UCLouvain

École polytechnique de Louvain

Analyzing and Predicting SNCB's Train Delays Using Open Data

Authors: **Samuel de Meester de Ravestein, Timothy Geerkens**
Supervisor: **Pierre Schaus**
Readers: **Alexandre Dubray, Augustin Delecluse, Guillame Derval**
Academic year 2023–2024
Master [120] in Computer Science

Acknowledgments

We would like to thank our attentive reader, Alexandre Dubray, for his precious guidance. We are grateful to our supervisor, Pierre Schaus, for giving us the opportunity to explore the fascinating field of public railway transportation and for fostering our curiosity by allowing us to attend the workshop "Comment l'IA peut accélérer la mobilité et le transport connectés ?" in Liège. We express our appreciation to Augustin Delecluse and Guillaume Derval for dedicating their time to reviewing our thesis.

We extend our heartfelt thanks to our friends from the "Réaumur" group who have been a source of encouragement throughout this year, as they have been in previous years. The journey through our computer science degree would not have been the same without your support and companionship.

Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

Contents

Chapter 1

Introduction

Today, one of the biggest problems of public transportation is its unpredictability. It is difficult to rely on a transportation method that cannot confidently ensure an arrival time. Everyone has experienced missing a train because of a prior delay. This master's thesis will focus on the railway system, and its goal is twofold. First, it is important to understand the scope of the problem. So, this work will clearly define what a late train means and use it to make statistics on train delays. The SNCB[1] already has its own definition of a late train, but it will be shown that this definition is not the most suitable. The second objective is to accurately predict near-future delays, which could be shared with the public via a website, informing them about potential rail journey delays.

The thesis is structured in six chapters. The first one is this introduction to the topic. Chapter 2 presents some important definitions and information on different kinds of models that will be used later. Then, chapter 3, will present previous work on the topic. Next, chapter 4 presents the two architectures used to build prediction models in this work. Followed by a description of our dataset and how it was built. After this, chapter 5 directly broaches one of our goals. The formalization of the definition of a late train and various statistics on delayed trains in the Belgian railway network. Then, chapter 6 details how the architectures defined in chapter 4 are being used in practice. It also provides a formal definition of the prediction problem. Subsequently, chapter 7 is about evaluating the different models and understanding the results. Finally, this paper finishes with a conclusion and possible areas of improvement for future work.

[1] SNCB stands for National Railway Company of Belgium in French.

Chapter 2

Technical Background

2.1 Formal definitions

Trains are the center of this work, they go from one station to another, taking some time for each move. A station is a hub where one or more trains can stop and let passengers enter or leave the wagons. Each station is characterized by a unique identifier.

Now, we can represent train trips through the railway network. Trips represent a single train going from a starting station to a terminus station. No change of trains is possible during a trip. A trip has two components: a spatial one and a temporal one. The spatial component can be represented by the set of stations through which the train passes. The temporal component is contained in the time at which the train arrives (a) and leaves (l) each station (s). At every station, there is a planned arrival and a planned leave time. Since trains don't always arrive and leave at the planned time a real arrival and a real leave time exists. Therefore, a trip is defined as a sequence of 5-tuples of station identifiers, planned and real arrival time, and planned and real leaving time.

Definition 1 *Let s_j be the j^{th} station in a trip, a_j^p and l_j^p the planned arrival and leave time at that station, and a_j^r and l_j^r the real arrival and leave time at that station. Then, this trip t of length n is defined as :*

$$t = \langle (s_1, a_1^p, l_1^p, a_1^r, l_1^r), ..., (s_n, a_n^p, l_n^p, a_n^r, l_n^r) \rangle$$

Trips only represent trains from their starting station to their terminus. Therefore, a single trip cannot represent a traveler's journey through

the network alone. A journey can be decomposed into multiple trip segments. Every time the traveler changes train a segment of the new trip is added to the sequence. Each segment starts at the station where the passenger boards the train and stops where he leaves the train.

Definition 2 *Let $t_i \in T$ be a trip in the set of all trips T. Let t_i^r be the r^{th} station in trip t_i, and (t_i^r, t_i^s) be the sequence of all stations between index r and s in that trip. Then, j is the journey from a station $t_1^{i_1}$ to a station $t_m^{k_m}$.*

$$j = \langle (t_1^{i_1}, t_1^{k_1}), ..., (t_m^{i_m}, t_m^{k_m}) \rangle$$

Train delays are the central part of this work, as such it is also important to define them. Planned and real arrival times exist for every stop on every trip. A delay happens when the real arrival time is larger than the planned one. If the real arrival time is smaller than the planned one the train is said to be in advance. However, this is a much rarer occurrence and not the focus of this work. With these concepts understood, we can define the arrival delay.

Definition 3 *Let s be a station in a trip t as defined in 1, $a_{t,s}^r$ be the real arriving time and $a_{t,s}^p$ be the planned arriving time. Then, $d_{t,s}^a$ is the leaving delay at station s for trip t and is defined as follows:*

$$d_{t,s}^a = a_{t,s}^r - a_{t,s}^p$$

In the same way that trains can be delayed in arriving, trains can also be delayed in leaving the station. In that case, the formula stays the same, except we use planned and real leaving time instead of arrival time.

Definition 4 *Let s be a station in a trip t as defined in 1, $l_{t,s}^r$ be the real leaving time and $l_{t,s}^p$ be the planned leaving time. Then, $d_{t,s}^p$ is the leaving delay at station s for trip t and is defined as follows:*

$$d_{t,s}^l = l_{t,s}^r - l_{t,s}^p$$

Those definitions will be used in section 5 to show multiple ways to define a late train. When using the definition of a trip, only the necessary elements will be shown in the tuple for conciseness.

