model's sensitivity to critical density levels, and further improving prediction accuracy.

Further improvements are expected to be gained through the XAI & AL process in the T4.2 emerald “Active Learning & XAI for Crowd Prediction”.

3.4 Active Learning & XAI for Crowd Prediction

This emerald harnesses explainable AI (XAI) and active learning (AL) techniques to establish a transparent and interpretable ML pipeline. This enables data experts to offer insights and enhancements to predictive models, thereby enhancing decision-making in urban planning and management systems.

AL optimizes the learning process by selecting the most informative data points for labelling, enhancing model accuracy with minimal human intervention. Meanwhile, XAI methods provide insights into model predictions, fostering trust and understanding by revealing the reasoning behind model decisions. By integrating these techniques, timeseries models can achieve improved accuracy, efficiency, and interpretability, leading to more actionable insights from data.

3.4.1 Brief Survey of the State-of-the-Art

In the following, we divide the analysis of the state of the art into two subsections: active learning and explainable AI state of the arts.

3.4.1.1 Explainable Artificial Intelligence State-of-the-Art

The literature on interpreting black-box models for timeseries using XAI methods is limited compared to other data types like images and text. Existing surveys by Guidotti et al. [74], Adadi and Berrada [75], and Hohman et al. [76]., highlight this gap, which is particularly noticeable when compared to methods available for images, text, and tabular data. However, recent efforts have begun exploring interpretability approaches for deep learning models in timeseries classification and very few for timeseries forecasting. For instance, Wang et al. [77], utilized deconvolutional networks to visualize meaningful representations, while Gee et al. [78], proposed an autoencoder-based approach for learning class representative prototypes.

Goodfellow et al. [79], employed class activation maps (CAMs) to highlight important aspects of timeseries data for decision-making. Additionally, Assaf and Schumann [80] utilized Grad-CAM for feature visualization in CNN-based regression tasks.

Wang et al. [81], introduced the multilevel Wavelet Decomposition Network (mWDN) along with an importance analysis method for timeseries interpretation. Furthermore, Hsu et al. [82], applied occlusion-based attention mechanisms for interpretable early classification of multivariate timeseries.

Recent work by Arnout et al. [83], compared various interpretability methods, such as saliency maps, Layer-wise Relevance Propagation (LRP), DeepLIFT [84], LIME [85], and SHAP [86], on LSTMs, fully convolutional networks, and ResNets. Their evaluation revealed differences in the effectiveness of these methods across different architectures.

Another XAI method for timeseries are counterfactuals. Hao et al. [87], for example, present a visual analytics framework that employs counterfactual explanations to clarify predictions within individual sliding windows. Their framework is equipped with interactive visualizations designed for large time series datasets and multiple variables. The framework functions through two stages: initially, transforming raw timeseries data into various representations during forecasting and explaining stages.

3.4.1.2 Active Learning State-of-the-Art

The examination of active learning (AL) in timeseries forecasting reveals a notable lack of emphasis on this domain, with most of the research centred around timeseries classification and anomaly detection. In regression tasks, the introduction of regression tree-based methods enhances model performance with limited labelled data, addressing the challenge of regression active learning [88].

For timeseries classification, nearest neighbour-based sampling strategies effectively utilize local information to measure uncertainty and utility, enabling classification with sparse labelled data [89]. In short-term load forecasting, a deep ensemble learning model integrated with an active learning framework dynamically selects key load segments for training, improving forecasting accuracy and mitigating data imbalance issues [90].

Anomaly detection benefits from reinforcement learning-based approaches, where models adaptively learn from real-world timeseries data, outperforming existing methods in detecting anomalies [91]. Multivariate timeseries anomaly detection sees advancements with active learning-based approaches that dynamically balance labelling costs and model performance, achieving superior results on public datasets [92]. These studies all highlight the versatility and effectiveness of active learning methodologies in improving model performance, adaptability, and scalability across diverse timeseries forecasting and anomaly detection applications.

3.4.2 Overview and Description

As shown in our SotA analysis, there is very little focus on XAI approaches for timeseries data and specifically for spatiotemporal data. The current XAI approaches are not able to explain the temporal changes in the data, neither in their outputs nor in a human understandable way. Moreover, these approaches are unable to incorporate the spatially specific explanations that are required for models trained on spatiotemporal data.

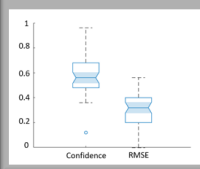

This emerald proposes a novel integrated XAI and AL approach for spatiotemporal prediction models, particularly for samples where the model exhibits uncertainty. In such cases, expert consultation and correct labelling are essential in the AL process. To improve the explanation of model decisions, we suggest that counterfactuals may offer a solution for better comprehension of model behaviour, enhancing the explanations' clarity for end users of such systems. We aim to integrate this improvement into an active learning loop, with the overarching goal of enhancing both the quality of the data and the performance of the models, while simultaneously refining the output of the XAI techniques. Figure 3-25 illustrates the envisioned framework designed to achieve this objective.



Interactive design

Choose model version

Model report as a boxplot:
RMSE: per week
Confidence: per week

Give an error threshold

Choose a problematic week




Interactive Design

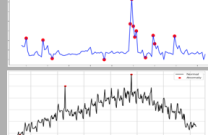
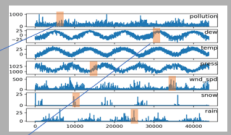
Interactive Quality Enhancement

Explanations of the influential parameters from train set

Where does the model fail?
Where does the model succeed?
How often a similar behaviour was observed?



Test/train set parameters:
Interactive analysis (correlation)
Correction of the test set parameters

Violated parameters with the expected range is shown



Interactive Design

Interactive Quality Enhancement

Are you sure you want to update the Train/Test set?

The model will be re trained

Expert keeps analyzing and/or keeps the existing version of the model

Figure 3-25: eXplainable interactive AL framework concept.

To establish a baseline for our novel XAI method, we apply SHAP for regression tasks and LIME for classification tasks, as described in the following Section.

3.4.3 Preliminary Evaluation

In this section, we evaluate the suitability of XAI methods for the previously introduced spatial timeseries prediction models “Parking Garage Occupancy Prediction” and “Crowd Density Prediction”. We apply SHAP for regression tasks and LIME for classification tasks, to analyse the models' performance. The implemented KPIs to measure the accuracy and effectiveness of the explanations include:

- **Fidelity score** measures the similarity or fidelity between the surrogate explanation model and the original black box model. A high-fidelity score indicates that the surrogate model accurately approximates the behaviour of the black box model.
- **Confidence level** measures the level of confidence or certainty exhibited by the explanation model when making predictions or classifications. Higher confidence levels indicate greater reliability of the explanations provided.

3.4.3.1 SHAP for Parking Occupancy Prediction

Using the SHAP technique, we calculated importance of each feature value on the prediction. An example of SHAP's visualization for Parklaan model is shown in Figure 3-26. In particular, Figure 3-26 displays that the lower the occupancy percentage of the previous hour (**Occupancy_percentage_target_lag-1**), the lower the calculated shapely value. Increasing value of the previous hour leads to increasing predicted value of the occupancy percentage for the next hour. Here we see also the effect of information given by Kurhaus occupancy (**Kurhaus_op_futcov_lag0**).

We also notice that the higher the date time (**darts_enc_pc_dta_hour_pastcov_lag-6**), the lower the calculated shapely value. This is trivial because the large values here are any time after 16-17, to 24 (in red). The very low values (blue) are for the 01 to early morning. The hours where the occupancy is higher are from range blue towards violate (7-8 to 15-16), hence it is expected to have more visitors at that time.

Moreover, The higher the value of the occupancy in the past 2-4 hours, the lower the calculated shapely value. This is an approximation that the visitors are most likely to leave the parking after 2-4 hours. Temperature and precipitation chance (**Daily_max_Temperature_futcov_lag-6**, **Daily_Precipitationchance_futcov_lag-6**) have lower feature importance compared to hour of the day and the parking occupancy lags. However, it makes sense that the higher temperature results in pushing the decision of the model towards predicting a higher occupancy percentage. This is vice versa for the precipitation chance. It seems that the days of the weeks (**darts_enc_dc_pta_day_of_week_pastcov_lag-1**) have almost zero influence on the model's decision. One solution might be weighting the days which could changes the influence of this feature on the prediction.

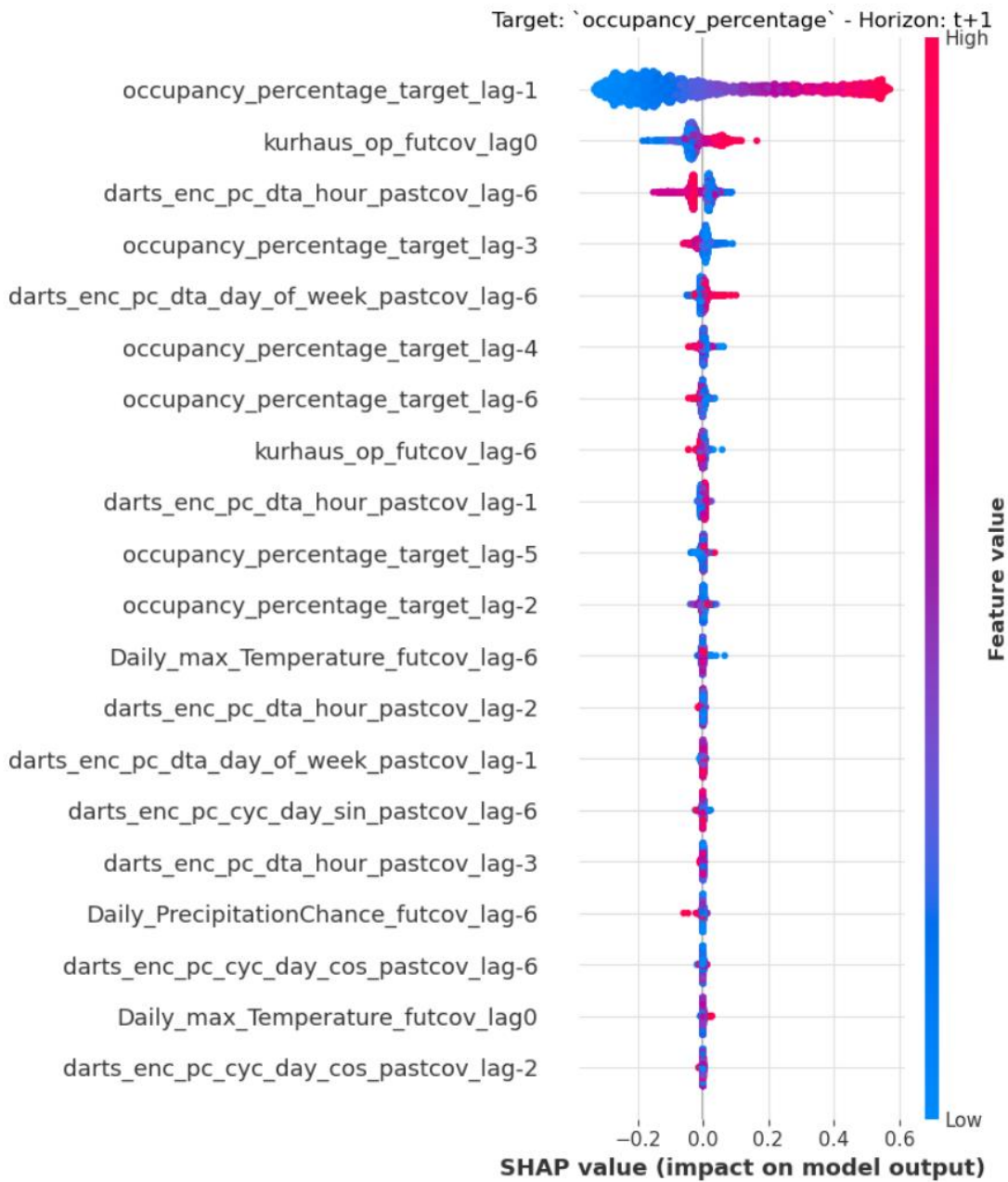


Figure 3-26: SHAP Visualization for Parking Occupancy Prediction of Parklaan.

3.4.3.2 LIME for Parking Occupancy Prediction

We proceed to elucidate the Parking prediction models for the classification task. Initially, we employ a global surrogate model, utilizing a decision tree, to gain insight into the overall estimation of the model's decision boundary. Focusing on the Parklaan model as a case study (depicted in Figure 3-27), we observe the remarkable fidelity of this surrogate model to the underlying black box model, boasting a Fidelity score of 99.67%. Notably, we identify the significance of previous occupancy values

and the Kurhaus occupancy percentage for the same hour in predicting the likelihood of Parklaan reaching full capacity. Conversely, other features in this context exhibit negligible influence.

| Prediction interval | Parking | XAI method | F1 | Recall | Precision | Balanced Accuracy |
|---------------------|-----------------|------------------------|--------|--------|-----------|-------------------|
| 22.07.21-22.11.01 | Garage Parklaan | Surrogate DecisionTree | 99.29% | 99.67% | 98.91% | 99.67% |

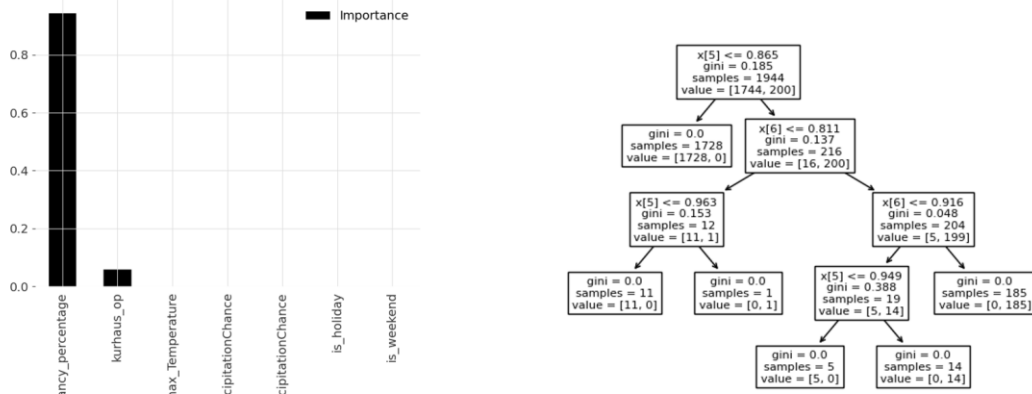


Figure 3-27: Illustration of the Global Surrogate decision Tree on Parklaan Adaboost Classifier.

Figure 3-28 shows LIME’s output for a true positive example. On the left side of Figure 3-28, we see the approximated model’s confidence (probability per class). In the middle part, we see that approximated model’s local decision boundary for this sample, the colour orange indicates the factors (with their coefficients) that contributes towards the class full (or here encoded as 1). The colour blue indicates the factors that contributes to the calculated probability for class not-full (here encoded as 0). We see only two significant contributors to the class full, previous occupancy percentage of Parklaan and the current occupancy percentage of Kurhaus (the adjacent parking that gets visited before Parklaan). This explanation also matches with the reality. The other factory is not contributing significantly.

