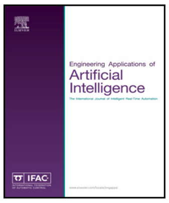
Engineering Applications of Artificial Intelligence 123 (2023) 106222

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



MLOps in freight rail operations

Juan Pineda-Jaramillo [*], Francesco Viti

Department of Engineering, University of Luxembourg, Luxembourg

ARTICLE INFO

Keywords:
Freight rail operations
Machine learning
MLOps
CI/CD/CT
Apache airflow
ML deployment

ABSTRACT

Railways are essential for freight transport due to their operational reliability advantages, but maintaining this advantage requires optimised railway infrastructure. Previous research has developed models to predict freight rail disruptions/disturbances and their associated delay times, in order to better understand the impact of multiple factors on them. However, because these models are built on static datasets, extracting real value from a model in a production environment remains difficult.

This paper presents a methodology that demonstrates the potential of MLOps in automating the entire workflow, from data extraction to model deployment for real-time delay predictions in freight rail operations, including good practices of Continuous-Integration, Continuous-Delivery, and Continuous-Training, as well as a tool list for each process.

Our research advances the field of railway operations by developing an entire MLOps workflow using data from the freight rail operations of the Luxembourgish National Freight Railway Company over a seventeen-month period. Furthermore, we employed a LightGBM model that had previously performed well in another study. This workflow can be automatically triggered to develop the processes and thus maintain an ML model capable of predicting delay times for CFL Multimodal operations in real-time.

Our findings demonstrate that MLOps have the potential to automate the entire process, opening up new avenues for future research in this field. Although the methodology presented is intended to optimise freight rail operations for a specific company, it can be easily transferable to other railway companies or other transportation industries, such as aviation, shipping, and trucking.

1. Introduction

Railways play an important role in freight operations because of the numerous advantages they have over their competitors in terms of operational reliability, operation and maintenance costs, safety, and low emissions (Cacchiani et al., 2010; Dong et al., 2022). As a result, public agencies and governments at the national and international levels promote the modal shift of freight transport from multiple alternatives to railways. Thus, to remain the dominant mode of freight transport, each aspect of railway operations must be optimised, including the use of railway infrastructure.

Delays are a critical factor to consider when optimising the use of railway infrastructure. They can occur due to the variability in the train's preparation times before departure or variability in the train's operation during its journey (Goverde and Hansen, 2013). Predicting these delays is critical in optimising freight rail operations, because when a disturbance or disruption occurs in railway operations, the operator must evaluate its impact on the overall operation of the railway network and modify the timetable to reduce the chain of delays that such disturbance/disruption can generate (Berger et al., 2011; Bešinović et al., 2016; Goverde, 2010; Goverde et al., 2016).

Previous research has predominantly focused on examining the direct relationships between multiple features and disturbances or disruptions in freight train operation, using conventional statistical methods or Machine Learning (ML) models (Cacchiani et al., 2014; Corman and Kecman, 2018; Ghofrani et al., 2018; Lessan et al., 2018; Marković et al., 2015; Milinković et al., 2013; Nair et al., 2019; Pineda-Jaramillo and Viti, 2023; Wen et al., 2019; Yaghini et al., 2013; Yuan et al., 2002), with the goal of determining the best approach in terms of precision in predicting the occurrence of these phenomena, or accuracy in predicting delay times that may occur in railway operation (Corman and Kecman, 2018; Dollevoet et al., 2015; Huang et al., 2020; Lessan et al., 2018; Marković et al., 2015; Milinković et al., 2013; Nair et al., 2019; Wang et al., 2022; Wen et al., 2019; Yaghini et al., 2013; Yuan et al., 2002). From a purely practical standpoint, this approach does not allow a railway operator to know with certainty, and in real time, the delay that one train will experience due to disturbances or disruptions in railway operation, necessitating the development of a tool that allows to know in real time the operating delay time of freight trains,

* Correspondence to: Department of Engineering, University of Luxembourg, 6 Avenue de la Fonte, Maison du Nombre, office 0435010, 4364 Esch-sur-Alzette, Luxembourg.
E-mail address: juan.pineda@uni.lu (J. Pineda-Jaramillo).

https://doi.org/10.1016/j.engappai.2023.106222
Received 10 October 2022; Received in revised form 13 March 2023; Accepted 25 March 2023
Available online xxxx

allowing decision making to reduce the impact on the chain of delays in the entire system's operation.

With the aim of developing a tool that allows a railway operator to know in real time the delay that one freight train will have due to disturbances or disruptions in the rail operations, this study aims to build a ML system by implementing the concepts of Machine Learning Operations (MLOps) in order to automate all operations from the ingestion of new data, processing, model development, performance monitoring, and using the model as a prediction service, allowing railway operators to make decisions to reduce the impact of delays in the entire operation of the system. MLOps can be considered as a methodology or a set of best practices that combine software engineering principles and machine learning workflows to build, test, deploy, and maintain machine learning models in a production environment.

This study addresses the process of automating ML pipelines for predicting arrival delay times resulting from disruptions and disturbances in freight rail operations using a supporting dataset of 10,265 trips between control stations in operations that occurred between November 2019 and March 2021 between the central terminal of the freight rail network of the National Railway Company of Luxembourg (CFL multimodal) in Bettembourg (Luxembourg) and other nine stations in France, Belgium, Germany, Poland, and Italy, with multiple possible routes to reach all final destinations. Even though this study includes data from a single freight rail company, the methodology presented in this study can be easily replicated for data from other freight rail companies. Furthermore, because this study builds on a previous study (Pineda-Jaramillo et al., 2022b) in which several ML models were trained and tested using the same data, and a LightGBM model was optimised with satisfactory metrics, processes such as data preparation, model training, optimisation, and further validation of ML models will not be discussed in this study.

The novelty of this study lies in the development of an MLOps workflow for predicting delays in railway operations using machine learning models. The study contributes to the field of railway operations by demonstrating the effectiveness of MLOps in automating the processes of data ingestion, model development, and model deployment for real-time prediction of delays. The use of MLOps in railway operations has not been extensively explored, making this study a unique contribution to the field.

This paper is structured in the following manner. Section 2 provides an overview of previous research on the use of machine learning to forecast delays in railway operations, as well as the integration of MLOps for deploying models that enable real-time predictions. Section 3 outlines the methodology employed to establish an MLOps workflow that automates data ingestion, model development, selection of the optimal model for prediction purposes, and ongoing monitoring of model performance. The outcomes and analysis of the study are presented in Section 4. The paper concludes with a summary of the key findings and recommendations for future research, which are presented in Section 5.

2. Literature review

Several studies have employed data-driven approaches for predicting rail operation delays resulting from disturbances or disruptions. These approaches typically use numeric features and regression/classification models to make predictions. For example, Kecman and Goverde (2015) developed a decision tree model and a robust linear regression model to predict the dwell and running times of trains. Li et al. (2016) used linear regression and K-Nearest Neighbour algorithms to anticipate the duration of station stops. Barbour et al. (2018) utilised support vector regression to estimate the arrival times of freight trains based on train, network, and traffic congestion characteristics. Similarly, Wen et al. (2017) and Huang et al. (2019) used data-driven techniques to investigate the features related to rail service interruptions and their resulting delays in High-Speed Railway Systems. Luo et al. (2022) and Van der Meer et al. (2010) found that some research results reveal a strong connection between train delays and dwell times, while others reveal a weaker relationship between running times and departure delays. More recently, Minbashi et al. (2023) proposed a machine learning-assisted framework to improve freight rail operations predictability by focusing on yard departures and arrivals and employing a random forest algorithm. Other data-driven models such as decision trees, support vector machines, random forests, and artificial neural networks have also been utilised to predict rail operation delays and investigate the factors associated with these events.

Machine Learning systems, defined as software systems that include ML concepts as one of their components, are known for considering multiple factors that were previously unconsidered in the field of software development. Unlike standard software, ML systems are complexly intertwined with data on top of standard code, making these systems more difficult to maintain in the long run (Ruf et al., 2021). Furthermore, because several professionals (e.g., analysts, researchers, data scientists, ML engineers, data engineers, cloud engineers, etc.) must collaborate to handle all the required aspects, from data collection from multiple sources to monitoring the performance of models when they are used in production, data management has become the new focus in Machine Learning systems.