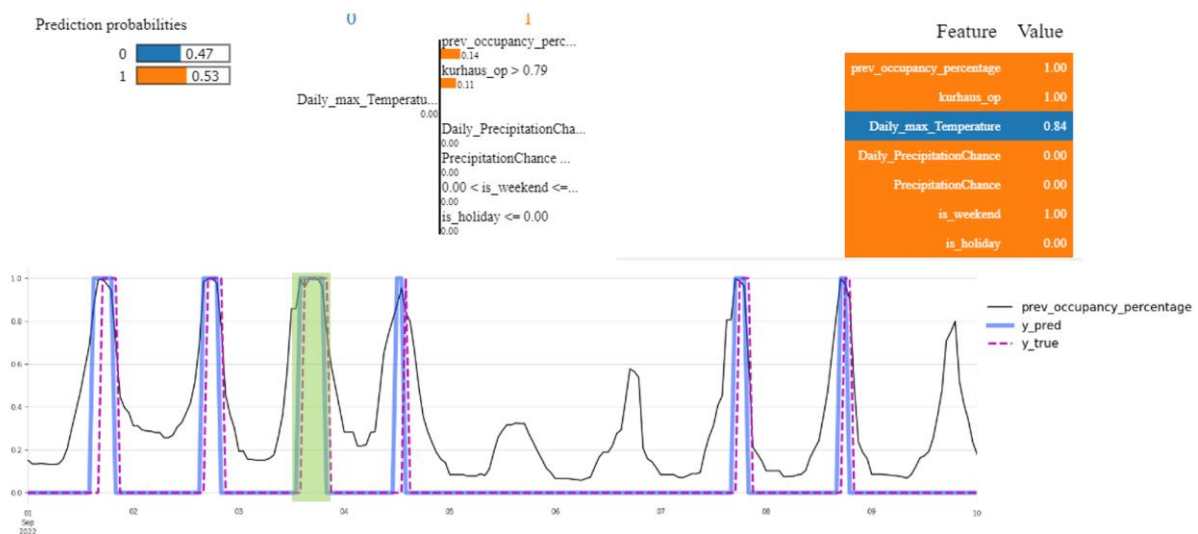


Figure 3-28. LIME's output on one true positive sample for Parklaan.

Another example is depicted in Figure 3-29, where the model failed to predict a full parking at 20:00, having the lower temperature and the higher precipitation chance pushing the decision of the model towards the class “not-Full” for Parklaan.

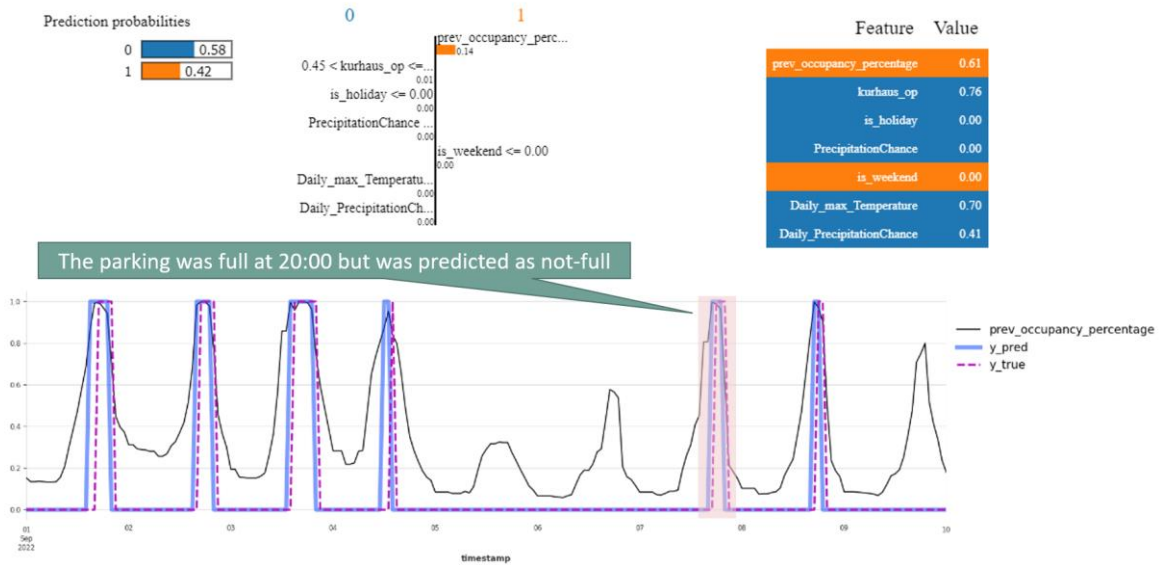


Figure 3-29: LIME's output on one misclassified (FN) sample for Parklaan in later hours of the day.

3.4.3.3 SHAP for Crowd Density Prediction

Employing the SHAP technique, we assessed the importance of each feature value on the crowd density prediction. A visual representation of SHAP's analysis for the Scheveningen volledig model is presented in Figure 3-30. The visualization illustrates that the lower density of the previous hour (*density_target_lag-1*), the lower the calculated shapely value. Increasing value of the previous hour leads to increasing predicted value of the density for the next hour. Moreover, The higher the date time (*darts_enc_pc_dta_hour_pastcov_lag-6*), the lower the calculated shapely value. This is trivial because the large values here are any time after 16-17, to 24 (in red). The very low values (blue) are for the 01 to early morning. The hours where the crowd density is higher are from range blue towards violate (7-8 to 15-16), hence it is expected to have more visitors at that time.

The higher the value of the crowd density in the past 2-4 hours, the lower the calculated shapely value. This is an approximation that the visitors are most likely to leave the area after 2-4 hours. Unlike the parking data, here the temperature (*Temperature_futcov_lag-0*) comes as the fourth most important, meaning that the higher the most likely the density will be predicted higher. Day of the week (*darts_enc_pc_dta_day_of_week_pastcov_lag-6*) also has a positive effect on the prediction (the weekends are most likely to be predicted with higher number of visitors). precipitation chance (*Precipitatiochance_futcov_lag0*) has lower feature importance compared to hour of the day and the density lags.

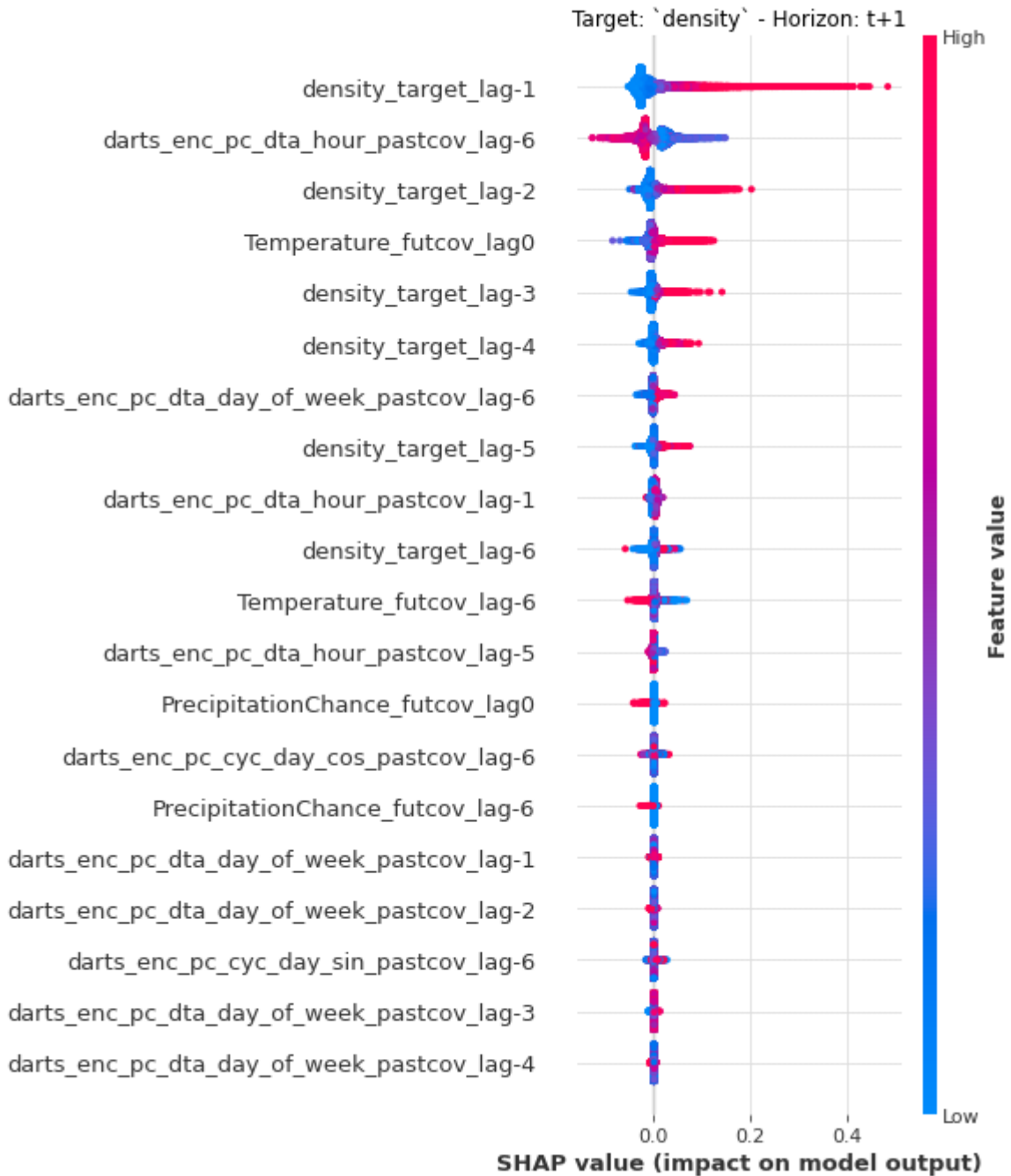


Figure 3-30: SHAP Visualization for Crowd Density Prediction of Scheveningen volledig.

3.4.3.4 LIME for Crowd Density Prediction

We now delve into the examination of critical crowd density classification models. Initially, we employ a decision tree-based global surrogate model to gain a comprehensive understanding of the model's decision boundary. Using the Toegang Kurhaus model as an illustrative example (referenced in Figure 3-31), we observe a high fidelity between the surrogate and black box model, with a fidelity score of 89.53%. In our findings we see the importance of previously observed density values, the Temperature, and whether it is on a weekend or not. Holiday feature and chance of precipitation also play a role on this classification task.

| Prediction interval | Area | XAI method | F1 | Recall | Precision | Balanced Accuracy |
|---------------------|-----------------|------------------------|--------|--------|-----------|-------------------|
| 22.07.21-23.11.01 | Toegang Kurhaus | Surrogate DecisionTree | 89.21% | 89.53% | 88.90% | 89.53% |

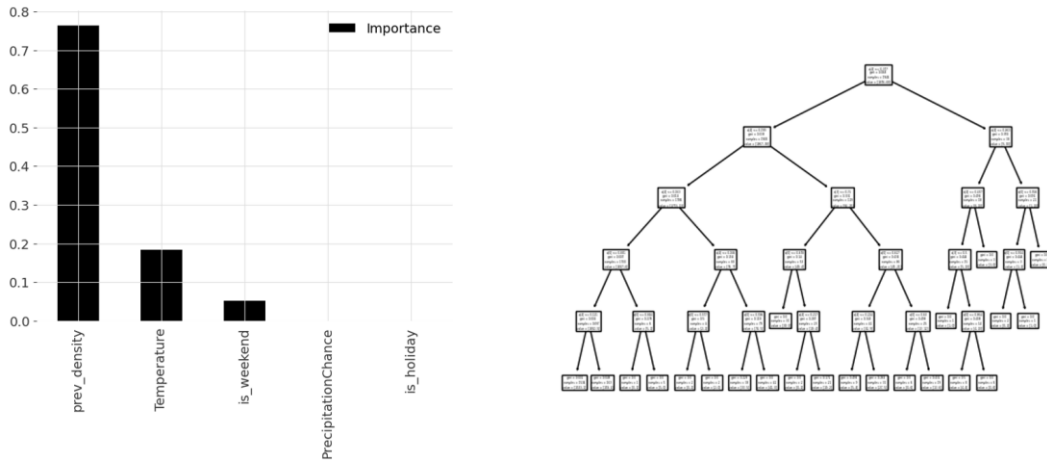


Figure 3-31: Illustration of the Global Surrogate Decision Tree on Toegang Kurhaus XGBoost Classifier.

Moving forward, we employ the LIME explainability technique to elucidate the instances of False Negatives within this critical density classification model. As depicted in Figure 3-32, we observe an illustrative output generated by LIME. Notably, the model exhibits higher confidence for this correctly classified sample (true positive for when the number of visitors is above 1 person per square meter). We notice that the model primarily considers the higher value of the previous hour and the Temperature in its decision-making process. The remaining features have no significance in the model’s decision.

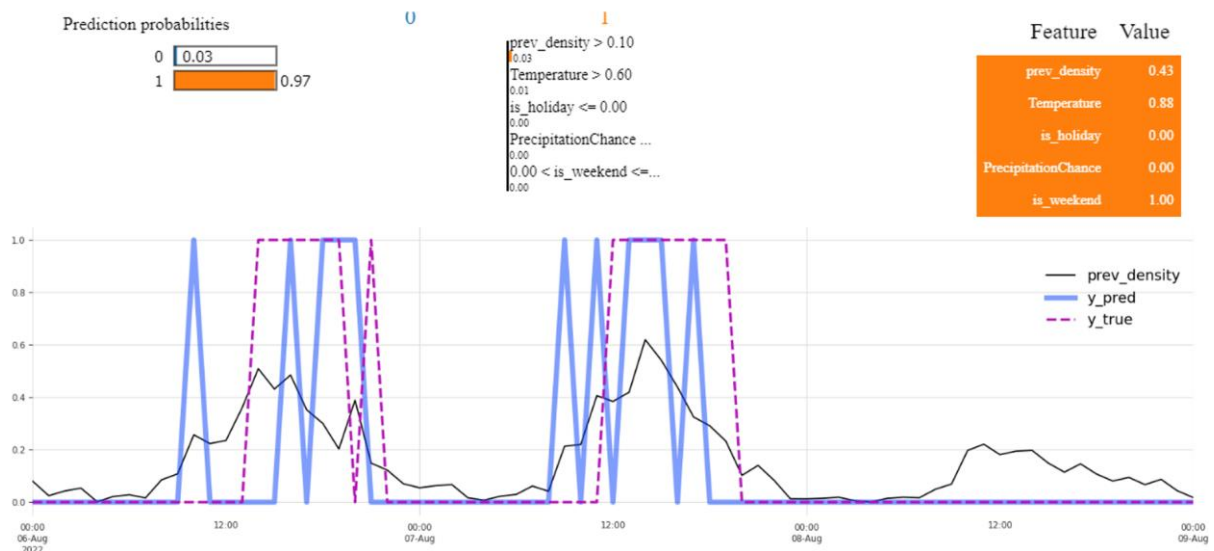


Figure 3-32: LIME's output on one correctly classified (TP) sample for Toegang Kurhaus.

3.4.3.5 Summary

The results presented in this section show that explanations provided by LIME and SHAP may not offer complete understanding, particularly for samples where the model exhibits uncertainty. In such cases, expert consultation and correct labelling are essential tools to improve model performance and trustworthiness through an active learning (AL) process. We advocate for the improved explanation of model decisions, suggesting that counterfactuals may offer a solution for better comprehension of

model behaviour than LIME and SHAP can provide, thus enhancing the explanations' clarity for end users of such systems.

The following table summarizes the KPIs relevant to this emerald, as previously presented in D2.1.

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|----------------------------|---|---------------------------|--|---|
| Explanation quality | LIME: 89.53% | >=95% | Fidelity Score (XAI) | Tbd in the 1 st assessment cycle |
| Prediction accuracy | Performance of "Crowd Density Prediction" emerald | Improvement over baseline | Comparison to the benchmark model without AL+XAI | Tbd in the 2 nd assessment cycle |

Table 24: Active Learning & XAI for Crowd Prediction KPIs

3.4.4 Next Steps

In the ongoing 1st integration cycle and the 1st assessment cycle that follows, we will follow and adapt to the ongoing developments of the spatial timeseries prediction models (Parking Garage Occupancy Prediction & Crowd Density Prediction).

In the 2nd implementation cycle, our focus will be directed towards implementing counterfactuals for XAI and integrate this improvement into an active learning loop, with the overarching goal of enhancing both the quality of the training data as well as and the performance of the models, while simultaneously refining the output of the XAI techniques.

In our further analysis, we will consider the similarity between the **Feature importances** using different XAI Techniques. Furthermore, end user's understanding of the feature importance provided by each XAI's output should be also taken into account, as it helps interpret the model's behaviour and identify the most influential factors. In addition, consistency and robustness should be considered as well. Consistency measures the stability and coherence of the explanations provided by the model across different samples or instances. Consistent explanations ensure reliability and trustworthiness in the interpretability of the model. Robustness evaluates the resilience of the explanation model to variations or perturbations in the input data.



4 Mobility AI-as-a-Service

The emeralds developed within WP3 and WP4 implement several ML models, specifically designed, and adapted to cater customized mobility data processing, analytics and intelligence services to the involved smart cities use case partners. EMERALDS Mobility AI-as-a-Service (MAIaaS) platform is intended to support these ML models' development and refinement, including training and testing stages, plus their deployment, supporting cloud/edge inferencing and/or models offloading to be run in use cases infrastructures. This is achieved through an agile approach considering design specifications, technical and business requirements. The EMERALDS MAIaaS platform constitutes a cutting-edge integrated compilation and customization of the novel MLOps tools, based, in turn, in DevOps techniques. This customization includes specific mobility tools, libraries, and interfaces to cover project level targets.

4.1 Brief Survey of the State-of-the-Art

In the field of software engineering, **DevOps** stands as a transformative paradigm, combining software development and IT operations to improve the efficiency, reliability, and speed of software deployment.

The core element of DevOps is the incorporation of **CI/CD pipelines** into software production processes. Continuous integration (CI) involves the automated integration of code changes from multiples contributors into a single software project, allowing developers to integrate code changes more frequently and reliably. This is often coupled with Continuous Deployment (CD), where after passing automated tests, changes are automatically applied and deployed to a production environment. Tools like **Gitlab CI**³³ or **Jenkins**³⁴ are important in this process, providing automation that ensures code quality, offering robust solutions for automating the build, test, and deployment phases.

Containerization is another fundamental aspect of DevOps. This technology encapsulates an application and its dependencies into a container that can be executed on any computing environment. The goal of containerization is to achieve scalability, portability, and efficient resource utilization while facilitating the deployment and integration of developed applications within heterogeneous infrastructures, such as edge frameworks or client premises **Docker**³⁵ has emerged as the leading containerization platform, providing an ecosystem for creating, deploying, and running containers.

For **managing containerized applications** at scale, **Kubernetes**³⁶ is used as the facto standard for container orchestration. It automates the deployment, scaling, and management of containerized applications, ensuring high availability and resource optimization. By managing clusters of hosts running containers, Kubernetes enables applications to run across multiple machines and environments, be they on-premises, in the cloud or hybrid setups.

Effective **monitoring** and **logging** are essential for maintaining the health and performance of applications and infrastructure. Tools like **Prometheus**³⁷ and **Grafana**³⁸ are widely used for monitoring

³³ <https://docs.gitlab.com/ee/ci/>

³⁴ <https://www.jenkins.io/>

³⁵ <https://www.Docker.com/>

³⁶ <https://kubernetes.io/docs/home/>

³⁷ <https://prometheus.io/>

³⁸ <https://grafana.com/>

metrics and visualizing data. Elasticsearch, Logstash and Kibana (ELK Stack³⁹) are commonly employed for logging and analysing large volumes of log data.

The evolution of DevOps has seen a significant shift towards **cloud platforms** like AWS⁴⁰, Azure⁴¹ and Google Cloud Platform⁴² (GCP). These platforms offer a range of services that support DevOps practices, including automated scaling, load balancing and managed Kubernetes services. This cloud integration has accelerated the implementation of IaC (Infrastructure as Code) as GitOps. IaC automates infrastructure management using code, easing automatic deployment, and reducing errors, with tools like Terraform⁴³ or AWS CloudFormation⁴⁴. GitOps extends Git’s version control to infrastructure, ensuring consistency and transparency.

The emphasis on **team collaboration** is mandatory in DevOps culture. These methodologies break down traditional silos between different departments, creating a culture of shared responsibility, continuous learning, and mutual support. The integration of development and operations teams leads to the formation of cross-functional teams. These teams are responsible for the entire software, from writing and testing code to deploying and monitoring applications. This approach ensures faster issues’ fixing, more efficient implementation of updates and improved software quality.

As machine learning (ML) has become a central key to software solutions, **MLOps emerges as a specialized field that applies DevOps** principles to the unique and special needs of ML projects, from data preparation and model training to deployment and monitoring (Figure 4-1). Apart from common DevOps practices previously mentioned, MLOps involves more than code: the dataset for model training, the model itself, metadata, and artifact.

In MLOps, continuous training of ML models is as crucial as continuous integration and deployment in DevOps. **Automated ML pipelines** are used for training, validating, and deploying models when the performance of the model is below certain threshold. This way, the best performing model is always deployed and available. Tools like **Kubeflow**⁴⁵ Pipelines facilitate these processes, integrating machine learning with CI/CD pipelines.

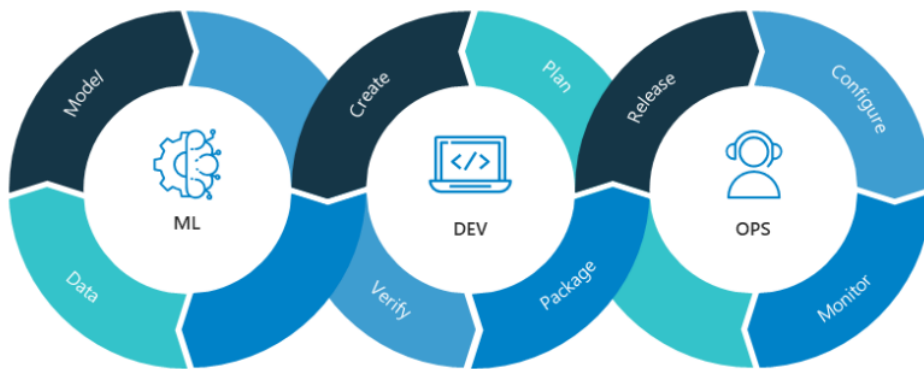


Figure 4-1: MLOps combine machine learning, applications development, and IT operations. Source: Neal Analytics.⁴⁶

³⁹ <https://www.elastic.co/elastic-stack>

⁴⁰ <https://aws.amazon.com/>

⁴¹ <https://azure.microsoft.com/en-us/>

⁴² <https://cloud.google.com/?hl=en>

⁴³ <https://www.terraform.io/>

⁴⁴ <https://aws.amazon.com/cloudformation/>

⁴⁵ <https://www.kubeflow.org/>

⁴⁶ <https://blogs.nvidia.com/blog/what-is-mlops/>

MLOps emphasizes the need for **tracking experiments** and **versioning** of data and models. DVC⁴⁷ (Data Version Control) is a great example that helps in versioning datasets and models, enabling reproducibility and easing collaboration among data scientists. **MLFlow**⁴⁸ is another good example for tracking experiments and versioning models. This way data scientists can track their progress and reproduce their trainings.

Scalability (increasing the number of nodes) and **Elasticity** (improving existing nodes) are significant challenges in MLOps, given the computationally intensive nature of ML tasks like Computer Vision, in which GPUs are needed to support this heavy workload. Kubernetes plays a key role in scaling and elasticity of ML workflows, while cloud-based solutions like Amazon SageMaker⁴⁹ or Google AI⁵⁰ platform provide managed services for training and deploying ML models at scale.

In post-deployment, ML models require **continuous monitoring** to ensure their performance does not degrade over time. Tools like **Evidently AI**⁵¹ or **TensorFlow Extended**⁵² (TFX) offer capabilities for monitoring model performance and automating the retraining process by executing ML pipelines as was mentioned previously.

The DevOps paradigm has revolutionized software development and operations, emphasizing automation, collaboration, and efficiency. This philosophy extends into MLOps, where the principles of DevOps are applied to meet the complexities of machine learning. It requires a deep understanding of ML workflows, data, and model lifecycle management. The collaboration between data scientists, ML Engineers and DevOps professionals is essential to address these challenges and get the full potential of MLOps.

In line with the EMERALDS project requirements in the context of mobility, MLOps plays a key role in improving and facilitating transportation and mobility solutions. By applying the principles aspects previously mentioned of MLOps to mobility, the power of ML can be used, for example, to analyse great amounts of transportation data, optimize traffic flows, conduct condition-based monitoring on top of ITS assets and predict maintenance needs. These innovations bear the potential to improve efficiency and sustainability in urban mobility systems but also establish decentralized decision-making processes entailed in smart city applications.

4.2 MAIaaS Platform Overview

The EMERALDS Mobility AI as a Service (MAIaaS) offers a catalogue of specific mobility services, internally known as “emeralds”, which embed different AI technologies. Behind this catalogue, there is an MLOps architecture intended to develop, train, deploy and serve these emeralds. The project analyses and combines current state of the art technologies and tools to implement and instantiate this architecture, thus creating the corresponding framework to support the full MAIaaS platform.

In this sense, the MAIaaS whole platform is composed by a set of tools, prepared to support several ML frameworks running on a Kubernetes environment with specific enabling platforms for each specific feature. All these components are distributed under an open-source license as a project requirement.

- In terms of **tools**, the platform implements instances of:

⁴⁷ <https://dvc.org/>

⁴⁸ <https://mlflow.org/>

⁴⁹ <https://aws.amazon.com/es/sagemaker/>

⁵⁰ <https://cloud.google.com/ai-platform/docs/technical-overview>

⁵¹ <https://www.evidentlyai.com/>

⁵² <https://www.tensorflow.org/tfx>

- Jupyter Notebooks for development,
- min.io and MobilityDB for data storage,
- DataHub and DVC for data management,
- Prometheus, Grafana, and Evidently for performance monitoring.
- For ML models development, testing and training, the platform is ready to work with most of the common **ML frameworks**, currently offering:
 - PyTorch, Scikit-learn, TensorFlow and XGBoost for classic ML and
 - Flower for FL use cases implementation.
- To support these tools and frameworks instances, the EMERALDS MAIaaS platform runs in a Kubernetes cluster which also includes specific Kubernetes-based **enabling platforms**. These platforms, in turn, enable ML model development and deployment, supporting the full ML lifecycle:
 - MLFlow intended for experimentation,
 - Katib plus Kubeflow pipelines build and deploy ML models using Docker containers,
 - Kserve is used to manage models inferencing, and
 - KubeEdge deals with deployments in edge nodes and cloud synchronization.

These components are, in turn, organized in functional blocks to create the ML development and the production / deployment frameworks according to the the MAIaaS platform architecture (Figure 4-2) already introduced in D2.1 and instantiated in this deliverable.

Additionally, a Federated Learning (FL) module has been developed as a combination of development and production enabling platforms and frameworks that comply with specific requirements in terms of deploying and training environments.

The platform development framework is integrated with the project's GitHub repository, offering an easy way for project's AI developers to link with Jupyter Notebooks and enter their code into the MAIaaS platform to experiment with the selected ML framework.

In terms of datasets for training and testing, the platform connects with use cases data sources using an array of data processing and ingestion tools (reported in D3.1) to gather and store this required information.

Once the models have their first exploitable version, the distribution platform containerizes and deploys these models in the cloud inferencing (production) framework, in the FL module, if applicable, and leave a copy in the models' repository to be exported to the use case proprietary infrastructure (or enrich other EU catalogues such as the EU AI on Demand platform). The production framework supports the execution of the exploitable version of the ML models running the corresponding use case processes that rely on the cloud infrastructure.

A set of monitoring tools is provided, common to all frameworks, collecting and showing different metrics related to performance of networks, servers, and models.

Additionally, a Federated Learning (FL) module has been developed as a combination of development and production enabling platforms and frameworks that comply with specific requirements in terms of deploying and training environments.

Finally, results from model inference are extractable and can be integrated with external EMERALDS looking platforms, such as the CARTO Analytics as a Service environment, enabling further use case

implementation requirements and feeding tailored visualizations and dashboards. Details on the demonstration of integrated AI/ML services and cross-inference will be reported in D2.4.

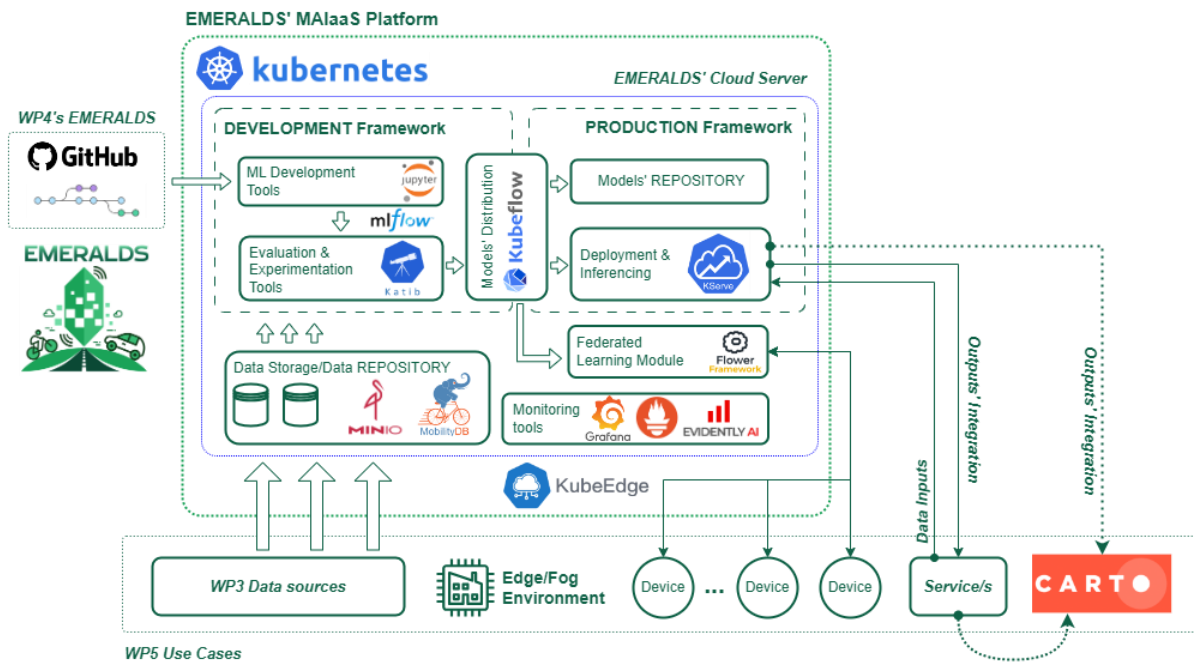


Figure 4-2: MAIaaS platform architecture – Logical view

The EMERALDS MAIaaS platform core is hosted and managed by ATOS and deployed using Kubernetes technologies. In this first platform version (M15) two Kubernetes clusters have been configured on top of the same physical servers' set, one to support the entire Kubeflow environment (Kserve, Jupyter Notebooks, Katib, etc) and the other one to host the rest of the tools (and frameworks) which are used and demanded by developers (MinIO, MobilityDB, Grafana, etc.) but both running the required enabling platforms to interconnect deployment and production environments, whilst supporting monitoring and FL modules. This configuration allows the experimentation with internal addressing and the evaluation of different integration approaches between frameworks. In addition, there exists a special machine implementing a Network File System (NFS) service devoted to support the data storage for both clusters. The resources assigned to each cluster are summarised in Table 25-Table 27.

| ID | Cluster | RAM | Cores | Storage | GPU (Virtualized) | O.S. |
|-----|----------|-------|-------|--------------|-------------------|--------------|
| 103 | Kubeflow | 32 GB | 30 | 0.5TB HDD | NVIDIA A40 (6 GB) | Ubuntu 22.04 |
| 104 | Kubeflow | 30 GB | 28 | 0.5TB HDD | | Ubuntu 22.04 |

Table 25: Kubeflow cluster resources

| ID | Cluster | RAM | Cores | Storage | O.S. |
|-----|---------|-------|-------|-----------|--------------|
| 105 | Apps | 16 GB | 10 | 0.2TB HDD | Ubuntu 22.04 |
| 106 | Apps | 16 GB | 8 | 0.2TB HDD | Ubuntu 22.04 |
| 107 | Apps | 16 GB | 8 | 0.2TB HDD | Ubuntu 22.04 |

Table 26: Application and tools cluster resources

| ID | Server | RAM | Cores | Storage | O.S. |
|-----|--------|------|-------|------------|--------------|
| 108 | NFS | 8 GB | 8 | 6.5 TB HDD | Ubuntu 22.04 |

Table 27: NFS server resources

To sum up, the EMERALDS MAIaaS is designed to implement the complete ML models lifecycle, from their creation and development to their deployment, offloading and inference in the use case scenarios. This has been realized catering to the use cases business needs and WP3/WP4 developers' requirements, supporting the data and ML pipelines of each emerald. In the first version of the MAIaaS platform the set of selected tools, frameworks and enabling platforms as in Figure 4-3 is arranged so as to validate this lifecycle in parallel with the development of the first versions of the WP4 AI/ML tools.