A commonly used methodology in machine learning is the traditional software development approach, which often follows a linear, sequential process from requirements gathering, to design, to coding, and finally to testing and deployment. However, this approach can lead to challenges in the integration and deployment of machine learning models, as it may not account for the unique characteristics and challenges of these models (Géron, 2019). Additionally, the traditional software development approach may not prioritise the ongoing maintenance and monitoring of machine learning models, which is crucial for ensuring their continued performance and accuracy over time.

As a response to the numerous challenges that arise in the field of ML systems, such as a continuous flow and significant increase in data, or possible variations in the behaviour of new data that cause model performance to degrade over time (Garg et al., 2021; Lwakatare et al., 2020), the concept of MLOps emerges, which promotes the automation of all the steps involved in building an ML system from development to delivery. MLOps is a discipline that combines Machine Learning models, Development Operations (DevOps), and Data Engineering in order to deploy ML systems in a dependable and efficient manner.

MLOps is a novel and emerging methodology that offers several advantages over traditional machine learning methodologies. One of the main advantages of MLOps is its focus on creating a collaborative and continuous process for developing, deploying, and monitoring ML models in production environments. This ensures that the models remain accurate and reliable over time, which is especially important for safety-critical applications in the transportation industry. MLOps also allows for the automation of many of the tasks involved in developing ML models, such as data pre-processing, model training, and model evaluation, which can reduce the time and effort required to deploy ML models in production. Additionally, MLOps provides a framework for version control, testing, and documentation, which can improve the transparency and reproducibility of ML models. Several academic studies have shown the advantages of using MLOps in various applications, such as healthcare (Granlund et al., 2021), finance (Xu, 2022), and energy (Gürses-tran, 2022; Subramanya et al., 2022). Overall, MLOps is a promising methodology that can improve the efficiency, reliability, and transparency of ML models in various fields.

Among the most important concepts that allow the optimal development of DevOps, and in turn the development of MLOps, are Continuous Integration, Continuous Training, and Continuous Delivery, which are commonly known as CI/CD/CT (Garg et al., 2021; Kazmierczak and Schut, 2021; Ruf et al., 2021). Continuous Integration (CI) aids in time management by allowing a company to have short and frequent software release cycles to improve software quality and overall team productivity. Continuous Delivery (CD) aids in automatically deploying software in a production environment, while Continuous Training (CT) entails orchestrating and automating the execution of model retraining with new data with the goal of improving their performance over time.

Despite the various approaches utilised to develop models for predicting train disruptions, a significant gap in the research has been identified. While many studies have focused on discovering useful insights and conducting thorough analyses, we could not find studies that successfully deployed a model that can provide real-time predictions to railway companies. This information is crucial for enabling these companies to make informed decisions and mitigate the impact of these delays on the overall operation of the railway system.

Within this context, the following are the study's objectives and main contributions:

- Explore various tools to create an MLOps workflow capable of automating the processes of data ingestion, model development, use of the best model as a prediction service, and constant model performance monitoring.
- Analyse the benefits of creating this automated MLOps workflow by implementing CI/CD/CT concepts to automate the entire MLOps pipeline, with the goal of optimising freight rail operations for a specific company but easily replicable to other freight rail companies.

3. Methodology

This section describes the process to develop an automated MLOps workflow that allows automating all the processes required to predict delay times in freight rail operations, including all steps from data ingestion to use an ML model as a prediction service, where good practices of CI/CD/CT are incorporated. A diagram outlining these steps is shown in Fig. 1, where the steps that can be included in an MLOps workflow are presented in a general way, which can be modified by a company according to its specific development needs.

It is important to consider that each of the phases presented, including the tasks performed within each phase, can be performed through the implementation of multiple tools, depending on the way each company works and its technological development. This study does not intend to describe the enormous number of tools available to date, but rather it intends to describe how to perform the processes in a general way. For an extensive description of existing tools, it is recommended to read the study carried out by Ruf et al. (2021). On the other hand, in order to focus this study on the automation process of MLOps workflows in freight rail operations, this study does not intend to explain issues related to the exploratory data analysis, the selection of the ML models implemented, nor the analysis of the impact of the features with delay times in freight rail operations, since theses aspects were discussed in a previous study (Pineda-Jaramillo et al., 2022b), from which the origin of this new study arises.

3.1. Traditional steps in the data science workflow for developing ML models

In any project that includes the development of ML models, once the objective is defined (e.g., prediction of the arrival delay time of a freight train in the next station), and the success criteria is established (e.g., an accuracy greater than a certain percentage), the process of creating a ML model for using it as a prediction service consists in the following steps, which are usually done manually or using an automatic pipeline (Bollegala, 2017; Kuflik et al., 2017; Mesa-Arango et al., 2023; Pineda-Jaramillo et al., 2022a):

- *Data extraction and analysis:* Selection of relevant data from available data to develop the prediction model, and the development of an exploratory data analysis to deeply comprehend the data.
- *Data preparation:* Data is prepared for training the predictive models. Multiple tasks are included in this step, such as data cleaning, data transformation according to data types, creation of new features using domain knowledge, dataset split in training, validation and test sets, etc.
- *Model training:* Several algorithms are built with the prepared data in order to train multiple ML models using the training set, which vary according to the prediction needs (e.g., to predict the arrival delay time of a freight train, which is a numerical value, a supervised ML model would be selected, using regression algorithms). Additionally, the models are optimised by tuning the internal hyperparameters of each model, with the aim of finding the best model that will be used as a prediction service.
- *Model evaluation:* The model is evaluated through the analysis of certain metrics to understand and measure the predictive performance of the model.
- *Model validation:* It is validated that the best trained and optimised model is suitable for deployment, confirming that its predictive power is better than a certain baseline.
- *Prediction service:* The validated model is deployed in an environment that allows it to be used by the customer (e.g., a freight railway company) to make predictions.
- *Performance monitoring:* The predictive performance of the model is monitored so that in case the model degrades, a new iteration of the ML process is performed.

3.2. Implementation of CI/CD/CT practices in the automated development of MLOps

Many of the previous studies carried out on freight rail operations elaborate the processes described above manually, a process that is sufficient when the objective is only to analyse the impact of certain features on the prediction of disturbances, disruptions, or delay times in freight rail operations resulting from these. However, these models rarely adapt both to changes in the dynamics of the operations, and to changes in the data of the same operations (Talby, 2019). To address these challenges, it is necessary to perform aspects such as (a) a constant monitoring of the quality of the model in production to detect possible degradations in model performance, (b) a continuous retraining of the model with more recent data to capture possible changing patterns in the new data, and (c) a continuous experimentation with new methods, models and features obtained from additional data sources, aiming to optimise the performance of the predictive model. To meet these multiple challenges of manual processes, the CI/CD/CT practices in MLOps are implemented.

The objective of an automated development of MLOps workflow implementing good CI/CD/CT practices, considers the following concepts that can be seen in Fig. 1.

- *Rapid experiments:* The aforementioned steps are orchestrated using tools such as Apache Airflow, Apache Beam or Kubeflow (Apache Software Foundation, 2016, 2015; Google, 2018; Kotliar et al., 2019; Li et al., 2018), where the transition between the various steps, from data collection to model deployment and monitoring, is automated, leading to quick iteration of experiments and better readability to bring the pipeline into production.
- *CT of the model in production:* The training of the model is automatically performed in production using new data based on live pipeline triggers.
- *Modularised code for components and pipelines:* It is important that the multiple components of the MLOps workflow are reusable and potentially shareable between different ML workflows. Furthermore, it is important that the components are containerised in order that each component has its own runtime and to have different programming languages and libraries working on the environment. This modularisation can be easily done using tools

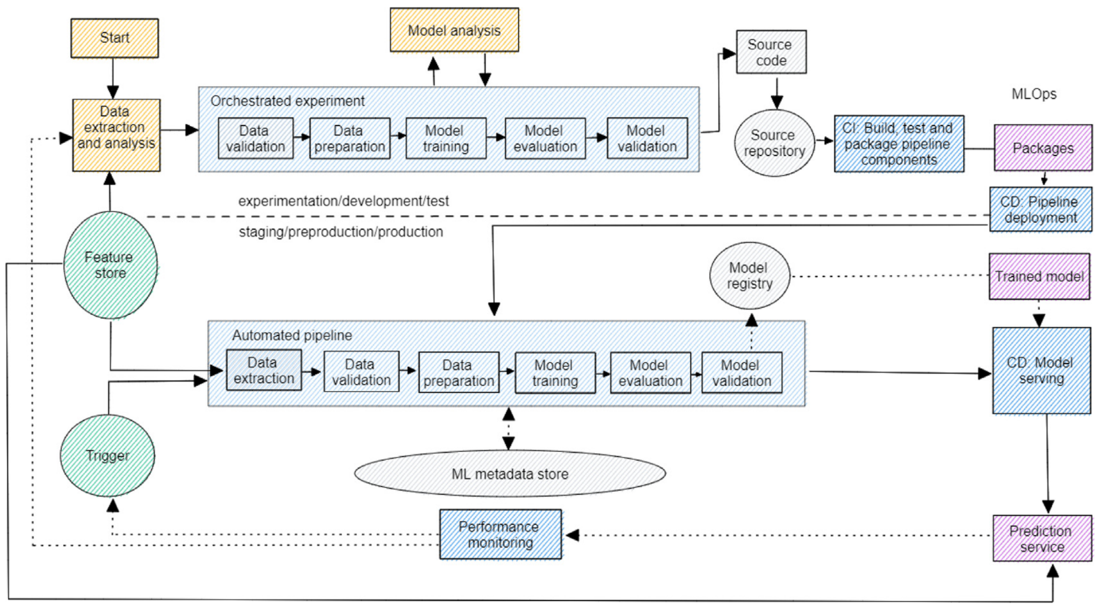


Fig. 1. Flowchart of the MLOps workflow that allows to automate all the processes required to predict delay times in freight rail operations.
Source: Adapted from Kazmierczak and Schut (2021).

such as Docker and Kubernetes (Google, 2014; Merkel, 2014), which provide the ability to package and run applications in small isolated environments, avoiding dependency problems between different versions of programming languages, libraries, etc.

- *CD of the models:* The MLOps workflow must be continuously automated to offer prediction services using new models which are trained using new data.
- *Pipeline deployment:* It is implemented a complete and recurring training process to use the trained model as a prediction service.
- *Constant data validation:* This phase is essential before training the model to decide if it is necessary to retrain the model or to stop the execution of the pipeline. To achieve this, the MLOps workflow must validate if there are data schema biases or data value biases, and if this occurs, it will be necessary to validate, fix and trigger a new model training to capture these changes.
- *Constant validation of the model:* This step is performed once the model is trained using new data, where the model must be evaluated before using it in production as a prediction service. Besides, the metrics of the new model must be evaluated and compared with those of the previous model.
- *Feature store:* This concept defines a repository where the definition, storage and access to features for model training and service are standardised. This repository allows exploring the use of new features aiming to optimise the performance of the models.
- *Metadata management:* The information about every execution of the MLOps workflow is saved in the ML metadata store to control the behaviour of the pipeline over time. This metadata management allows to perform data comparisons and to debug errors and anomalies. Among the data to be saved are the version of the executed pipeline, start and end dates of execution, time required in the execution of the pipeline, parameters of the optimised model, evaluation metrics of the model in the training and test sets, etc.
- *ML pipeline triggers:* The ML production pipeline can be automated to retrain models with new data as needed. For example, if a freight railway company wants to run the pipeline once a month, or only every two months depending on the amount of data it collects from its operations, or when there is clear model performance degradation, etc.

At a summary level, the diagram presented in Fig. 2 shows the phases of a MLOps workflow implementing good practices of CI/CD/CT which can be orchestrated through the use of tools such as Apache Airflow, Apache Beam or Kubeflow (Apache Software Foundation, 2016, 2015; Google, 2018; Li et al., 2018). These phases are described below.

- *Development and experimentation:* new techniques for data preparation, inclusion of features and algorithms are tested iteratively to develop the ML models in which the steps of the experiment are orchestrated.
- *Pipeline CI:* The source code is compiled by running various tests. The result of this stage is the creation of multiple pipeline components, such as packages and artifacts (i.e., code snippets that perform specific tasks) to be deployed at a later phase.
- *Pipeline CD:* The artifacts produced in the previous phase are implemented. The product of this phase is a pipeline deployed using the new model implementation.
- *CT:* In this phase, the pipeline is executed through a previously defined trigger (according to the needs of the company) to retrain the model with new data.
- *Model CD:* The new trained model is used as a prediction service.
- *Monitoring:* In this phase, statistics and all possible information of each execution are collected, such as information related to the data used to train the model, parameters obtained in the optimisation of the model and performance metrics of the model in production.

In addition, it is important to keep in mind that Continuous Training of ML models can have significant implications for energy consumption and climate impact. Some machine learning-based models necessitate the use of powerful computational resources such as GPUs and training these models for extended periods of time can result in significant energy consumption, leading to an increase in greenhouse gas emissions (Patterson et al., 2022; Tavares et al., 2022). Using energy-efficient hardware, such as specialised hardware accelerators or low-power CPUs, could be one solution for reducing the energy consumption and climate impact of Continuous Training. Another option is to use cloud providers that invest in renewable energy to power their data centres, which can help to reduce the environmental impact of training machine learning models. It is critical to consider the environmental impact of machine learning models and to look for ways to reduce this impact. As the demand for machine learning grows, it is critical to develop more energy-efficient and sustainable ML training methodologies.

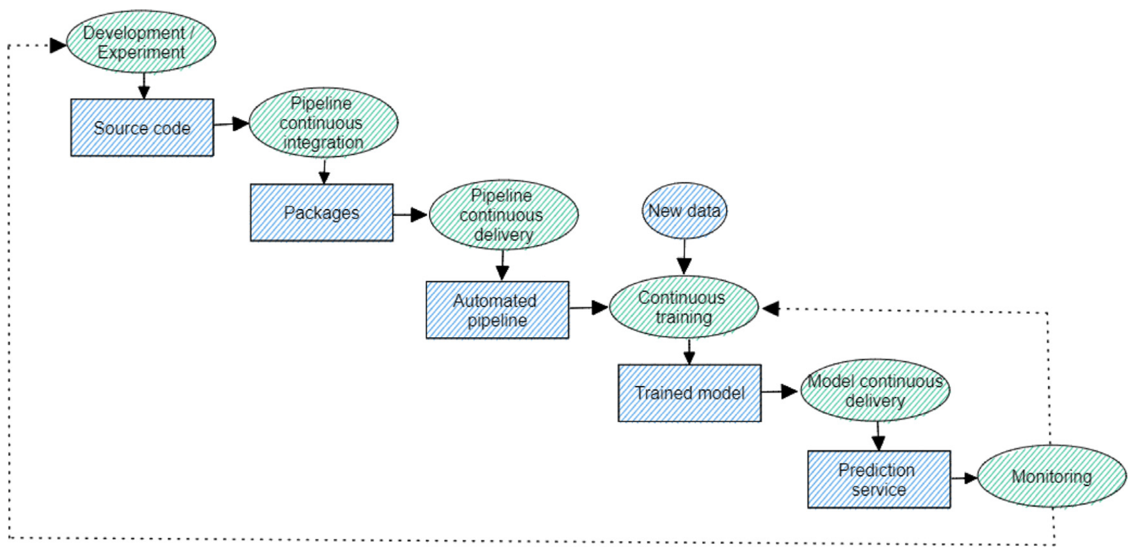


Fig. 2. Phases of a MLOps workflow implementing good practices of CI/CD/CT.
Source: Adapted from Kazmierczak and Schut (2021).

4. Results and discussion

This section describes the specific process developed in this study to create a MLOps workflow capable of automating the process from data ingestion to the construction of a prediction service, implementing good practices of CI/CD/CT, to predict delay times in freight rail operations, a process that was explained and discussed in our previous study (Pineda-Jaramillo et al., 2022b). It is important to highlight that the process described in this section includes the tools used for our specific use case, where many of the phases can be carried out in multiple ways and using different tools, considering the development and technological stack used by each company to build and run their projects.

4.1. Data extraction

Depending on the technological development of a company, the technological stack, the data storage and, in general, data security and privacy, the company will be able to provide the data in one or several formats. For instance, the most technologically advanced company will be able to provide access to their data lakes (the centralised repository of data designed to store and manage the data of the company) which are stored in cloud storage. Cloud storage involves the use of a network of computers and/or servers to store and manage the data of the company, and the most commonly used cloud storage are offered by Google, Amazon and Microsoft. It is also possible to store the data in local systems, which is commonly known as on-premises data storage. In this case, companies can give access to their data lakes to their employees. Additionally, it is also common that some companies send their data to a third party, which is called data sharing, and this data can be in different formats, such as JSON, XML, CSV, XLSX, etc.