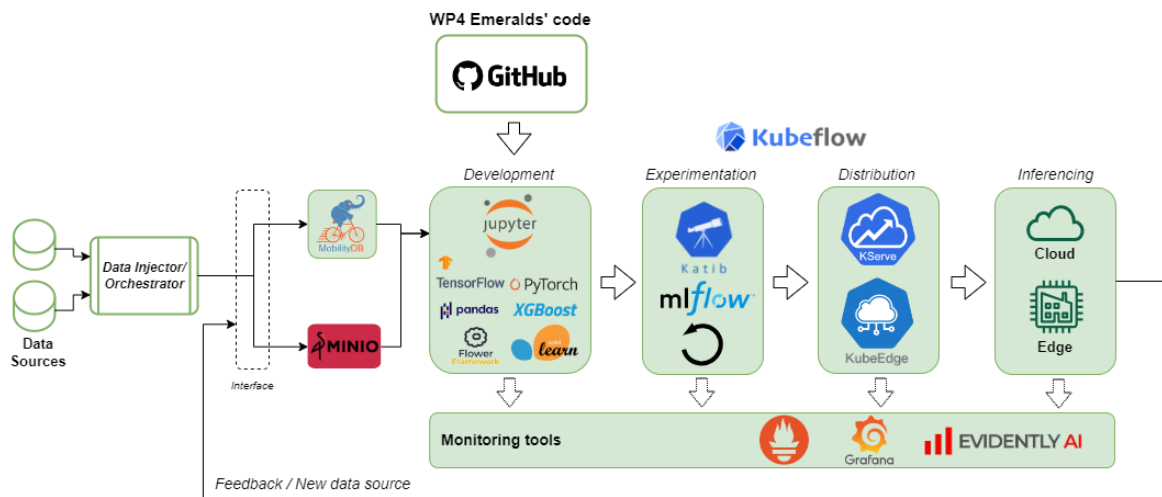


Figure 4-3: ML lifecycle implementation v1

4.3 MAIaaS Platform Components

Task 4.3 designs and provides a common platform for supporting the complete lifecycle of the ML models within each emerald developed by the project (Figure 4-3). This platform starts with the cutting edge MLOps technologies, analysing those tools and frameworks that better fit the use cases and developers' requirement. The target is to implement and offer a first customized instantiation that supports the initial versions of the ML models and mobility analytics as a service offering, including the distribution and exploitation of these models. This first platform version is to be evolved during the second stage of the project, together with the emeralds needs and towards its updated release.

The supported lifecycle covers the **data gathering and preparation**, jointly with WP3 tasks and use cases; the **model development**, including its planning, engineering, experimentation, and evaluation; their **deployment and catering** through the cloud and/or edge environment, using EMERALDS project and EU repositories; and finally, their **monitoring and maintenance**, to guarantee their best performance according to the gathered data. In this regard, the current implementation supports classic **ML development**, as well as **federated learning** approaches.

The following sections describe this first MLOps architecture version and the different technologies offered to support the emeralds on their different cycle stages, according to the specific use cases'

needs. To facilitate the understanding and use of this platform, a series of tutorials are available on the project's GitHub repository:

| Component | Repositories |
|---|---|
| Overall MAIaaS platform walkthrough | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/mlops-platform-tutorial |
| Description and usage of the DVC tool for datasets management | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/mlops-platform-dvc-tutorial |
| Templates for ML models' deployment | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/mlops-platform-kserve-template |
| Set of available Docker images for mobility data analysis | <ul style="list-style-type: none"> • https://github.com/emeralds-horizon/mlops-platform-Docker-images |

New tutorials (and updates) will be included as the EMERALDS MAIaaS platform evolves to support new features.

4.3.1 Data Management

The data management components include data ingestion interfaces and the data repository. Although the MLOps platform is not directly related to data processing or modification, it is a key part in the data management process, especially in data storage. It allows data to be highly available and accessible after data processing for eventual model training phases.

4.3.1.1 Data Ingestion Interfaces

The interface the platform presents is used in an easy way by WP3, which releases a set of tools responsible for data processing that in turn port data to the platform. To support data storage requirements, two key tools have been deployed on the MLOps platform:

1. The first tool is a **MinIO**⁵³ instance (S3-compliant object storage solution), which supports datasets in file formats such as CSV, JSON, Excel, etc. It has an intuitive user interface (UI) that allows users to easily upload datasets via drag-and-drop for training purposes (Figure 4-4). While the UI⁵⁴ is user-friendly, the usage of the pre-existing API⁵⁵ is more powerful to enhance efficiency and enable automated post-processed data ingestion. Users are provided with individual **credentials** for both **UI and API access**, along with private storage buckets per user for their experiments. In addition, a shared public bucket is available to share datasets among all stakeholders of the MLOps platform.
2. The second tool implemented is a **MobilityDB**⁵⁶ instance, which is a PostgreSQL-based storage that offers mobility functionalities which can be applied to data. This way, both relational data storage and mobility-specific functionalities are covered. Like MinIO, MobilityDB (which is also used in T3.3) is integrated within the platform, providing each user with a private database and access credentials. A shared database is also available for sharing datasets among users, facilitating collaborative efforts. Users usually connect to MobilityDB without a

⁵³ <https://min.io/>

⁵⁴ <https://minio-console.apps.emeralds.ari-aidata.eu/>

⁵⁵ <https://minio-api.apps.emeralds.ari-aidata.eu/>

⁵⁶ <https://mobilitydb.com/>

special interface⁵⁷, just using their usual MobilityDB/PostgreSQL login details via CLI. If needed, developers can choose to use different tools with UI to connect.

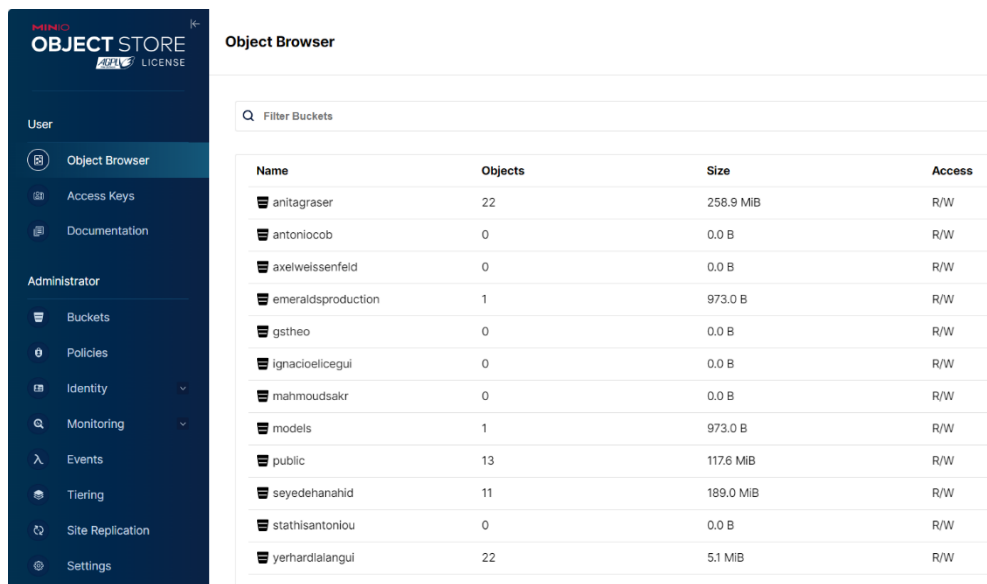


Figure 4-4: EMERALDS MAIaaS MinIO UI

These tools create a connection between data processing tasks and model training activities, ensuring processed datasets are readily available for training in an efficient and user-friendly manner, as displayed in Figure 4-5.

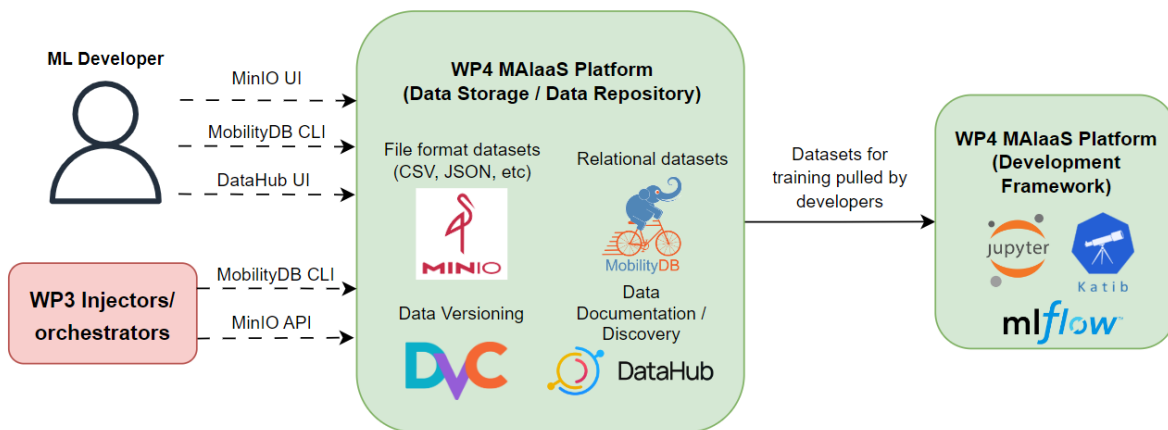


Figure 4-5: EMERALDS MAIaaS data management interfaces overview

4.3.1.2 Data Repository

The integration of MinIO and MobilityDB establishes a unified data repository for model training, creating a single access point for data retrieval. However, to ease the eventual steps and ensure access to datasets, it is important to document these datasets. Unfortunately, neither MinIO nor MobilityDB are focused on data discovery or documentation. To address this gap, **DataHub**⁵⁸ (Figure 4-6) has been

⁵⁷ mobilitydb.apps.emeralds.ari-aidata.eu

⁵⁸ <https://datahubproject.io/>

deployed on the platform to differentiate storage from documentation and discovery tasks. Users can access the tool’s web UI to see a list of all the datasets in the project⁵⁹.

DataHub automatically pulls metadata from both MinIO and MobilityDB by using their API (set up automatically without needing any action from developers), interpreting dataset characteristics such as data structure, size, and storage location. Users are recommended to add free text documentation related to datasets in DataHub, promoting the storage of only clean, high-quality, and training-ready data on the platform. This documentation includes important information such as dataset responsible, origin and any notes or considerations related to the data.

Another valuable tool available for developers is **DVC**. As we have mentioned before, this tool makes it really easy for users to keep track of different versions of their datasets and manage them efficiently.

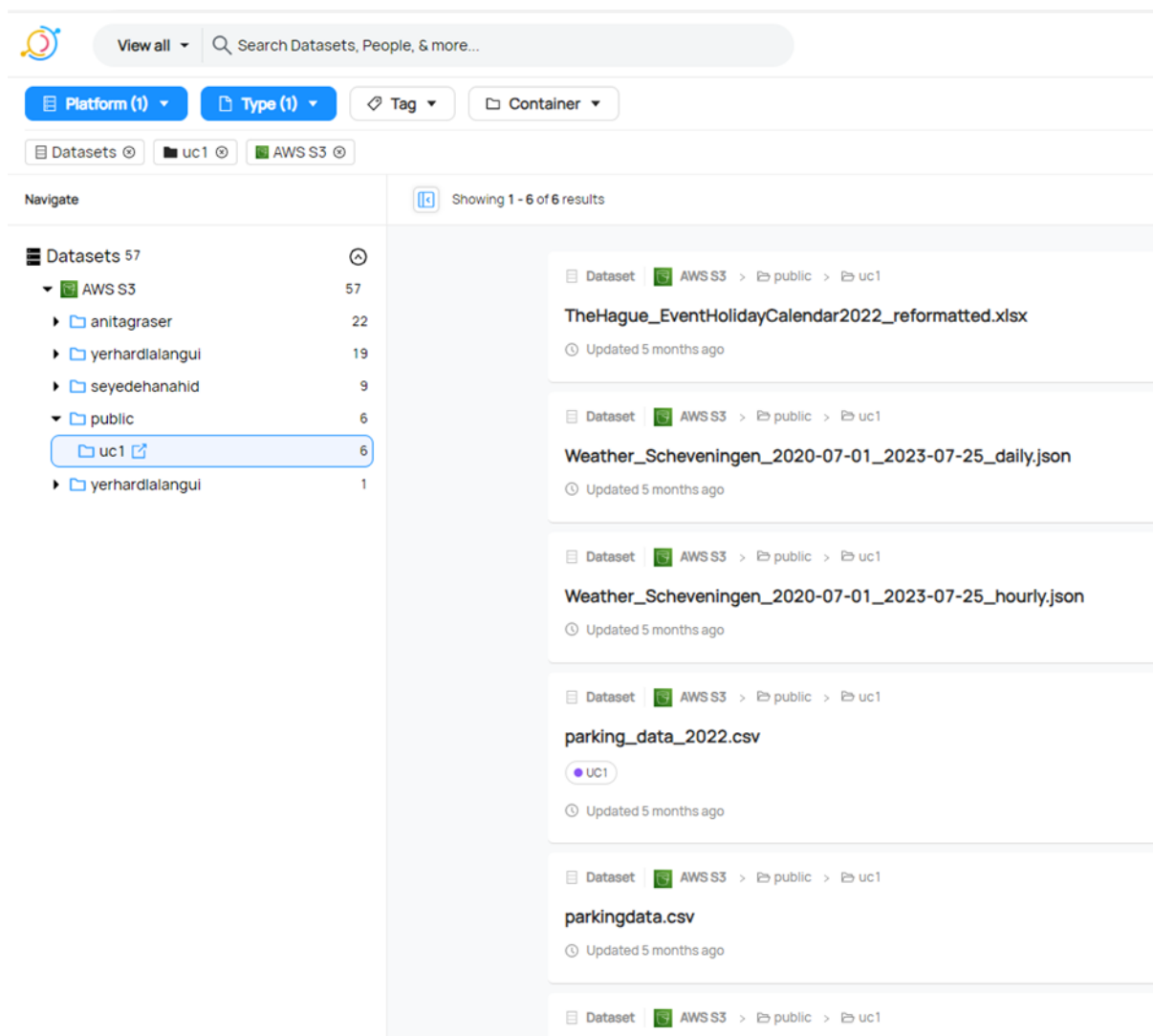


Figure 4-6: EMERALDS MAIaaS datahub example with UC1 datasets

4.3.2 ML Models Development Framework

The MLOps platform facilitates the development, experimentation, and evaluation stages of AI model creation. The tools deployed on the platform for these goals are currently being used by developers

⁵⁹ <https://datahub.apps.emeralds.ari-aidata.eu/>

and they are deployed in a Kubernetes environment. This architecture allows developers to leverage cloud resources and share knowledge, models, and data.