The National Freight Railway Company of Luxembourg - CFL Multimodal, stores their data in local hardware (i.e., "on-premises"), where they provided information on their freight rail operations conducted between November 2019 to April 2021 between Bettembourg (Luxembourg) and other nine stations within the EU (Boulou, Champigneulles and Lyon in France; Zeebrugge and Antwerp in Belgium; Kiel and Rostock in Germany; Poznan in Poland; and Trieste in Italy). This data was provided in form of several excel files (.xlsx files), which contained tables related to trains, wagons, stations, and operations, with multiple attributes. However, managing large amounts of data in Microsoft Excel has several drawbacks, such as the requirement for manual operations, slow processing of large datasets, and the lack of replicability. Therefore, we opted to use a Relational Database Management System (RDBMS) to store and manage the data. In this study, PostgreSQL (Postgres) was selected as the RDBMS due to its free and open-source nature and adherence to established which is very popular due to its adherence to established Structured Query Language (SQL) rules and protocols (Bowman et al., 2001).

To facilitate data integration and merging of the tables, key features were identified and utilised (refer to Fig. 3), following the process of data modelling (Batra and Marakas, 1995; Datta and Thomas, 1999). The data modelling process involved the collection of requirements from CFL Multimodal, followed by conceptual modelling, where the key features for data integration were identified and utilised. The next step was logical modelling, where the data structure and relationships between tables were established. The final stage involved physical modelling, where the tables were created in the PostgreSQL database. To ensure data accuracy, testing and validation were conducted through queries using SQL. The use of an RDBMS allowed for the extraction of data and the execution of transformations, providing a more efficient and replicable approach to data management.

To transfer the data from the Excel files to Postgres, we first created the tables in Postgres. Next, we developed a Python script to extract the relevant information from the original Excel tables and populate the multiple tables in Postgres, storing each feature as type "string" (VARCHAR in Postgres). Then, we created a Python script specifically to automate the data migration process and run queries in Postgres using operators in Apache Airflow. This enabled us to streamline the data migration process and enhance the efficiency and accuracy of the migration, building in this way the data lake of the project. The use of operators in Apache Airflow allowed us to define and execute complex data workflows, ensuring that the migration was performed consistently and reliably. The process is illustrated in Fig. 4, which shows the step-by-step data migration process for all the tables.

4.2. Data preparation

With the data stored in the data lake built with Postgres, the next step is to perform a conversion of the data types of the existing features in all the existing tables of the data lake. For instance, time-dependent features are converted to timestamp type, and integer numeric variables are changed to integer type. This process is manual, but it is done only once, where it is sought that new acquired data over time meets the specifications indicated here. The new tables, with the cast data, are stored in the data warehouse, a concept similar to the data lake, with

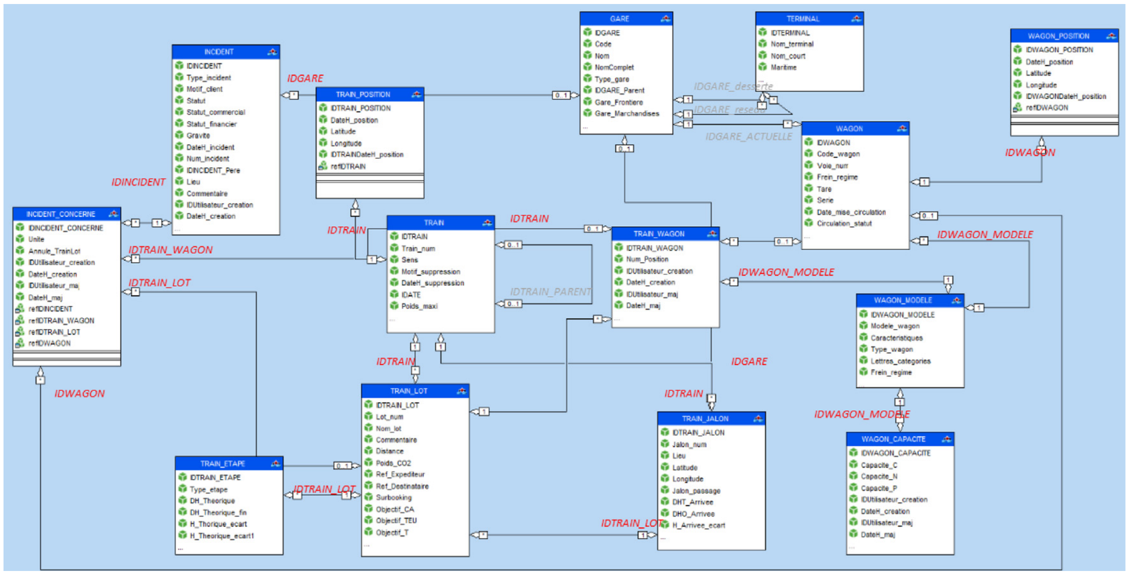


Fig. 3. Datasets provided by CFL Multimodal.

Table 1
Composition of the final dataset. Total of 10,265 trips.

Feature	Description	Feature distribution and statistics
TARGET: y	Arrival delay time [min]	Range: −281.0–1570.0; median: 12.0, mean: 74.4, std: 242.0
x1	Number of TEU (Twenty-foot Equivalent Unit)	Range: 0.0–984; median: 64.0, mean: 60.8, std: 18.0
x2	Train length [m]	Range: 34.0–720.0; median: 544.0, mean: 537.6, std: 108.7
x3	Distance of the TOTAL trip [km]	Range: 84.5–1454.1; median: 322.2, mean: 486.7, std: 280.1
x4	Departure delay time [min]	Range: −260.0–1562.0; median: 17.0, mean: 79.5, std: 241.7
x5	Distance between control stations [km]	Range: 1.5–131.1; median: 50.2, mean: 53.9, std: 37.6
x6	Train weight over train length [t/m]	Range: 0.9–4.1; median: 2.2, mean: 2.2, std: 0.5
x7	Train weight over number of wagons [t/wagon]	Range: 20.9–100.7; median: 69.0, mean: 66.5, std: 16.6

the difference that the casted data is stored here, and it is designed to enable and support analytical activities, which also works as the repository of features that can later be used for new explorations in order to optimise the models.

Once the data warehouse is created, the next step is to perform the operations of joining tables, cleaning data, transforming features, creating new features using domain knowledge, among others detailed in the aforementioned study (Pineda-Jaramillo et al., 2022b). The multiple steps performed in this data preparation are performed using Postgres, due to its enormous facility to perform this type of operations. All tables are also stored in the data warehouse, and multiple checks are performed during its execution, with the aim of validating that the features' types found in the tables make sense, that nulls are treated according to analyses previously performed (Pineda-Jaramillo et al., 2022b), among others. This process can be seen in Fig. 5.

4.3. Model training and validation

Once multiple tests and comparative analyses were performed in our previous study (Pineda-Jaramillo et al., 2022b), the best combination of features were identified to achieve optimal performance in the training of ML models (see Table 1), and multiple ML models were trained where it was validated that an optimised LightGBM model (an open-source gradient boosting framework) (Ke et al., 2017) is able to predict delay times in freight rail operations carried out by CFL Multimodal.

Thus, in order to automate the model training process, the tables built in the previous step located in Postgres are migrated to Python, using the pandas library (McKinney, 2010), this process is performed due to the great facility that Python has to perform the tasks of training and optimising ML models thanks to the large number of available open source libraries such as Pycaret and Scikit-learn (Ali, 2020; Pedregosa et al., 2011). After this migration, the data is randomly divided into training and test sets, with the aim of training the models using the first set and assessing the performance of the model using the last set, using the metrics of the coefficient of determination (R-squared), and the Mean Absolute Percentage Error (MAPE) as loss functions to assess the model's performance. These metrics are also logged in order to monitor the performance of the algorithm and compare it to the performance of various runs made over time. These two processes are depicted graphically in Fig. 6, where the model performed well with R2: 0.9365, and MAPE: 1.2400 (Pineda-Jaramillo et al., 2022b).

4.4. Prediction service

After the model is trained, evaluated, and validated, it is made available as a prediction service to enable CFL Multimodal to forecast the delay times resulting from disruptions and disturbances in their freight rail operations in real-time. The deployment of the model can be executed in different ways, with the "Model as a service" and the "Embedded model" being the two most widely used methods (Treveil and the Dataiku Team, 2020).