4.3.2.1 ML Development Module

Normally, AI model development is carried out by using Jupyter Notebooks⁶⁰, a well know tool in the AI community. However, developers often face challenges when they train models on personal computers, such as limitations in computational resources and difficulties sharing models, data, or knowledge. To address these issues, the platform offers a Jupyter Notebook environment that is deployed on Kubernetes, enabling the creation of containers with the developments. Jupyter Notebook is a flexible web tool that facilitates the use of most relevant ML development frameworks, as well as their combination and integration with the rest of the MLOps modules. Current instance of the Jupyter environment is configured to support XGBoost, PyTorch, TensorFlow, and Scikit-Learn, as ML frameworks, plus many other libraries to connect developments with the rest of the MAIaaS tools, such as DVC, MinIO, MovingPandas, hvPlot⁶¹ or GeoViews⁶².

Special mention should be made here of the **Explainable AI (XAI)** feature. As a requirement of the MAIaaS platform, and to support XAI for ML developers (specifically for the T4.2 emerald “Active Learning & XAI for Crowd Prediction”), the **SHapley Additive exPlanations (SHAP)**⁶³ libraries set is added to the Jupyter development tool. This software provides a wide range of explainers that allow the ML programmer to easily create visual functions to explain the output of their ML models.

The Jupyter deployments are carried out by Kubeflow, an MLOps framework that tackles some aspects of the MLOps lifecycle which include Jupyter Notebooks, automated hyperparameter tuning and model deployment. Kubeflow allows users to create their own containerized Jupyter Notebooks on demand with all necessary Python libraries pre-installed based on developer needs. This way, they enhance efficiency and scalability of AI model development. Development tools like Jupyter Notebooks or Katib can be accessed through a unified entry point by using Kubeflow. The project’s Kubeflow is also accessible to developers⁶⁴, offering a user-friendly interface (Figure 4-7).

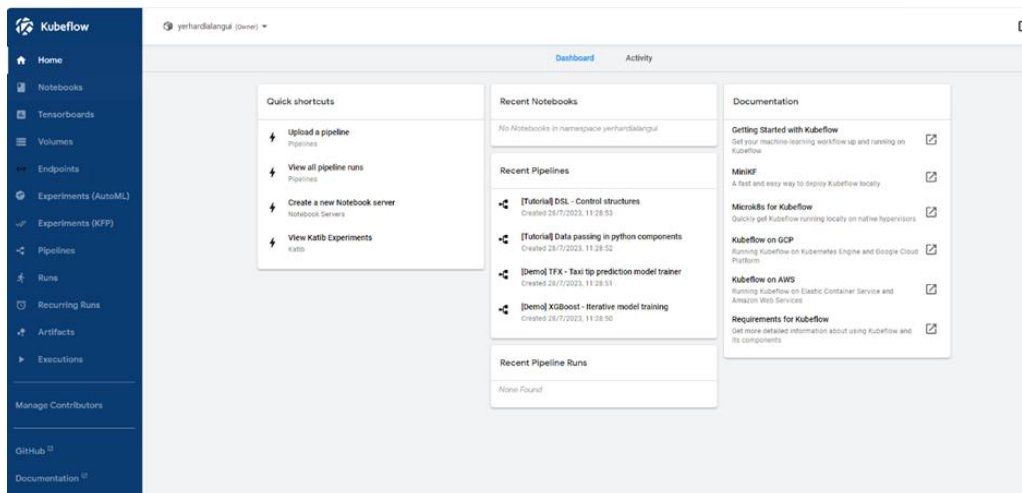


Figure 4-7: EMERALDS MAIaaS Kubeflow UI

⁶⁰ <https://jupyter.org/>

⁶¹ <https://hvplot.holoviz.org/>

⁶² <https://geoviews.org/>

⁶³ <https://shap.readthedocs.io/en/latest/>

⁶⁴ <https://kubeflow.emeralds.ari-aidata.eu/>

4.3.2.2 ML Evaluation and Experimentation Module

After defining the environment for model development, it is crucial to monitor developers' experiments to ensure reproducibility. To solve this task, MLFlow tool has been integrated into the platform, so each user has its own MLFlow server⁶⁵. It is a well-known tool within the MLOps community, which includes modules that speed up developer tasks such as experiment tracking and model registry and it has a user-friendly UI. This setup provides a centralized approach for documenting model experiments and managing models which will be deployed in a production environment.

To enhance the efficiency of the experimentation phase (hyperparameter tuning), parallelization is key. **Katib**⁶⁶ offers an optimal Kubernetes-based solution that automates hyperparameter tuning through containerization. Like this specific Jupyter Notebook approach, Katib is deployed along Kubeflow. This approach accelerates the experimentation process.

Within the platform, the already mentioned tools for development, evaluation and experimentation are seamlessly integrated (Figure 4-8). From their Jupyter Notebooks they can send information to the MLFlow tracking server or take advantage of Katib for parallel hyperparameter tuning.

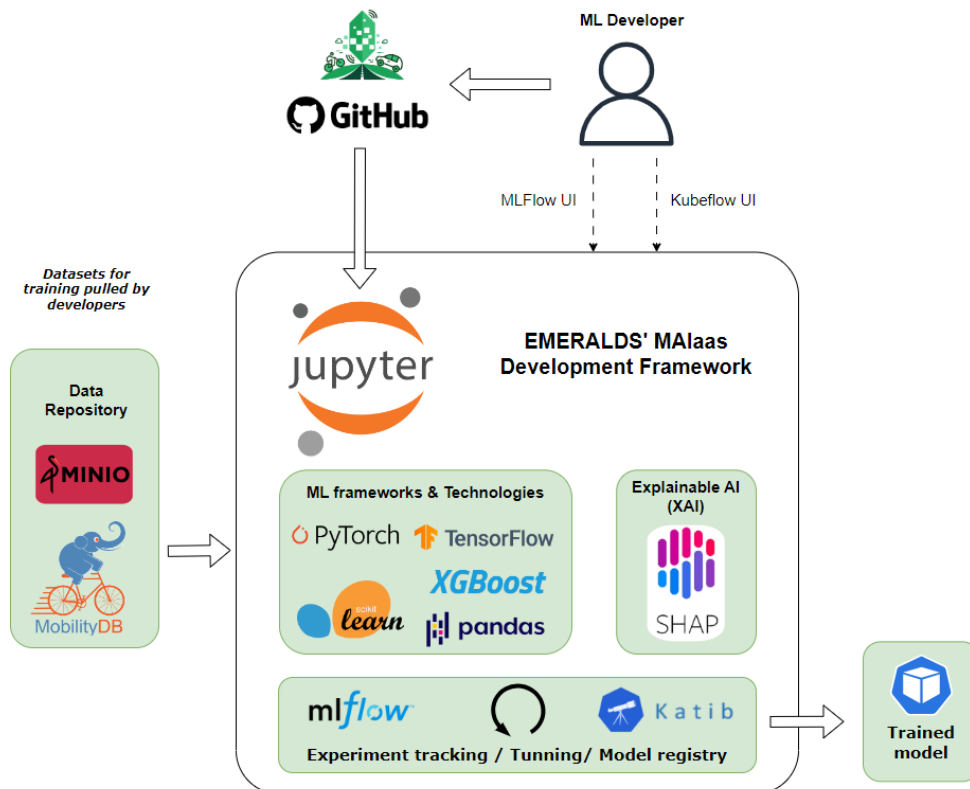


Figure 4-8: Model development overview.

4.3.3 ML Models Deployment (Production) Framework

Once a model has been trained, versioned and registered, it becomes ready for deployment in a production environment. The platform supports these deployment stages, offering solutions from

⁶⁵ <https://mlflow-username.apps.emeralds.ari-aidata.eu/>

⁶⁶ <https://www.kubeflow.org/docs/components/katib/overview/>

Kubernetes-based deployment to model monitoring, which (for example) could trigger model retraining actions if any decrease in model performance is detected (Figure 4-9).

4.3.3.1 ML Models (and Tools) Repository

The platform offers multiple methods for storing models prepared for deployment. Some developers opt to use the MLFlow Model Registry module, while others prefer a Docker registry, such as the private GitHub Docker registry⁶⁷ of EMERALDS project, or even public repositories like AI4EU⁶⁸. All these solutions are available within the platform. The most common method is via Docker registry. The EMERALDS project offers an internal Docker registry hosted on GitHub, which is integrated with the platform, so this enables push/pull Docker image actions from/to this private registry. For those who choose to use MLFlow or AI4EU the process is simple: They only need to provide their credentials. This way, models can be made available to be downloaded and executed locally, for example, as part of T4.1 edge analytics framework.

4.3.3.2 ML Deployment and Inferencing tools

To minimize human error, the model deployment within the platform is designed to be as automated as possible. This automation is related to both deploying and hosting models, supported by specific tools integrated into the platform. For deployment automation, Continuous Integration/Continuous Deployment (CI/CD) pipelines by using GitHub Actions⁶⁹ are used. These pipelines simplify the deployment process: once a developer has developed and registered their model, deployment is initiated by simply pushing code to the designated repository. This action triggers a CI/CD pipeline that automatically pulls the trained model, containers the model and deploys it within a Kubernetes environment, removing any complexity of Kubernetes-based deployment from developers.

For model hosting, Kserve⁷⁰ was integrated into the platform. It is a tool that automates and simplifies model deployment in Kubernetes environments. Kserve, like Katib and Jupyter Notebooks, is deployed along Kubeflow. This way, models are ready for inferences and hosted in the MLOps platform, leveraging Kubernetes robustness and scalability.

4.3.3.3 ML Monitoring Tools

It is common to see a model performance decrease over time, being periodic retraining needed. EvidentlyAI⁷¹ is a tool used to identify such data drift issues, providing insights into whether a model is maintaining its expected performance levels. Normally, ground truth label may not be available, so this monitoring technique is useful in these scenarios, which relies on analyzing the probability distribution if the data used for making predictions. A “shift” between the real-world data and the data used to train the model means that a retraining is needed.

In addition to tracking model performance, computational resources (such as CPU and RAM usage) that support the models are important as well. To monitor these resources, Grafana/Prometheus setup has been integrated on the cluster. This combination offers an overview of the platform’s performance, allowing for real-time monitoring of both model performance and computational resource usage.

⁶⁷ <https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-Docker-registry>

⁶⁸ <https://www.ai4europe.eu/>

⁶⁹ <https://docs.github.com/en/actions>

⁷⁰ <https://kserve.github.io/website/0.11/>

⁷¹ <https://www.evidentlyai.com/>

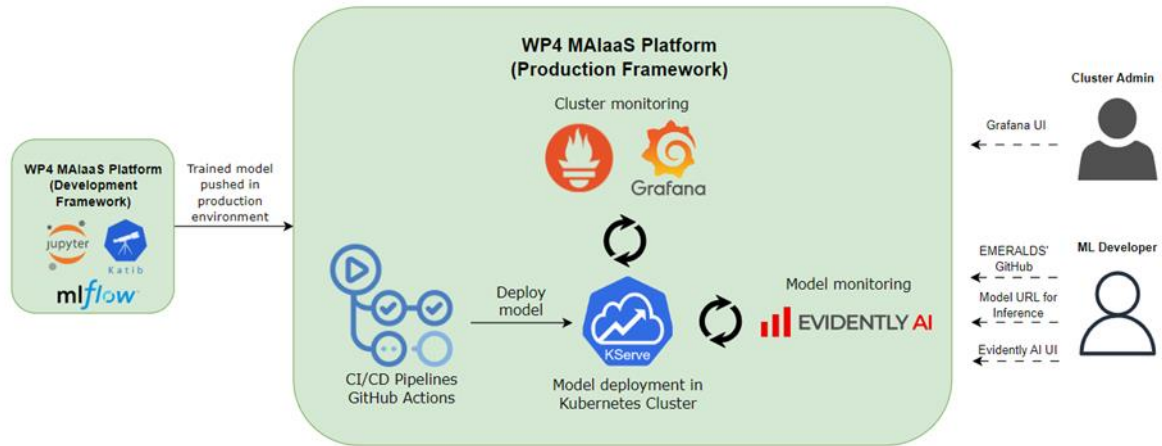


Figure 4-9: Model deployment overview.

4.3.4 Federated Learning Module

The EMERALDS project includes the development and adaptation of specific mobility services to be deployed and run within a federated learning environment. These are intended to work with distributed and sensitive spatiotemporal timeseries data, directly on the edge layer and so fostering privacy-preserving ML models' training and sharing. The project's MAIaaS platform also provides a framework for its AI scientist to develop and run these FL scenarios, fully aligned (and integrated) with the MLOps approach described in previous sections. This section provides an overview of the proposed technologies within EMERALDS to support these initial implementations.

According to recent studies and environments' evaluations [93,94,95] driven by the open-source principle and aligned with the project's MAIaaS cloud-kubernetes approach, it is proposed a FL baseline framework based on Flower⁷² for development, and KubeEdge⁷³ for deployment and edge layer management.

4.3.4.1 FL Development Framework

The development stage of the federated ML models covers the creation and initialization of the classic model within the cloud server as well as the definition of two new components:

- The **aggregation server** is located in the cloud side of the framework to collect the models' trained parameters from the registered edge nodes and aggregate them using a selected algorithm. Then, this new set of parameters is sent to the edge nodes to update the local versions of the models and start a new iteration.
- The **edge client** represents the code to be deployed in the edge node that updates the local version of the main model, trains it with the datasets in the edge source, and reports the new model's parameters to the aggregation server.

In this development context, Flower FL framework is intended to be used with most of the common ML frameworks (PyTorch, TensorFlow, scikit-learn, JAX, fastai, XGBoost, Pandas and more) to define the initial ML models and provide the tools to support the creation and synchronization of the

⁷² <https://flower.dev/>

⁷³ <https://kubedge.io/>

aggregation server and edge nodes, according to the use cases' scenarios. Flower is open-source and integrates with EMERALDS MAIaaS by using its Jupyter Notebooks tool.

4.3.4.2 FL Deployment Framework

The EMERALDS FL module relies on the Kubernetes deployment framework provided by its MAIaaS platform based on Docker containers and assisted by MLFlow tool for the cloud components and pods. The edge layer is to be integrated using the KubeEdge⁷⁴ system. This is an open-source extension for Kubernetes native frameworks that provides fundamental infrastructure support for network, application deployment and metadata synchronization between cloud and edge. By using KubeEdge, the MLOps platform can deploy and manage Docker containers on low-resource devices and low-bandwidth environments, handling networking and data synchronization between edge nodes and the aggregation server. This system also extends the security layer supported by Kubernetes, supporting SPIFFE⁷⁵ and SPIRE⁷⁶ mechanisms to ensure that only verified edge nodes can participate in data transmission, and so, in federation.