The "Model as a service" is commonly used when the ML model intends to serve different customers, and the prediction model is deployed on a web server exposing it as a Representational State Transfer (REST) API endpoint, in such a way that any application can get predictions by passing the input data (i.e., TEU count, train length, total distance of the trip, departure delay, distance between control stations, weight per length of the train, weight per wagon of the train) through of an API call. This method has been implemented in our MLOps workflow, in such a way that when the input data is entered, the model generates predictions as presented in Table 2.

Moreover, the "Embedded model" is intended for a private use of the model, where it is packaged in an application, and then published

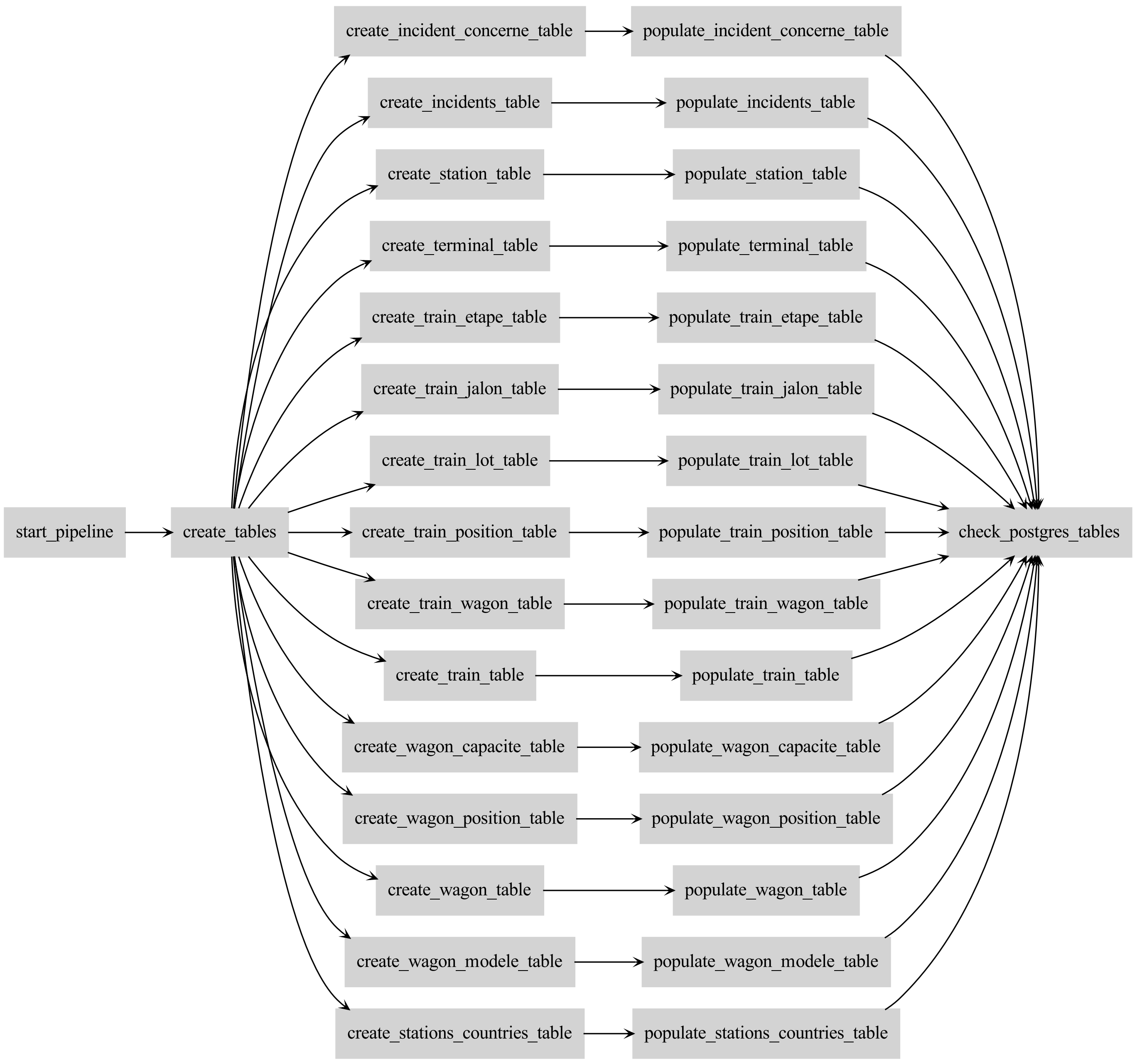

create_incident_concerne_table
populate_incident_concerne_table
create_incidents_table
populate_incidents_table
create_station_table
populate_station_table
create_terminal_table
populate_terminal_table
create_train_etape_table
populate_train_etape_table
create_train_jalon_table
populate_train_jalon_table
create_train_lot_table
populate_train_lot_table
start_pipeline
create_tables
create_train_position_table
populate_train_position_table
check_postgres_tables
create_train_wagon_table
populate_train_wagon_table
create_train_table
populate_train_table
create_wagon_capacite_table
populate_wagon_capacite_table
create_wagon_position_table
populate_wagon_position_table
create_wagon_table
populate_wagon_table
create_wagon_modele_table
populate_wagon_modele_table
create_stations_countries_table
populate_stations_countries_table

Fig. 4. Creation of tables in Postgres, and population of the tables using original data from Microsoft Excel.

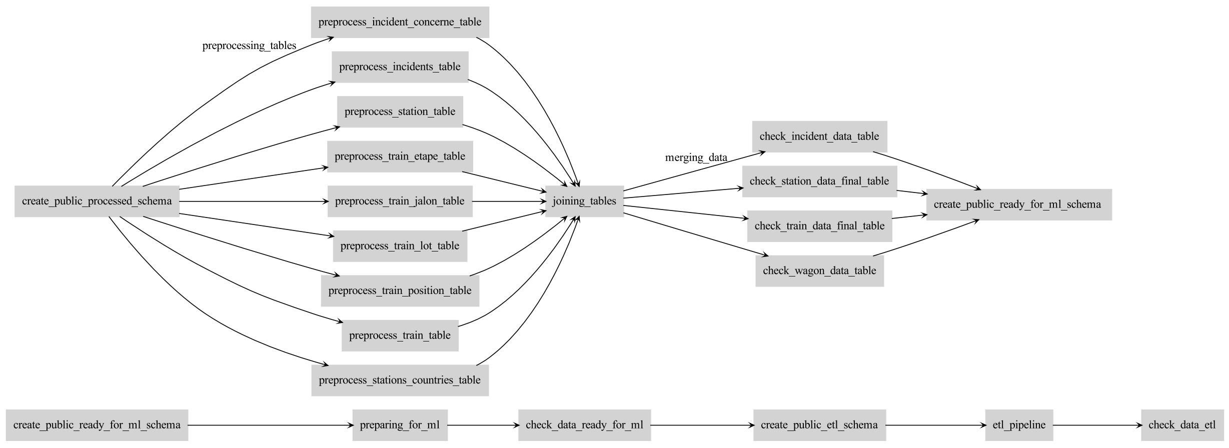

preprocess_incident_concerne_table
preprocessing_tables
preprocess_incidents_table
preprocess_station_table
preprocess_train_etape_table
create_public_processed_schema
preprocess_train_jalon_table
preprocess_train_lot_table
preprocess_train_position_table
preprocess_train_table
preprocess_stations_countries_table
joining_tables
merging_data
check_incident_data_table
check_station_data_final_table
check_train_data_final_table
check_wagon_data_table
create_public_ready_for_ml_schema
create_public_ready_for_ml_schema
preparing_for_ml
check_data_ready_for_ml
create_public_etl_schema
etl_pipeline
check_data_etl

Fig. 5. Data preparation.

Table 2
Predictions, using the model as a prediction service.

TEU count	Train length [m]	Total distance trip [km]	Departure delay [min]	Distance between control stations [km]	Weight per length of train [t/m]	Weight per wagon of train [t/wagon]	*PREDICTION:* Arrival delay time [min]
60	550	650	30	70	2.2	80	−9
50	500	700	35	75	2.2	78.6	26
40	540	750	40	80	1.9	87.5	11
48	666	513	111	682	2.2	81.3	88
34	424	1140	760	749	2.8	80.3	783
39	548	1102	397	531	2.4	130.5	419
88	517	1242	−14	782	1.2	39.9	−24
87	717	251	438	294	2	144.2	411
26	170	688	593	464	3.8	20.3	696
52	556	926	804	41	2.6	104.1	790
62	338	1402	−193	647	6.3	93.1	82
66	533	406	386	549	3.8	58.3	445
50	148	1349	−96	708	11.2	118.9	27
38	617	885	845	42	1.4	32.2	795
53	187	1195	208	164	6.5	202.5	332

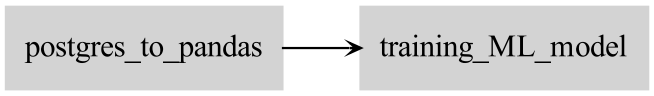


Fig. 6. Model training and validation.

for internal use (as it would be in this case, for a company like CFL Multimodal). A commonly used tool for creating this type of web apps that allows to create and deploy custom web apps for ML models as a prediction service for internal use, is Streamlit. Thus, a custom web app is built in such a way that, by selecting the values of the previously described input features, it is possible to predict the delay times resulting from disruptions and disturbances in CFL Multimodal freight rail operations, as shown in Fig. 7.

4.5. Orchestration of the entire process and use of triggers

In order to automate the entire MLOps process, Apache Airflow has been used in this study to build all the previously mentioned tasks, from data collection to model deployment as a prediction service, as can be seen in Fig. 8. Apache Airflow is an open source workflow management platform that provides functionality to manage and automate tasks, represented as Direct Acyclic Graphs (DAGs) (Apache Software Foundation, 2015), which owes its popularity to its native Python integration design. Additionally, the tasks can be automated as a "trigger as a service", where the pipeline is executed once there is a considerable amount of new data, or once every certain time (depending on the needs of the company), and in this way make a new execution from data ingestion to feeding the prediction service with the new model, monitoring that both the data behaviour and model performance are similar.

Additionally, in order to make all the different MLOps components effectively reusable and potentially shareable between multiple MLOps workflows, the orchestrator pipeline developed on Apache Airflow has been built on Docker (Merkel, 2014), that allows the containerisation of all the components of the pipeline so that each component has its own runtime and to have different programming languages and libraries working on the environment.

Even though all the processes described in this study were done using a simple Dell laptop with an Intel Core i9-10885H CPU @ 2.40 GHz with a 1TB NVMe class 40 SSD, a Linux Ubuntu 20.04.2.0 LTS 64-bit operating system, with a memory RAM of 32 GB, DDR4 and an NVIDIA Quadro P620 DDR5 Graphics Processing Unit, and the run time was around 50 min, these times may vary according to the development and technological stack used by each company, and may reach much lower times.

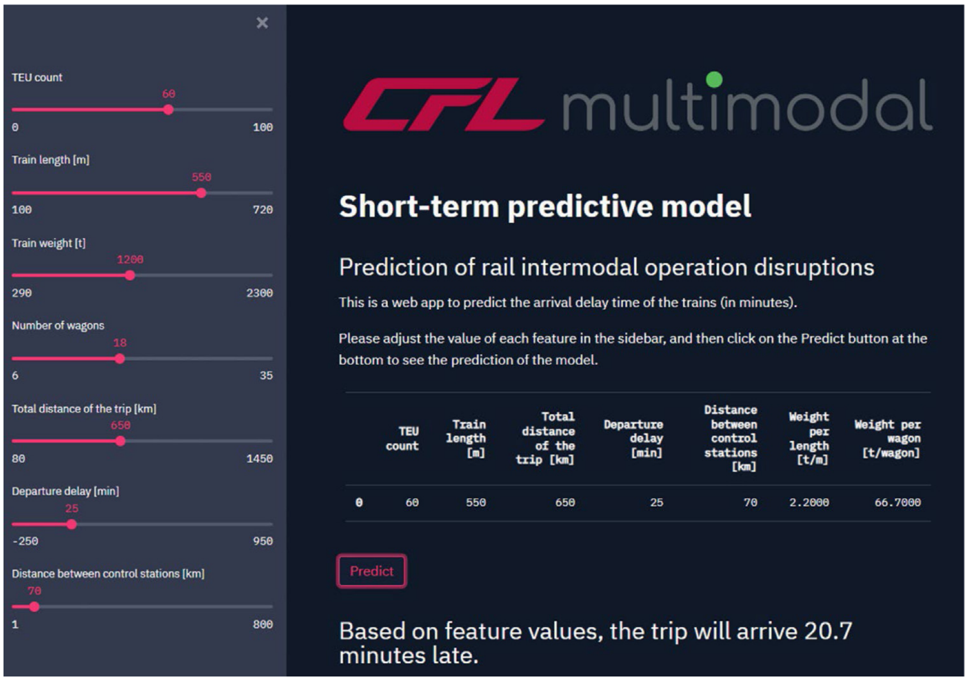
In general, previous studies that apply ML models to predict the occurrence of disruptions and disturbances in freight rail operations (passenger or freight), and the studies that aims to predict the delay times of the operations resulting from these disturbances and disruptions, train prediction models to analyse the impact of some features in the prediction. Additionally, many companies have also developed ML models trained on static datasets but obtaining the real value of a model in a production environment is still a challenge. These approaches do not allow a railway operator to know with certainty and in real time the delay times that a train will have due to a disruption or a disturbance in the rail operations, preventing it from making decisions to reduce the impact of these delays on the entire operation of the system.

This study presents a methodology to develop an MLOps workflow that allows to automate all the processes required by a railway freight company to analyse its operations data, predict delay times resulting from disturbances and disruptions in its operations, and make decisions to reduce the impact on the chain of delays in the entire operation of the system. The major contributions of this study are summarised below:

- The exploration of multiple tools to develop a MLOps workflow to automate all the processes required in the development of an ML model to predict the delay times of a freight train resulting from disturbances and disruptions in its operations, from data ingestion to the use of the model as a prediction system, using good practices of CI/CD/CT.
- Analysis of the advantages of building an MLOps workflow, including good practices of CI/CD/CT, with the aim of optimising freight rail operations for a particular company. Nevertheless, the methodology can be easily replicated for other companies.

The implications of this study's findings are significant for the field of railway operations and beyond. The development of an MLOps workflow for predicting delays in railway operations using machine learning models provides a new approach to automating the prediction of delays resulting from disturbances and disruptions in railway operations. This approach can help railway operators make better decisions in real-time, reducing the impact on the chain of delays in the entire operation of the system. The automation of ML pipelines using MLOps can also reduce the time and effort required for data preparation, model development, and model deployment, making it easier and faster for railway operators to implement predictive models in their operations.

Furthermore, the methodology presented in this study can be easily replicated for data from other freight rail companies, making it relevant to the broader field of transportation logistics. The use of MLOps in other transportation industries, such as aviation, shipping, and trucking, could also benefit from the findings of this study, improving the efficiency and reliability of operations.



TEU count: 60 (0–100)
Train length [m]: 550 (100–720)
Train weight [t]: 1200 (290–2300)
Number of wagons: 18 (6–35)
Total distance of the trip [km]: 650 (80–1450)
Departure delay [min]: 25 (-250–950)
Distance between control stations [km]: 70 (1–800)

Short-term predictive model

Prediction of rail intermodal operation disruptions

This is a web app to predict the arrival delay time of the trains (in minutes).

Please adjust the value of each feature in the sidebar, and then click on the Predict button at the bottom to see the prediction of the model.

	TEU count	Train length [m]	Total distance of the trip [km]	Departure delay [min]	Distance between control stations [km]	Weight per length [t/m]	Weight per wagon [t/wagon]
0	60	550	650	25	70	2.2000	66.7000

Based on feature values, the trip will arrive 20.7 minutes late.

Fig. 7. Use of the model as a prediction service, using the Embedded model technique.

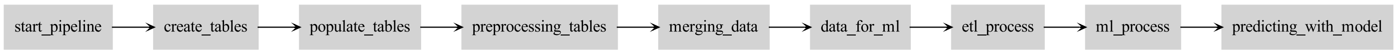


Fig. 8. Orchestration of the entire MLOps workflow using Apache Airflow.

5. Conclusions

This paper presents a methodology that enables railway operators to implement an automated MLOps workflow for their freight rail operations. The workflow extracts and processes data, trains an ML model that predicts delay times resulting from disruptions and disturbances, and deploys the model as a prediction service to identify delay times in real-time. This approach differs from previous studies, which mainly focused on identifying the most powerful predictive models or analysing the impact of specific features on the occurrence of disruptions or delays. However, these studies trained models using static datasets, limiting their practical value in real-world applications.