Combining the FL development and deployment frameworks within the FL module, the EMERALDS' FL overall approach is summarized in Figure 4-10. Flower plus the preferred ML software libraries will be used to design and code the model, the aggregation server, and the edge client, as well as the model parameters' flow, relying on Jupyter Notebooks. These will be packed into Docker containers and deployed as Kubernetes pods among the cluster's cloud and the edge nodes using Kubernetes + KubeEdge. During each single iteration, edge nodes are updated with the initial (or aggregated) set of model's parameters and triggered to train their model' versions with their local datasets to get a new set of model's parameters. These sets are updated in the aggregation server where the next common one is produced, and a new iteration can start. After a programmed number of iterations, the last version of the parameters can be implemented in the original ML model and can be distributed using the corresponding CI/CD pipeline in the MAIaaS platform.

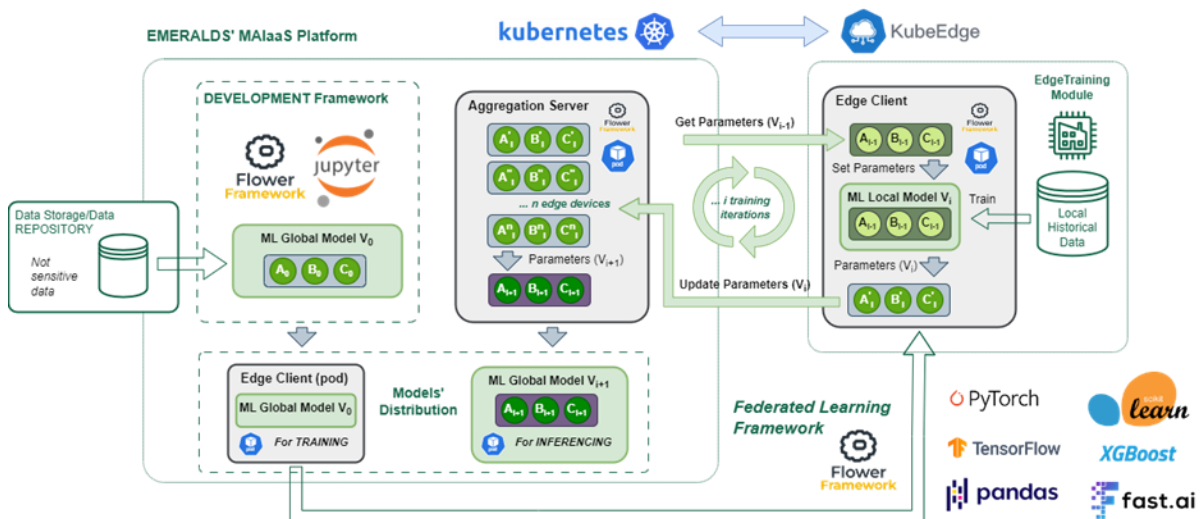


Figure 4-10: MAIaaS Federated Learning module – logical view.

⁷⁴ <https://kubedge.io/>

⁷⁵ <https://spiffe.io/>

⁷⁶ <https://spiffe.io/docs/latest/spire-about/spire-concepts/>

4.4 Preliminary Evaluation

As already defined in D2.1, the MAIaaS platform is to be evaluated and validated from two different approaches: a) the platform performance, which measures e.g., the execution, processing and deployment times or the number of concurrent processes running; and b) the platform usage, considering the number of stored datasets, models trained and developed or platform users.

Platform's performance metrics are intended to validate the proper integration and execution of the different frameworks, tools and enablers deployed to cover the MAIaaS requirements. In this sense, a) **Data Ingestion Speed** is considered as the time taken for data to be ingested and available for ML usage within the internal repository (MinIO). The goal is to achieve speeds similar to platforms such as AWS or Azure, commonly used for DevOps platforms; b) the Number of Models in Training Phase has to do with the capability of the platform to support concurrent models' training and so, parallel emeralds' development. In this line, the number of training processes will be monitored (**Number of concurrent models in training stage**), considering those that do require a GPU and those that not, and compared with the whole models' performance to check its impact; c) this is to be replicated for the Number of models in production within the inferencing platform (**Number of concurrent models in production - cloud**) with and without GPU; d) for federated learning, it is interesting to provide the platform with the ability to integrate edge infrastructures and to know what are their limitations. To validate the platform's scalability and capacity to handle distributed devices (and build a FL scenario), the **Minimum number of connected edge devices** (using KubeEdge) are to be checked; e) finally, the **Anomaly Detection and Alert Resolution Time** is defined as the time required to identify anomalies in the execution of the models and their correct resolution. This is to be tracked and measured using the deployed ML monitoring tools, to guarantee that the platform detects and addresses issues promptly.

Platform Performance

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|--|----------------|-----------------------|--|---|
| Data Ingestion speed | 1 MB/s | Between 1 and 10 MB/s | Time for data to be ingested and made available for processing or analysis after its arrival at the data ingestion interface | Tbd in the 1 st assessment cycle |
| Number of concurrent models in TRAINING stage (with GPU) | 0 | >=3 models | Track the number of pods (models) running in training namespaces | Tbd in the 2 nd assessment cycle |
| Number of concurrent models in TRAINING stage (without GPU) | 0 | >=6 models | Track the number of pods (models) running in training namespaces | Tbd in the 2 nd assessment cycle |
| Number of concurrent models in PRODUCTION (cloud) with GPU | 0 | >=3 models | Track the number of pods (models) running in production namespaces | Tbd in the 2 nd assessment cycle |

| | | | | |
|--|-----|-----------------------------|--|---|
| Number of concurrent models in PRODUCTION (cloud) without GPU | 0 | >=6 models | Track the number of pods (models) running in production namespaces | Tbd in the 2 nd assessment cycle |
| Minimum number of connected edge devices | 0 | >=3 devices | Identify the devices connected in the edge layer using KubeEdge | Tbd in the 2 nd assessment cycle |
| Anomaly detection and alert resolution time | n/a | <=5min (Grafana alert time) | Time to identify anomalies in model behaviour and resolve associated alerts using the ML Monitoring tools | Tbd in the 2 nd assessment cycle |

Table 28: Mobility AI-as-a-Service Platform Performance KPIs

Platform usability KPIs assess the acceptance and adoption of the MAIaaS platform by the EMERALDS developer community. This is to be done by monitoring: a) the **User engagement or adoption rate**, considering the number of registered users accessing the development and the deployment frameworks; b) the number of **Available Datasets for Training and/or Testing** stored in the MLOps repository; and c) the **Model repository utilization**, by counting the number of models offered and shared to be downloaded and deployed on proprietary frameworks. These numbers are directly related to the number of development partners and the number of EMERALDS to be deployed.

Platform Usage

| Description | Baseline Value | Target Value | Method of Measurement | Achieved Value |
|--|----------------|--------------|---|---|
| User engagement or adoption rate | 0 | >=5 users | Number of active users & frequency of usage | Tbd in the 1 st assessment cycle |
| Available datasets for training / testing | 0 | >=6 dataset | tbc | Tbd in the 2 nd assessment cycle |
| Model repository utilization | 0 | >=6 models | tbc | Tbd in the 2 nd assessment cycle |

Table 29: Mobility AI-as-a-Service Platform Usage KPIs

This evaluation will be carried out on the basis of the tools in place for the monitoring of the ML models' development, with a specific user interface designed to capture and track the KPIs. In detail:

- **Prometheus** monitoring system will be programmed to include **two different sets of metrics**: a) a first set will capture the status of the physical servers that support the MAIaaS platforms' cluster, so the efficiency usage of the dedicated resources (RAM, CPUs, GPU, storage, etc.) and the overall status of the cluster can be tracked. This will allow the detection and identification of whole performance bottlenecks and redistribute resources to improve general performance KPIs. The b) second set of metrics will be devoted to the Kubernetes cluster's distribution and the set of deployed tools and pods. This way, the different dedicated namespaces and running pods can be individually tracked, identifying so the resources, timings, network traffic etc. that each of them are consuming. This will provide a customized picture of what's currently happening within each clusters' area (development, experimentation, cloud/edge deployment, etc.) and its evolution in time. This contributes to

granular platform performance and platform utilization KPIs monitoring. All these metrics results will be used within **Grafana** to create the specific visualization and reporting dashboards for the selected KPIs. An example of what is currently being configured is shown in Figure 4-11.

- The **EvidentlyAI** monitoring tool is, in principle, intended for ML models' validation, helping the ML developers to evaluate, test, and monitor their models. The overall MAIaaS platform evaluation will consider this component to contribute to the measurement and show (Figure 4-12) the platform's usage KPIs, in terms of models' running and datasets consumed.

In the ongoing 1st integration cycle, the first EMERALDS models will be deployed in the MAIaaS platform. In the 1st assessment cycle that follows, the evaluation of these deployed models will be carried out, thus going beyond the integration and deployment testing of tools and frameworks presented here. The design and implementation of the dashboards to track and show platform's KPIs will be done in parallel to the first (preliminary) set of EMERALDS' deployments.

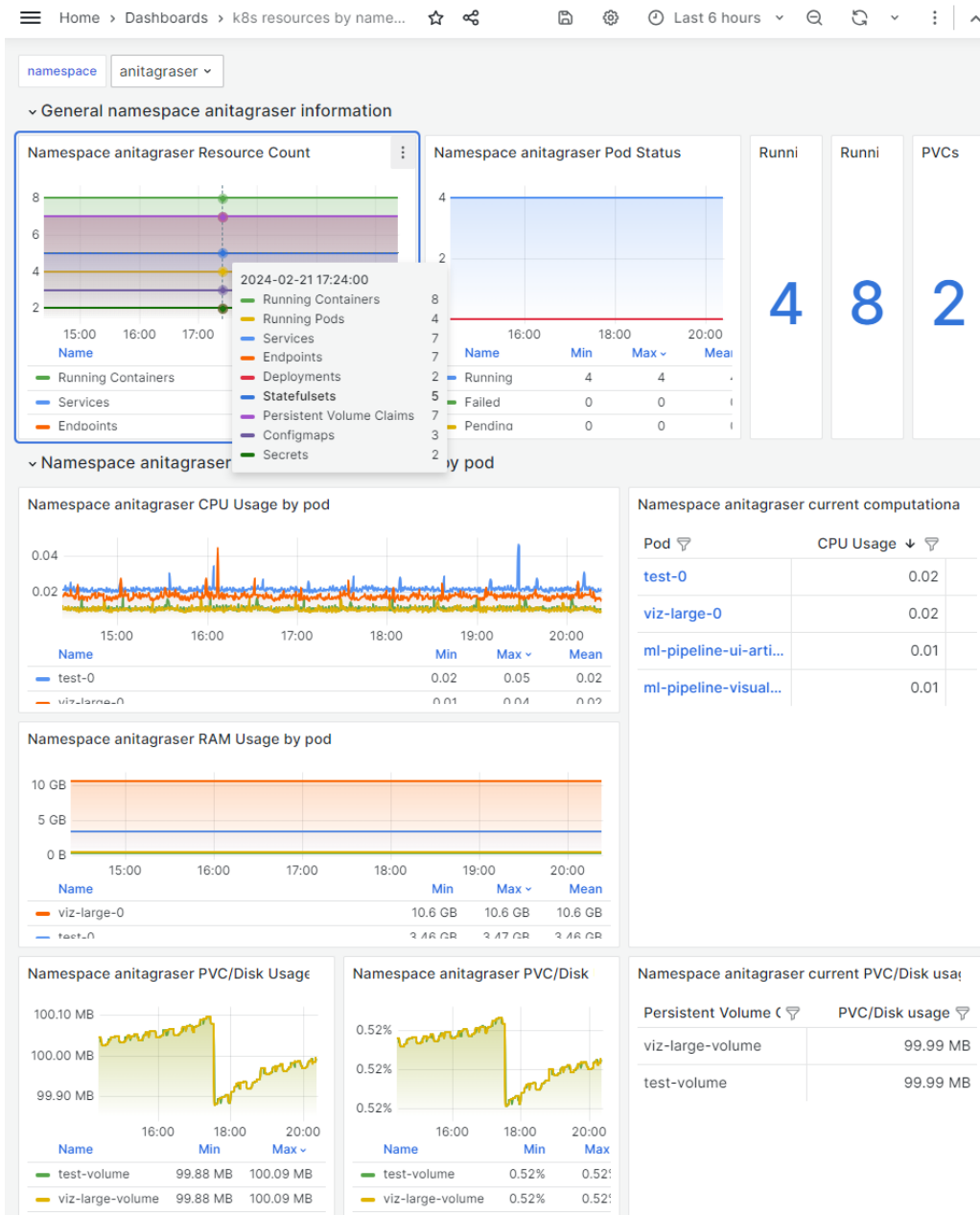


Figure 4-11: Example of a Kubernetes' cluster monitoring dashboard created with Grafana for EMERALDS MAIaaS KPIs.

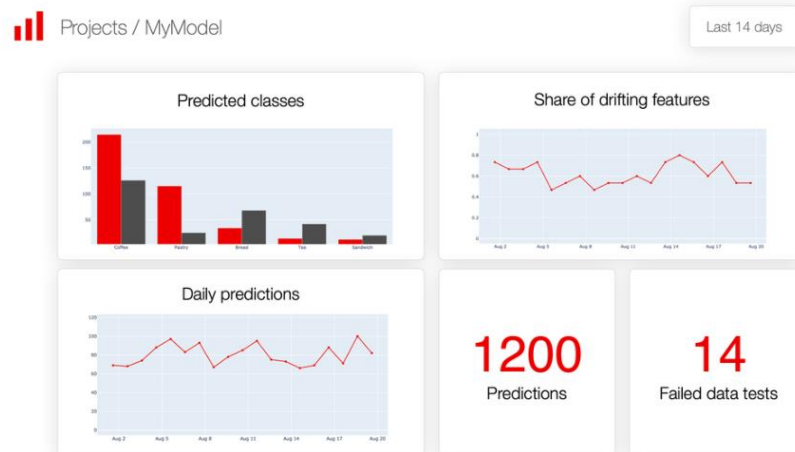


Figure 4-12: EvidentlyAI monitoring dashboard example (from evidentlyai.com).

4.5 Next Steps

The MAIaaS evolution is linked to the use cases and ML developers progress and needs, so several versions of this platform are planned to adapt its supported features.

The full cloud servers set with a first basic set of tools and frameworks was released by ATOS in **August 2023 (M8)**, intended to present the MAIaaS approach with its main features ready for ML development stages. This first version of the EMERALDS MAIaaS platform integrated (and offered) three main tools:

- Dockerized **Jupyter Notebooks** with **Kubeflow**, which provides the development environment.
- **MinIO** instance for storing and organizing datasets, as the core of the data management layer.
- **MLFlow** to support and track ML experiments.

Other additional tools and resources were also set up to prepare the MAIaaS environment. These include Keycloak⁷⁷ Identity Manager for access control management; the option to use GPUs for training when needed; and all the necessary credentials to directly connect the platform with the code and Docker repositories.

A new functional version, including some new features requested by the ML developers (WP3/WP4), was released in **January 2024 (M13)** to support the primary implementation cycle and the initial integration tests (Milestone 2). This update was improved to better assist developers with their daily model development tasks and covered: new tools for handling data, including **MobilityDB** for managing mobility-related data in a relational database and **DVC** for keeping track of different versions of data; **DataHub** was also added to make it easier to share, discover and document datasets to be further used in ML models' development; finally, some specific tools to model development and mobility were integrated as well, such as basic tools for data visualization (Hvplot⁷⁸ & Panel⁷⁹). During this stage, some KPIs related to the dataset management and data ingestion can be preliminarily assessed.