Using data from the National Railway Company of Luxembourg (CFL Multimodal) operations in a network across multiple European Union countries (Luxembourg, France, Germany, Belgium, Poland, Austria, Italy), and data processing and ML models previously trained and described in a previous study (Pineda-Jaramillo et al., 2022b), this study developed an MLOps workflow that automated data ingestion, model development, and model deployment, implementing good practices of CI/CD/CT.

On the other hand, this study leverages the use of various tools to develop the different processes within the pipeline, such as PostgreSQL, Python, Docker and Apache Airflow, but highlights the broad range of tools available for MLOps workflows, where their selection depends on the development and technological stack used by each company to build and run their projects. In fact, many technologically advanced companies develop many of these tasks using the numerous tools offered on cloud platforms, such as Google Cloud (GCP), Amazon Web Services (AWS), Microsoft Azure or IBM. In addition to using cloud platforms, many companies also use open-source tools to implement MLOps. These tools are often free and customisable, allowing companies to tailor their MLOps workflows to their specific needs. Some popular open-source MLOps tools include Kubeflow, MLflow, and MLJAR.

This study has made a unique contribution to the railway operations field by demonstrating the potential of MLOps in automating the processes of data ingestion, model development, and model deployment for real-time prediction of delays. The results showed the effectiveness of machine learning models in predicting delays in railway operations, which has not been extensively explored in the context of MLOps. By developing an MLOps workflow for this purpose, we have opened up new avenues for future research in this field.

Among the limitations of this study is the training of a single ML model (LightGBM), which was done because it was the best performing ML model from our previous study (Pineda-Jaramillo et al., 2022b) using the same data that were used in the present study. A more comprehensive MLOps workflow should consider training and optimising more ML models in the development of the pipeline, with the goal of evaluating the performance of various ML models as new data is used to train the models. This new task should be considered in a later phase of this study once data from new CFL Multimodal operations can be included. Furthermore, future research could concentrate on applying the MLOps approach to other aspects of railroad operations, such as scheduling, maintenance, resource allocation, and energy consumption. Moreover, additional research could look into incorporating real-time sensor data, such as weather and track conditions, into the MLOps workflow to improve the accuracy of delay predictions.

CRediT authorship contribution statement

Juan Pineda-Jaramillo: Conceptualisation, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Visualisation. **Francesco Viti:** Conceptualisation, Validation, Supervision, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Juan Pineda-Jaramillo reports financial support was provided by National Research Fund.

Data availability

Data will be made available on request

Acknowledgements

This study was possible thanks to the collaboration agreement signed between the University of Luxembourg and CFL Multimodal, and funding obtained by the Luxembourg National Research Fund FNR, through the project "ANticipatory Train Optimisation with Intelligent maNagEment (ANTOINE)", under grant BRIDGES2020/MS/14767177/ANTOINE. Special thanks to Nathalie Stef and Michael Maraldi from CFL for sharing the data used in this study.

References

Ali, M., 2020. Pycaret: An open source, low-code machine learning library in python.

Apache Software Foundation, 2015. Apache airflow, a platform created by the community to programmatically author. Sched. Monit. Work..

Apache Software Foundation, 2016. Apache beam, the easiest way to do batch and streaming data processing.

Barbour, W., Martinez Mori, J.C., Kuppa, S., Work, D.B., 2018. Prediction of arrival times of freight traffic on US railroads using support vector regression. Transp. Res. C 93, 211–227. http://dx.doi.org/10.1016/j.trc.2018.05.019.

Batra, D., Marakas, G.M., 1995. Conceptual data modelling in theory and practice. Eur. J. Inf. Syst. 4, 185–193. http://dx.doi.org/10.1057/ejis.1995.21.

Berger, A., Gebhardt, A., Müller-Hannemann, M., Ostrowski, M., 2011. Stochastic delay prediction in large train networks. OpenAccess Ser. Inform. 20, 100–111. http://dx.doi.org/10.4230/OASIcs.ATMOS.2011.100.

Bešinović, N., Goverde, R.M.P., Quaglietta, E., Roberti, R., 2016. An integrated micro–macro approach to robust railway timetabling. Transp. Res. B 87, 14–32. http://dx.doi.org/10.1016/j.trb.2016.02.004.

Bollegala, D., 2017. Dynamic feature scaling for online learning of binary classifiers. Knowl.-Based Syst. 129, 97–105. http://dx.doi.org/10.1016/j.knosys.2017.05.010.

Bowman, J., Emerson, S., Darnovsky, M., 2001. The Practical SQL HandBook: Using SQL Variants, Fourth ed. Addison-Wesley Professional.

Cacchiani, V., Caprara, A., Toth, P., 2010. Scheduling extra freight trains on railway networks. Transp. Res. B 44, 215–231. http://dx.doi.org/10.1016/j.trb.2009.07.007.

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., Wagenaar, J., 2014. An overview of recovery models and algorithms for real-time railway rescheduling. Transp. Res. B 63, 15–37. http://dx.doi.org/10.1016/j.trb.2014.01.009.

Corman, F., Kecman, P., 2018. Stochastic prediction of train delays in real-time using Bayesian networks. Transp. Res. C 95, 599–615. http://dx.doi.org/10.1016/j.trc.2018.08.003.

Datta, A., Thomas, H., 1999. The cube data model: a conceptual model and algebra for on-line analytical processing in data warehouses. Decis. Support Syst. 27, 289–301. http://dx.doi.org/10.1016/S0167-9236(99)00052-4.

Dollevoet, T., Huisman, D., Kroon, L., Schmidt, M., Schöbel, A., 2015. Delay management including capacities of stations. Transp. Sci. 49, 185–203. http://dx.doi.org/10.1287/trsc.2013.0506.

Dong, K., Romanov, I., McLellan, C., Esen, A.F., 2022. Recent text-based research and applications in railways: A critical review and future trends. Eng. Appl. Artif. Intell. 116, 105435. http://dx.doi.org/10.1016/j.engappai.2022.105435.

Garg, Satvik, Pundir, P., Rathee, G., Gupta, P.K., Garg, S., 2021. On continuous integration/ continuous delivery for automated deployment of machine learning models using mlops. In: Proc. - 2021 IEEE 4th Int. Conf. Artif. Intell. Knowl. Eng. AIKE 2021. pp. 25–28. http://dx.doi.org/10.1109/AIKE52691.2021.00010.

Géron, A., 2019. HandS-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, second ed. O'Reilly Media, Inc..

Ghofrani, F., He, Q., Goverde, R.M.P., Liu, X., 2018. Recent applications of big data analytics in railway transportation systems: A survey. Transp. Res. C 90, 226–246. http://dx.doi.org/10.1016/j.trc.2018.03.010.

Google, 2014. Kubernetes, an open-source container orchestration system for automating software deployment, scaling, and management.

Google, 2018. Kubeflow: The machine learning toolkit for kubernetes.

Goverde, R.M.P., 2010. A delay propagation algorithm for large-scale railway traffic networks. Transp. Res. C 18, 269–287. http://dx.doi.org/10.1016/j.trc.2010.01.002.

Goverde, R.M.P., Bešinović, N., Binder, A., Cacchiani, V., Quaglietta, E., Roberti, R., Toth, P., 2016. A three-level framework for performance-based railway timetabling. Transp. Res. C 67, 62–83. http://dx.doi.org/10.1016/j.trc.2016.02.004.

Goverde, R.M.P., Hansen, I.A., 2013. Performance indicators for railway timetables. In: 2013 IEEE International Conference on Intelligent Rail Transportation Proceedings. IEEE, pp. 301–306. http://dx.doi.org/10.1109/ICIRT.2013.6696312.

Granlund, T., Stirbu, V., Mikkonen, T., 2021. Towards regulatory-compliant MLOps: Oravizio's journey from a machine learning experiment to a deployed certified medical product. SN Comput. Sci. 2 (342), http://dx.doi.org/10.1007/s42979-021-00726-1.

Gürses-tran, G., 2022. Advances in time series forecasting development for power systems ' operation with MLOps. pp. 501–524.

Huang, P., Wen, C., Fu, L., Lessan, J., Jiang, C., Peng, Q., Xu, X., 2020. Modeling train operation as sequences: A study of delay prediction with operation and weather data. Transp. Res. E 141, 102022. http://dx.doi.org/10.1016/j.tre.2020.102022.

Huang, P., Wen, C., Peng, Q., Jiang, C., Yang, Y., Fu, Z., 2019. Modeling the influence of disturbances in high-speed railway systems. J. Adv. Transp. 2019, 1–13. http://dx.doi.org/10.1155/2019/8639589.

Kazmierczak, J., Schut, D., 2021. Practitioners guide to MLOps : A framework for continuous delivery and automation of machine learning. Google Cloud.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y., 2017. Lightgbm: A highly efficient gradient boosting decision tree. Adv. Neural Inf. Process. Syst. 2017-Decem, 3147–3155.

Kecman, P., Goverde, R.M.P., 2015. Predictive modelling of running and dwell times in railway traffic. Public Transp. 7, 295–319. http://dx.doi.org/10.1007/s12469-015-0106-7.

Kotliar, M., Kartashov, A.V., Barski, A., 2019. CWL-airflow: a lightweight pipeline manager supporting common workflow language. Gigascience 8, http://dx.doi.org/10.1093/gigascience/giz084.

Kuflik, T., Minkov, E., Nocera, S., Grant-Muller, S., Gal-Tzur, A., Shoor, I., 2017. Automating a framework to extract and analyse transport related social media content: The potential and the challenges. Transp. Res. C 77, 275–291. http://dx.doi.org/10.1016/j.trc.2017.02.003.

Lessan, J., Fu, L., Wen, C., Huang, P., Jiang, C., 2018. Stochastic model of train running time and arrival delay: A case study of Wuhan–Guangzhou high-speed rail. Transp. Res. Rec. J. Transp. Res. Board 2672, 215–223. http://dx.doi.org/10.1177/0361198118780830.

Li, D., Daamen, W., Goverde, R.M.P., 2016. Estimation of train dwell time at short stops based on track occupation event data: A study at a dutch railway station. J. Adv. Transp. 50, 877–896. http://dx.doi.org/10.1002/atr.1380.

Li, S., Gerver, P., MacMillan, J., Debrunner, D., Marshall, W., Wu, K.-L., 2018. Challenges and experiences in building an efficient apache beam runner for IBM streams. Proc. VLDB Endow. 11, 1742–1754. http://dx.doi.org/10.14778/3229863.3229864.

Luo, J., Peng, Q., Wen, C., Wen, W., Huang, P., 2022. Data-driven decision support for rail traffic control: A predictive approach. Expert Syst. Appl. 207, 118050. http://dx.doi.org/10.1016/j.eswa.2022.118050.

Lwakatare, L.E., Raj, A., Crnkovic, I., Bosch, J., Olsson, H.H., 2020. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. Inf. Softw. Technol. 127, 106368. http://dx.doi.org/10.1016/j.infsof.2020.106368.

Marković, N., Milinković, S., Tikhonov, K.S., Schonfeld, P., 2015. Analyzing passenger train arrival delays with support vector regression. Transp. Res. C 56, 251–262. http://dx.doi.org/10.1016/j.trc.2015.04.004.

McKinney, W., 2010. Data Structures for Statistical Computing in Python. 56–61. http://dx.doi.org/10.25080/Majora-92bf1922-00a.

Merkel, D., 2014. Docker: lightweight linux containers for consistent development and deployment. Linux J. 2014, http://dx.doi.org/10.5555/2600239.2600241.

Mesa-Arango, R., Pineda-Jaramillo, J., Araujo, D.S.A., Bi, J., Basva, M., Viti, F., 2023. Missions and factors determining the demand for affordable mass space tourism in the United States: A machine learning approach. Acta Astronaut. 204, 307–320. http://dx.doi.org/10.1016/j.actaastro.2023.01.006.

Milinković, S., Marković, M., Vesković, S., Ivić, M., Pavlović, N., 2013. A fuzzy Petri net model to estimate train delays. Simul. Model. Pract. Theory 33, 144–157. http://dx.doi.org/10.1016/j.simpat.2012.12.005.

Minbashi, N., Sipilä, H., Palmqvist, C.-W., Bohlin, M., Kordnejad, B., 2023. Machine learning-assisted macro simulation for yard arrival prediction. J. Rail Transp. Plan. Manag. 25, 100368. http://dx.doi.org/10.1016/j.jrtpm.2022.100368.

Nair, R., Hoang, T.L., Laumanns, M., Chen, B., Cogill, R., Szabó, J., Walter, T., 2019. An ensemble prediction model for train delays. Transp. Res. C 104, 196–209. http://dx.doi.org/10.1016/j.trc.2019.04.026.

Patterson, D., Gonzalez, J., Holzle, U., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D.R., Texier, M., Dean, J., 2022. The carbon footprint of machine learning training will plateau, then shrink. Computer (Long. Beach. Calif) 55, 18–28. http://dx.doi.org/10.1109/MC.2022.3148714.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É, 2011. Scikit-learn: Machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.

Pineda-Jaramillo, J., Barrera-Jiménez, H., Mesa-Arango, R., 2022a. Unveiling the relevance of traffic enforcement cameras on the severity of vehicle–pedestrian collisions in an urban environment with machine learning models. J. Safety Res. http://dx.doi.org/10.1016/j.jsr.2022.02.014.

Pineda-Jaramillo, J., Bigi, F., Viti, F., 2022b. A data-driven model for short-term prediction of arrival delay times in freight rail operations. In: Triennial Symposium on Transportation Analysis Conference. Mauritius Island.

Pineda-Jaramillo, J., Viti, F., 2023. Identifying the rail operating features associated to intermodal freight rail operation delays. Transp. Res. C 147, 103993. http://dx.doi.org/10.1016/j.trc.2022.103993.

Ruf, P., Madan, M., Reich, C., Ould-Abdeslam, D., 2021. Demystifying mlops and presenting a recipe for the selection of open-source tools. Appl. Sci. 11, http://dx.doi.org/10.3390/app11198861.

Subramanya, R., Sierla, S., Vyatkin, V., 2022. From DevOps to MLOps: Overview and application to electricity market forecasting. Appl. Sci. 12, 9851. http://dx.doi.org/10.3390/app12199851.

Talby, D., 2019. Why machine learning models crash and burn in production. Forbes.

Tavares, C., Wang, X., Saha, S., Grasley, Z., 2022. Machine learning-based mix design tools to minimize carbon footprint and cost of UHPC. Part 1: Efficient data collection and modeling. Clean. Mater. 4, 100082. http://dx.doi.org/10.1016/j.clema.2022.100082.

Treveil, M., the Dataiku Team, 2020. Introducing MLOps. how to Scale Machine Learning in the Enterprise. O'Reilly Media, Inc., Sebastopol, CA, USA.

Van der Meer, D., Goverde, R.M.P., Hansen, I.A., 2010. Prediction of train running times using historical track occupation data. In: 12th World Conference on Transport Research. Lisbon.

Wang, X., Li, S., Cao, Y., Xin, T., Yang, L., 2022. Dynamic speed trajectory generation and tracking control for autonomous driving of intelligent high-speed trains combining with deep learning and backstepping control methods. Eng. Appl. Artif. Intell. 115, 105230. http://dx.doi.org/10.1016/j.engappai.2022.105230.

Wen, C., Huang, P., Li, Z., Lessan, J., Fu, L., Jiang, C., Xu, X., 2019. Train dispatching management with data- driven approaches: A comprehensive review and appraisal. IEEE Access 7, 114547–114571. http://dx.doi.org/10.1109/ACCESS.2019.2935106.

Wen, C., Li, Z., Lessan, J., Fu, L., Huang, P., Jiang, C., 2017. Statistical investigation on train primary delay based on real records: evidence from Wuhan–Guangzhou HSR. Int. J. Rail Transp. 5, 170–189. http://dx.doi.org/10.1080/23248378.2017.1307144.

Xu, J., 2022. Mlops in the financial industry: Philosophy, practices, and tools. In: The Future and FinTech. WORLD SCIENTIFIC, pp. 451–488. http://dx.doi.org/10.1142/9789811250903_0014.

Yaghini, M., Khoshraftar, M.M., Seyedabadi, M., 2013. Railway passenger train delay prediction via neural network model. J. Adv. Transp. 47, 355–368. http://dx.doi.org/10.1002/atr.193.

Yuan, J., Goverde, R., Hansen, I., 2002. Propagation of train delays in stations. In: Allan, J., Hill, R.J., Brebbia, C.A., Sciutto, G., Sone, S. (Eds.), Computers in Railways VIII. WIT Press, Southhampton, UK, pp. 975–984.