⁷⁷ <https://www.keycloak.org/>

⁷⁸ <https://hvplot.holoviz.org/>

⁷⁹ <https://panel.holoviz.org/>

Next steps related to the evolution of the EMERALDS MAIaaS platform are along the line of supporting the different pipelines that will make up the set of EMERALDS development and deployment. This task will mainly focus on data collection and orchestration support, assisting cities and WP3 data providers to collect and reuse datasets for training and testing ML models; and the deployment of the final ML models in both, cloud and edge layers, covering also the Federated Learning scenario.

These steps will culminate in the creation of the final version of the platform, which is internally planned for **mid-2024**. This version will tackle the EMERALDS deployment and production stage, making it easier to put models in a real-world setting. It will include:

- Full implementation of the **CI/CD pipelines** using **GitHub** Actions to automate the setup as much as possible.
- Two specific tools designed for the Kubeflow environment will be added as well: i) **Kserve**, to simplify the process of model deployment on a Kubernetes cluster for developers; and ii) **Kubeflow pipelines**, to automate and organize steps of training and deploying models automatically in production.
- In addition to traditional model deployment, the platform will support Federated Learning using the **Flower framework** and **KubeEdge** for managing devices at the edge.
- Monitoring tools like **Grafana**, **Prometheus** and **EvidentlyAI** will be deployed and configured for monitoring clusters and models' performance.
- **Demonstration** of the **use cases' pipelines** implementation is also considered to be carried out, as long as these are defined, and AI models are developed.

With this final version, the actual KPI validation will be performed, using the monitoring tools available in the cloud MAIaaS platform instance.

5 Conclusions and Next Steps

In this deliverable, we presented the current state of the developed mobility data analytics and learning tools and services (known as “emeralds”), as well as the Mobility AI-as-a-Service (MAIaaS) platform, as implemented and integrated within the 1st respective cycles and the planned next steps. In total, this deliverable covers:

| | Emeralds presented in D4.1 | | Project target |
|---------------------------|--|---|--------------------|
| | Number of emeralds | Emerald names | Number of emeralds |
| Analytics emeralds | 3 | <ul style="list-style-type: none"> • Probabilistic Trip Chaining • Trajectory Data Analysis • Extreme Scale Map Matching | 3 |
| Learning emeralds | 5 | <ul style="list-style-type: none"> • Dropoff/Destination Prediction • Traffic State / Flow Forecasting • Parking Garage Occupancy Prediction • Crowd Density Prediction • Active Learning & XAI for Crowd Prediction | 8 |
| MAIaaS Platform | 4 | <ul style="list-style-type: none"> • Data Management • ML Models Development Framework • ML Models Deployment Framework • Federated Learning Module | 4 |
| | Emeralds integrated in the MAIaaS Platform: | | |
| | 3 | <ul style="list-style-type: none"> • Parking Garage Occupancy Prediction • Crowd Density Prediction • Active Learning & XAI for Crowd Prediction | >= 5 |

Table 30: Summary of emeralds presented in D4.1.

In line with the objectives, expected results, and means of verification specified in the GA, the WP-level next steps are:

In the **1st integration and assessment cycle**, we will:

- Ensure and demonstrate interoperability of the developed WP4 emeralds through the design of cross-WP pipelines, including ingestion of data from WP3 emeralds and presentation of results in the EAD platforms. (O1 & O3)
- Link with use case testing, measurement, and validation. (O4)

In the **2nd iteration**, we will:

- Demonstrate the containerized EMERALDS toolset and service advancements in three selected use cases in the context of urban mobility deployed/showcased in the EAD platforms (ATOS & CARTO) addressing different niche user bases. (O1)
- Demonstrate the applicability and accuracy of the developed tools along the compute (edge/fog/cloud) continuum. (O3)
- Evaluate the privacy-awareness effect in the developed tools and services. (O3)
- Demonstrate the integration of data analytics and AI/ML tools in a Mobility AlaaS platform. (O3)
- Validate the Mobility AlaaS platform through real-world data (from use cases). (O3)

Annex

Annex 1: README Template

WPx Emerald: [Task] - [Service Name]

Description

This repository contains the first version of the service [Service Name], developed as part of [Task] of WPx within EMERALDS project. [Service Name] is a service for [brief description, including info about input/output]. This README provides essential information for deploying, testing, and [+++].

Table of Contents

[Include a table of contents in order to allow other people to quickly navigate especially long or detailed READMEs.]

Requirements

requirements.txt

Sample data input/output structures

[Describe the structure of sample input and output data for the service.]

Input/Output interfaces & interactions

[Describe the input and output interfaces and interactions; ensure compliance with the EMERALDS Ref.Arch.]

Deployment

[Along with the source code, include in this repository a Dockerfile containing the requirements to build the docker image and a docker-compose.yaml file defining the required configuration to execute each component in terms of ports, networks, and volumes.]

Usage - Executing program

[Provide instructions on how to use the project or run the code. Include examples and code snippets if applicable. Include references to sample data if applicable.]

Authors

[List the authors or contributors involved in the project.]

+++

Feel free to customize this template further to suit your specific service requirements.

Annex 2: Ethics Checklist and Questionnaire



Ethics Checklist and Questionnaire

A. PERSONAL DATA

1. Are **personal data** going to be processed for the completion of this deliverable?

NO

- If “yes”, do they refer only to individuals connected to project partners or to third parties as well?

2. Are “**special categories of personal data**” going to be processed for this deliverable? (whereby these include personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, and trade union membership, as well as, genetic data, biometric data, data concerning health or data concerning a natural person's sex life or sexual orientation)

NO

3. Has the **consent** of the individuals concerned been acquired prior to the processing of their personal data?

N/A

- If “yes”, is it based on the Project’s Informed Consent Form, either on the provided Template or on other attached herein Template?
- If “no” is it based on a different legal basis?

4. In the event of processing of personal data, is the processing:

N/A

- obviously “**Fair and lawful**”, meaning executed in a fair manner and following consent of the individuals concerned or based on another - acknowledged as adequate and proportionate as per above - legal basis?
 - Performed for a **specific (project-related) cause** only?
 - Executed on the basis of the principle of **proportionality and data minimisation** (meaning that only data that are **necessary** for the processing purposes are being processed and such deductive reasoning is documented)?
 - Based on **high-quality, updated and precise personal data**?
5. Are there any provisions for a storage limitation period of the personal data-in case of storage- after which they must be erased?

N/A

6. Are all **other lawful requirements** for the processing of the data (for example, **notification of the competent Data Protection Authority(s)** or undergoing a **DPIA procedure** and consulting with the competent DPA, if and where applicable) adhered to and on what legislative basis are such notifications justified as necessary or dismissed as unnecessary?

N/A

7. Have individuals been **made aware of their rights** on the processing of the personal data as per the GDPR and the relevant and executive national legislation (particularly the rights to access, rectify and delete the personal data and their right to lodge a complaint with the relevant Competent Authority) and if yes, by what demonstrable means (e.g. the informed consent form as per above or as per other Templates, attached herein?)

N/A

8. Even if anonymized or pseudonymized or aggregated data are referred to, does the dataset contain **location data** that could potentially (even via the combined use of other datasets) be **traced back to individuals**? If yes, what specific measures are taken to ensure this data (i) is anonymized or pseudonymized and (ii) cannot be used to track individuals without their consent? If no, what is the scientific methodology used to collect and gather said data?

We do not facilitate any deanonymization through the tools we develop. We alleviate the need to store data on this granularity.

9. In the context of risk assessment, prediction and forecasting, as foreseen in the scope of the EMERALDS project, during traffic, population movement monitoring or weather events, **is there any risk** that personal data could be inadvertently revealed in the event of an **emergency or unusual event**, because of the dataset usage, either on its own or combined with other openly available datasets, triggering identification or unwanted disclosure of PII? What measures are in place to protect - still identifiable if the dataset allows such extraction - **personal data** in these circumstances?

NO

10. For the use case of Trip Characteristics Inference as per the EMERALDS project scope, are there specific measures to ensure that **inferences made about trip characteristics** cannot be linked back to **specific individuals** or reveal **sensitive information** about their **habits** or **routines** i.e. by identifying specific individuals' absence or presence routines whether in the home or in a professional environment or in other premises?

We do not facilitate any deanonymization through the tools we develop. We alleviate the need to store data on this granularity.

11. Are there any potentially personal identifiable information (PII) in the datasets, **disclosable by combination with other datasets**, either open data or proprietary (e.g., E-tickets validation data)? If yes, how is PII adequately anonymized or pseudonymized or how other datasets that by combination may result in unwanted or illegal disclosures or identification before any processing takes place?

NO

B. DATA SECURITY

1. Have proportionate security measures been undertaken for protection of the data, taking into account project requirements and the nature of the data?

- If yes, brief description of such measures (including physical-world measures, if any)
- If yes, is there a data breach notification policy in place within your organization (including an Incident Response Plan to such a breach)?

YES. ATOS (as MAIaaS providers) store the datasets, supporting the platform to share the data among the AI developers. ATOS implement mechanisms to control the access to the platform and thus the stored data, identifying the users and keeping records of who is accessing, when and what. Each data provider (uploader) is responsible for their datasets, also deciding who can access and who cannot.

2. Given the **large-scale nature** of some datasets, are there specific measures in place to protect **included personal data** at scale at the data source or in the possession of data processors?

YES. The AI/Services developers (Data Processors) are responsible to process the personal data only according to the Use Cases data providers' instructions. Also, the Use Cases data providers (and those AI developers that produce datasets that might contain personal data) should indicate the MAIaaS provider (ATOS) of any special indication about how to proceed with their datasets (e.g., specific deletion and access rules).

3. Regardless of personal data, in the case of Multi-modal integrated traffic management as defined under the EMERALDS scope, are there specific measures in place to ensure **the availability and integrity of data spanning multiple modes of transport** from being disclosed in other manners than the ones intended and covered under an open data scheme?

N/A

4. Are there specific measures in place to secure **sensitive infrastructure data**, if present?

N/A

C. DATA TRANSFERS

1. Are personal data transfers beyond project partners going to take place for this deliverable?

NO

- If "yes", do these include transfers to third (non-EU) countries and if what policies apply?

2. Are personal data transfers to public authorities going to take place for this deliverable?

NO

3. Do any state authorities have direct or indirect access to personal data processed for this deliverable?

NO

3. Taking into account that the Project Coordinator is the “controller” of the processing and that all other project partners involved in this deliverable are “processors” within the same contexts, are there any other personal data processing roles further attributed to any third parties for this deliverable? And if any, are they conformed to the GDPR provisions?

NO

4. Given the geographical diversity of the datasets, are there measures in place to ensure compliance with specific personal data protection regulations **in different jurisdictions** i.e. at the place of the data source establishment as well as at the place of the establishment of a Data processor?

N/A

5. Are there additional protocols for data transfers involving **sensitive infrastructure data**, if present?

N/A

D. ETHICS AND RELATED ISSUES

1. Are personal data of children going to be processed for this deliverable (ie. “underage” signified e-tickets)?

NO

2. Is **profiling** of identifiable individuals in any way enabled or facilitated for this deliverable?

NO

3. Are **automated decisions** for identifiable individuals made or enabled on the basis this deliverable?

NO

4. Have partners for this deliverable taken into consideration system architectures of **privacy by design** and/or **privacy by default**, as appropriate?

YES

5. Have partners for this deliverable taken into consideration gender equality policies or is there an explicit reasoning that dismisses such risk as unsubstantiated or such need as irrelevant as per the methodology of work and production of the deliverable?

YES, if this question refers to the composition of the team. x out of y members working D4.1 are female.

6. Have partners for this deliverable taken into consideration means of protecting the confidentiality of the dataset if it is not signified as open data?

YES, all non-open datasets provided by WP5 use case partners are used only in compliance with the usage rules set by the use case partners.

7. Are there additional considerations around the collection and processing of **location data** and data **that could potentially be used to infer patterns about individuals' movements**?

NO

8. Have partners identified any **additional ethical issues** related to the processing of sensitive infrastructure data?

NO

9. Are shared economy (ie. "Uber" transfer services or "Lime" Scooters or other solution) or other shared mobility infrastructures used by the data sources? If yes, are there measures in place to ensure that the processing of **shared mobility data** respects privacy rights?

NO

10. In the context of Traffic Flow Data Analytics, are there specific considerations to ensure that the **analysis of traffic flow data** does not infringe on privacy rights or reveal sensitive information about individuals' movements or routines?

NO specific considerations necessary, since traffic flow data is already aggregated.

11. Is the Project taking into account the need for an all people-inclusive policy in the future within its overall goals and not only the "tech-savvy" (i.e. elderly people not familiar with some tech devices, poor people) and does it entail possible proposals for that?

YES

References

- 1 A. Graser, A. Jalali, J. Lampert, A. Weißenfeld, and K. Janowicz, "MobilityDL: A Review of Deep Learning From Trajectory Data," 2024, doi: 10.48550/ARXIV.2402.00732.
- 2 F. Schneider, W. Daamen, and S. Hoogendoorn, "Trip chaining of bicycle and car commuters: an empirical analysis of detours to secondary activities," *Transportmetrica A: Transport Science*, vol. 18, no. 3, pp. 855-878, 2022. DOI: 10.1080/23249935.2021.1901793.
- 3 M. Mokbel et al., "Towards Mobility Data Science (Vision Paper)," 2023, doi: 10.48550/ARXIV.2307.05717.
- 4 Department of Transport, "National Travel Survey Trip Chaining: 2002-2014," [Online]. Available: <https://assets.publishing.service.gov.uk/media/5a74f94ded915d3c7d52966c/nts-trip-chaining.pdf>
- 5 F. Primerano, M.A.P. Taylor, L. Pitakringkarn, et al., "Defining and understanding trip chaining behaviour," *Transportation*, vol. 35, pp. 55-72, 2008. DOI: 10.1007/s11116-007-9134-8.
- 6 N. McGukin and Y. Nakamoto, "Differences in Trip Chaining by Men and Women," in *Research in Women's Issues in Transportation*, Technical Papers, Transportation Research Board, Washington DC, vol. 2, pp. 49-56, 2004.
- 7 D. Bautista-Hernández, "Urban structure and its influence on trip chaining complexity in the Mexico City Metropolitan Area," *Urban, Planning and Transport Research*, vol. 8, no. 1, pp. 71-97, 2020. DOI: 10.1080
- 8 D. Esztergár-Kiss, "Trip Chaining Model with Classification and Optimization Parameters," *Sustainability*, vol. 12, no. 16, p. 6422, 2020. DOI: 10.3390/su12166422.
- 9 R. B. Noland and J. V. Thomas, "Multivariate Analysis of Trip-Chaining Behavior," *Environment and Planning B: Planning and Design*, vol. 34, no. 6, pp. 953-970, 2007. DOI: 10.1068/B32120.
- 10 A. Thomas and M. Ben-Akiva, "A theoretical and empirical model of trip chaining behavior," *Transportation Research Part B: Methodological*, vol. 13, no. 3, pp. 1979. DOI: 10.1016/0191-2615(79)90016-X
- 11 Z. Zhan, H.N. Koutsopoulos, and J. Zhao, "Individual mobility prediction using transit smart card data," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 19-34, 2018. DOI: 10.1016/j.trc.2018.01.022.
- 12 F. Schneider, W. Daamen, and S. Hoogendoorn, "Trip chaining of bicycle and car commuters: an empirical analysis of detours to secondary activities," *Transportmetrica A: Transport Science*, vol. 18, no. 3, pp. 855-878, 2022. DOI: 10.1080/23249935.2021.1901793
- 13 D.A. Hensher and A.J. Reyes, "Trip chaining as a barrier to the propensity to use public transport," *Transportation*, vol. 27, pp. 341-361, 2000. DOI: 10.1023/A:1005246916731.
- 14 E.B. Lunke and Øystein E., "Public Transport Use on Trip Chains: Exploring Various Mode Choice Determinants," *Findings*, 2023. DOI: 10.32866/001c.74112.
- 15 V. Costa, T. Fontes, J.L. Borges, and T.G. Dias, "Prediction of Journey Destination for Travelers of Urban Public Transport: A Comparison Model Study," in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 2018.
- 16 F. Zúñiga, J.C. Muñoz, and R. Giesen, "Estimation and prediction of dynamic matrix travel on a public transport corridor using historical data and real-time information," *Public Transport*, vol. 13, pp. 59-80, 2020.
- 17 J. Zhao, L. Zhang, J. Ye, and C. Xu, "MDLF: A Multi-View-Based Deep Learning Framework for Individual Trip Destination Prediction in Public Transportation Systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 13316-13329, 2022.
- 18 P. Zhang, H.N. Koutsopoulos, and Z. Ma, "DeepTrip: A Deep Learning Model for the Individual Next Trip Prediction With Arbitrary Prediction Times," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, pp. 5842-5855, 2023.
- 19 T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1, pp. 278-282, 1995.
- 20 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*, 2016.
- 21 G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Neural Information Processing Systems*, 2017.

-
- 22 A. Graser, "MovingPandas: efficient structures for movement data in Python," *GI_Forum*, vol. 7, no. 1, pp. 54–68, 2019, doi: 10.1553/giscience2019_01_s54.
- 23 L. Pappalardo, F. Simini, G. Barlacchi, and R. Pellungrini, "scikit-mobility: A Python library for the analysis, generation, and risk assessment of mobility data" *J. Stat. Softw.*, vol. 103, no. 1, pp. 1–38, 2022, doi: 10.18637/jss.v103.i04.
- 24 H. Martin, Y. Hong, N. Wiedemann, D. Bucher, and M. Raubal, "Trackintel: An open-source Python library for human mobility analysis," *Comput Environ Urban Syst*, vol. 101, p. 101938, 2023, doi: 10.1016/j.compenvurbsys.2023.101938.
- 25 R. Joo, M. E. Boone, T. A. Clay, S. C. Patrick, S. Clusella-Trullas, and M. Basille, "Navigating through the R packages for movement," *J Anim Ecol*, vol. 89, no. 1, pp. 248–267, 2020, doi: 10.1111/1365-2656.13116.
- 26 A. Graser, "Tools for the analysis of movement data," Accessed: Jan. 23, 2024, <https://github.com/anitagraser/movement-analysis-tools>
- 27 A. Graser, "The State of Trajectory Visualization in Notebook Environments," *GI_Forum*, vol. 1, pp. 73–91, 2023, doi: 10.1553/giscience2022_02_s73.
- 28 G. Andrienko, N. Andrienko, W. Chen, R. Maciejewski, and Y. Zhao, "Visual analytics of mobility and transportation: state of the art and further research directions," *IEEE T-ITS*, vol. 18, no. 8, pp. 2232–2249, 2017, doi: 10.1109/TITS.2017.2683539.
- 29 A. Graser, M. Straub, and M. Dragaschnig, "Towards an open source analysis toolbox for street network comparison," *Transactions in GIS*, vol. 18, no. 4, pp. 510–526, 2014, doi: 10.1111/tgis.12061.
- 30 G. Cosentino and F. Pennica, "QGIS geoprocessing model to simplify first level seismic microzonation analysis," *PeerJ Preprints*, 2016, doi: 10.7287/peerj.preprints.2250v2.
- 31 L. Raimondi, G. Pepe, M. Firpo, D. Calcaterra, and A. Cevasco, "An open-source and QGIS-integrated physically based model for spatial prediction of rainfall-induced shallow landslides (SPRIn-SL)," *Environ. Model. Softw.*, vol. 160, p. 105587, 2023, doi: 10.1016/j.envsoft.2022.105587.
- 32 N. Andrienko and G. Andrienko, "Spatiotemporal visual analytics: a vision for 2020s," *JOSIS*, no. 20, pp. 87–95, 2020, doi: 10.5311/JOSIS.2020.20.661.
- 33 OGC, "OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture." 2011. <https://www.opengeospatial.org/standards/sfa>
- 34 OGC, "OGC® Moving Features Encoding Part I: XML Core." 2015. <https://www.ogc.org/standards/movingfeatures>
- 35 A. Graser, E. Zimányi, and K. C. Bommakanti, "From Simple Features to Moving Features and Beyond?," arXiv:2006.16900 [cs], 2020.
- 36 N. Zhao, L. Yu, H. Zhao, J. Guo, and H. Wen, "Analysis of Traffic Flow Characteristics on Ring Road Expressways in Beijing: Using Floating Car Data and Remote Traffic Microwave Sensor Data," *Transportation Research Record: Journal of the Transportation Research Board*, pp. 178-185, 2009.
- 37 O. Altıntasi, H. Tuydes-Yaman, and K. Tuncay, "Monitoring Urban Traffic from Floating Car Data (FCD): Using Speed or a LOS-Based State Measure," in *Directions of Development of Transport Networks and Traffic Engineering. TSTP 2018*.
- 38 A.M. Nagy and V. Simon, "Survey on traffic prediction in smart cities," *Pervasive and Mobile Computing*, vol. 50, pp. 148-163, 2018.
- 39 M. Shaygan et al., "Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities," *Transportation Research Part C: Emerging Technologies*, vol. 145, p. 103921, 2022. DOI: 10.1016/j.trc.2022.103921.
- 40 J. Gu et al., "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354-377, 2018.
- 41 Y. Yu et al., "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235-1270, 2019.
- 42 X. Shi et al., "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.
- 43 Y. Wang et al., "Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

-
- 44 G. Jin et al., "Spatiotemporal Graph Neural Networks for Predictive Learning in Urban Computing: A Survey," 2023. DOI: 10.48550/ARXIV.2303.14483.
- 45 T.N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," 2016. DOI: 10.48550/ARXIV.1609.02907.
- 46 B. Yu, H. Yin, and Z. Zhu, "Spatiotemporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," in Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, pp. 3634–3640, 2018. DOI: 10.24963/ijcai.2018/505.
- 47 D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.
- 48 A. Vaswani et al., "Attention is All You Need," in Advances in Neural Information Processing Systems, vol. 30, 2017.
- 49 P. Wu et al., "A combined deep learning method with attention-based LSTM model for short-term traffic speed forecasting," Journal of Advanced Transportation, vol. 2020, pp. 1-15, 2020.
- 50 S. Guo et al., "Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting," in Proceedings of the AAAI Conference on Artificial Intelligence, pp. 922-929, 2019.
- 51 C. Zheng et al., "GMAN: A Graph Multi-Attention Network for Traffic Prediction," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 01, pp. 1234-1241, 2020. DOI: 10.1609/aaai.v34i01.5477.
- 52 F. Li et al., "Dynamic Graph Convolutional Recurrent Network for Traffic Prediction: Benchmark and Solution," ACM Transactions on Knowledge Discovery from Data, vol. 17, no. 1, pp. 1-21, 2023. DOI: 10.1145/3532611.
- 53 Z. Shao et al., "Decoupled Dynamic Spatial-Temporal Graph Neural Network for Traffic Forecasting," Proc. VLDB Endow., vol. 15, no. 11, pp. 2733-2746, 2022.
- 54 K. Bandara, C. Bergmeir, and H. Hewamalage, "LSTM-MSNet: Leveraging forecasts on sets of related timeseries with multiple seasonal patterns," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 4, pp. 1586-1599, 2020.
- 55 K. Bandara, R. Hyndman, and C. Bergmeir, "MSTL: A Seasonal-Trend Decomposition Algorithm for Timeseries with Multiple Seasonal Patterns," International Journal of Operational Research, vol. 1, no. 1, p. 1, 2022. DOI: 10.1504/IJOR.2022.10048281.
- 56 D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533-536, 1986.
- 57 K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint, arXiv:1406.1078, 2014.
- 58 S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.
- 59 Z. Zhao, Y. Zhang, and Y. Zhang, "A comparative study of parking occupancy prediction methods considering parking type and parking scale," Journal of Advanced Transportation, 2020, pp. 1-12.
- 60 W. Ye, H. Kuang, X. Lai, and J. Li, "A Multi-View Approach for Regional Parking Occupancy Prediction with Attention Mechanisms," Mathematics, vol. 11, no. 21, p. 4510, 2023.
- 61 J. Li, H. Qu, and L. You, "An Integrated Approach for the Near Real-Time Parking Occupancy Prediction," IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 4, pp. 3769-3778, 2022.
- 62 S. Gutmann, C. Maget, M. Spangler, and K. Bogenberger, "Truck parking occupancy prediction: Xgboost-LSTM model fusion," Frontiers in Future Transportation, vol. 2, article 693708, 2021.
- 63 Z. Niu, X. Hu, M. Fatmi, S. Qi, S. Wang, H. Yang, and S. An, "Parking occupancy prediction under COVID-19 anti-pandemic policies: A model based on a policy-aware temporal convolutional network," Transportation Research Part A: Policy and Practice, vol. 176, article 103832, 2023.
- 64 L. Zheng, X. Xiao, B. Sun, D. Mei, and B. Peng, "Short-term parking demand prediction method based on variable prediction interval," IEEE Access, vol. 8, pp. 58594-58602, 2020.
- 65 R. Jiang, Z. Cai, Z. Wang, C. Yang, Z. Fan, Q. Chen, et al., "DeepCrowd: A deep model for large-scale citywide crowd density and flow prediction," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 1, pp. 276-290, 2021.

-
- 66 X. Fu, G. Yu, and Z. Liu, "Spatial-temporal convolutional model for urban crowd density prediction based on mobile-phone signaling data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14661-14673, 2021.
- 67 H. Minoura, R. Yonetani, M. Nishimura, and Y. Ushiku, "Crowd density forecasting by modeling patch-based dynamics," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 287-294, 2020.
- 68 R. Zhao, Y. Wang, Q. Li, M. Li, D. Dong, and C. Li, "Crowd Density Prediction Model Based on Image Processing and Support Vector Regression," in *Proceedings of the 2020 International Conference on Computing, Networks and Internet of Things*, April 2020, pp. 94-98.
- 69 L.W. Chen and C.C. Weng, "Time-Dependent Visiting Trip Planning with Crowd Density Prediction Based on Internet of Things Localization," *IEEE Transactions on Mobile Computing*, 2022.
- 70 X. Ding, F. He, Z. Lin, Y. Wang, H. Guo, and Y. Huang, "Crowd density estimation using fusion of multi-layer features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4776-4787, 2020.
- 71 V. Sundaram, A.K. Tripathy, R. Deshmukh, and A. Pawar, "A Crowd Density Estimation Approach Using GPS Mobility for Its Dynamics and Predictions," in *2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA)*, April 2020, pp. 67-72.
- 72 Y. Wang, X. Hou, and L.P. Chau, "Dense point prediction: A simple baseline for crowd counting and localization," in *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, July 2021, pp. 1-6.
- 73 M.R. Bhuiyan, J. Abdullah, N. Hashim, F. Al Farid, M.A. Haque, J. Uddin, et al., "A deep crowd density classification model for Hajj pilgrimage using fully convolutional neural network," *PeerJ Computer Science*, vol. 8, p. e895, 2022.
- 74 F. Bodria, F. Giannotti, R. Guidotti, F. Naretto, D. Pedreschi, and S. Rinzivillo, "Benchmarking and survey of explanation methods for black box models," *Data Mining and Knowledge Discovery*, pp. 1-60, 2023.
- 75 A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138-52160, 2018.
- 76 F. Hohman, M. Kahng, R. Pienta, and D.H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 8, pp. 2674-2693, 2018.
- 77 Z. Wang, W. Song, L. Liu, F. Zhang, J. Xue, Y. Ye, et al., "Representation learning with deconvolution for multivariate timeseries classification and visualization," *arXiv preprint arXiv:1610.07258*, 2016.
- 78 A.H. Gee, D. Garcia-Olano, J. Ghosh, and D. Paydarfar, "Explaining deep classification of time-series data with learned prototypes," in *CEUR workshop proceedings*, vol. 2429, August 2019, p. 15.
- 79 S.D. Goodfellow, A. Goodwin, R. Greer, P.C. Laussen, M. Mazwi, and D. Eytan, "Towards understanding ECG rhythm classification using convolutional neural networks and attention mappings," in *Machine Learning for Healthcare Conference*, November 2018, pp. 83-101, PMLR.
- 80 R. Assaf and A. Schumann, "Explainable deep neural networks for multivariate timeseries predictions," in *IJCAI*, August 2019, pp. 6488-6490.
- 81 J. Wang, Z. Wang, J. Li, and J. Wu, "Multilevel wavelet decomposition network for interpretable timeseries analysis," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, July 2018, pp. 2437-2446.
- 82 E.Y. Hsu, C.L. Liu, and V.S. Tseng, "Multivariate timeseries early classification with interpretability using deep learning and attention mechanism," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, March 2019, pp. 541-553, Springer International Publishing.
- 83 U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D.A. Keim, "Towards a rigorous evaluation of XAI methods on timeseries," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, October 2019, pp. 4197-4201, IEEE.
- 84 A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*, July 2017, pp. 3145-3153, PMLR.
- 85 M.T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?" Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2016, pp. 1135-1144.

-
- 86 S.M. Lundberg and S.I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- 87 J. Hao, Q. Shi, Y. Ye and W. Zeng, "TimeTuner: Diagnosing Time Representations for Time-Series Forecasting with Counterfactual Explanations," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 1183-1193, Jan. 2024, doi: 10.1109/TVCG.2023.3327389.
- 88 A. Jose, J.P.A. de Mendonça, E. Devijver, N. Jakse, V. Monbet, and R. Poloni, "Regression tree-based active learning," *Data Mining and Knowledge Discovery*, pp. 1-41, 2023.
- 89 H. Gweon and H. Yu, "A nearest neighbor-based active learning method and its application to timeseries classification," *Pattern Recognition Letters*, vol. 146, pp. 230-236, 2021.
- 90 Z. Wang, B. Zhao, H. Guo, L. Tang, and Y. Peng, "Deep ensemble learning model for short-term load forecasting within active learning framework," *Energies*, vol. 12, no. 20, p. 3809, 2019.
- 91 T. Wu and J. Ortiz, "Rlad: Timeseries anomaly detection through reinforcement learning and active learning," arXiv preprint arXiv:2104.00543, 2021.
- 92 R. Yu, Y. Wang, and W. Wang, "AMAD: Active learning-based multivariate timeseries anomaly detection for large-scale IT systems," *Computers & Security*, vol. 137, 103603, 2024.
- 93 J. Kim, D. Kim and J. Lee, "Design and Implementation of Kubernetes enabled Federated Learning Platform," *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Korea, Republic of, 2021, pp. 410-412, doi: 10.1109/ICTC52510.2021.9620986.
- 94 D.J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, and N.D. Lane, "Flower: A Friendly Federated Learning Research Framework," 2020.
- 95 P. K. Quan, M. Kundroo and T. Kim, "Experimental Evaluation and Analysis of Federated Learning in Edge Computing Environments," in *IEEE Access*, vol. 11, pp. 33628-33639, 2023, doi: 10.1109/ACCESS.2023.3262945.