2.2 Regression models

Several machine learning models are presented to solve the task at hand. This section presents each kind of model that will be used later on.

First, linear regression is a technique used to model the relationship between one or more regressor variables (features) and a response variable (target). The goal is to predict the target from features. It works by finding the optimal linear function that minimizes the mean squared error. This error is computed as the sum of squared distances between the function and points in the dataset. The function takes the following form :

$$Y = X_0 + b_1X_1 + b_2X_2 + ... + b_nX_n + \varepsilon \tag{2.1}$$

Where Y is the target, X_i is a regressor variable, and b_i is a regression coefficient attached to that variable. Coincidentally, ε is the error term, and X_0 is the intercept, the value of Y when all variables are equal to zero. This technique is very simple and straightforward but lacks the ability to model nonlinear dependence and tends to perform poorly in a nonlinear environment [10].

Ridge regression is an improvement over linear regression when the regressor variables are heavily correlated. In that case, linear regression estimators tend to overfit [27]. In ridge regression, a ridge parameter introduces a degree of bias to the regression estimates to reduce errors. Overfitting can be deduced from the regression coefficients. Indeed, they tend to be large when it happens. Therefore, the parameter is the sum of squared regression coefficients multiplied by a hyperparameter λ. λ is used to control the importance of the parameter. This parameter is added to the objective function of the linear regression, which becomes as follows :

$$MSE = \sum_{i=1}^{E}(Y_i - \hat{Y}_i)^2 + \lambda \sum_{j=1}^{n} b_j^2 \tag{2.2}$$

With E, the number of examples in the dataset, and n the number of features [13].

Next are the Decision trees, a supervised learning algorithm that can be used for regression and classification. They represent decisions as a

tree of choices. Nodes represent outcomes, and branches represent decision rules. Nodes with no children are called leaves and are the final predictions of the tree [3]. Optimal decision trees achieve the highest possible performance on training data under well-defined constraints. Learning optimal decision trees is an NP-hard problem, and traditional algorithms use a greedy approach to approximate a solution. However, this technique can lead to non-optimal trees in terms of accuracy. The one we use is `DL8.5`, which works under a depth limit constraint. It considers paths through the tree as itemsets. This makes the use of itemmining algorithms possible to find solutions. Even using state-of-the-art algorithms, optimal trees take much longer to train than greedy-based decision trees [1].

Random forest learning is an ensemble learning method using decision trees. A single decision tree, even if optimal, is generally not very good at analyzing complex data with many interacting features. The idea of random forests is to use a large set of trees using different datasets and feature sets to solve those problems. Then, aggregate their results to make one prediction. In the case of classification, the aggregation generally consists of a majority vote. When the model is used for regression, a mean of all predictions can be used. A weighted random forest uses the same principles but gives each tree a weight depending on its individual performance and uses weighted votes and means. Together, they can model interactions between features more accurately and make better predictions [4].

Finally, neural networks are structures used for classification or regression. They are composed of multiple layers of neurons, nodes that contain a function in the form :

$$y = \sigma(W^T x) \tag{2.3}$$

Where W^T is a vector of weights, x is an input vector, y is the output, and σ is a nonlinear activation function. The network comprises an input layer, one or more hidden layers, and an output layer. Neurons in each layer take the output of neurons from the previous layer as input. Except for the input layer, which takes features as input. The output of the output layer represents the prediction of the model. The network is trained on a dataset using the backpropagation algorithm, which finds the optimal weight vectors for each neuron. These models can arbitrarily well approximate any function with only two hidden layers if they

have enough neurons. This is the universal approximation theorem. It is very powerful and ensures that neural networks can analyze complex nonlinear relationships. In practice, using more than two hidden layers often offers better performance [14, 20].

Chapter 3

State of the art

This chapter will discuss state-of-the-art approaches to public transport delays, focusing on train networks first and then expanding to other public means of transport. The problem of delays in public transport can be tackled from different angles. Depending on the scale considered, the solutions might differ. A station has tracks and platform usage that can be optimized, while at the network level, the schedule is the main focus.

3.1 Station level

The most challenging aspect of train delays lies in the interdependencies inherent in the undirected graph structure. Trains circulate from station to station in both directions. This intricacy due to track junctions and the knock-on delay is explored at the station level in [31]. The knock-on delay is the delay provoked on a train by an other train arriving late. In their article, Yuan and Hansen focused on understanding train interactions at The Hague CS station in the Netherlands. They found empirical evidence that an analytical stochastic model can estimate the propagation of train delays very well. Therefore, their model is suitable for optimizing station capacity utilization, including platforms and tracks.

Based on the simulation software Rail Traffic Controller (RTC) from Berkeley Simulation Software, [7] found that the primary source of delay was the waiting time caused by a meet conflict (one or more trains traveling in opposite directions). This supports the idea that capacity

utilization of the railway network is a prime reason for knock-on delays. Minor delays can be absorbed under a certain utilization threshold, but delays start to have ripple effects past this threshold.

Capacity utilization alone can't explain variation in punctuality, as underlined by [18], who has examined empirical results from Norwegian studies. The authors highlight the impact of passenger numbers. Effective management of boarding and alighting passengers is crucial for ensuring the punctuality of local and regional trains in congested areas. The well-known self-discipline in Japan is one of the reasons why Japan has such an efficient railway system.

3.2 Network-wide level

At a network-wide scale, different models exist to represent the network and derive predictions from them.

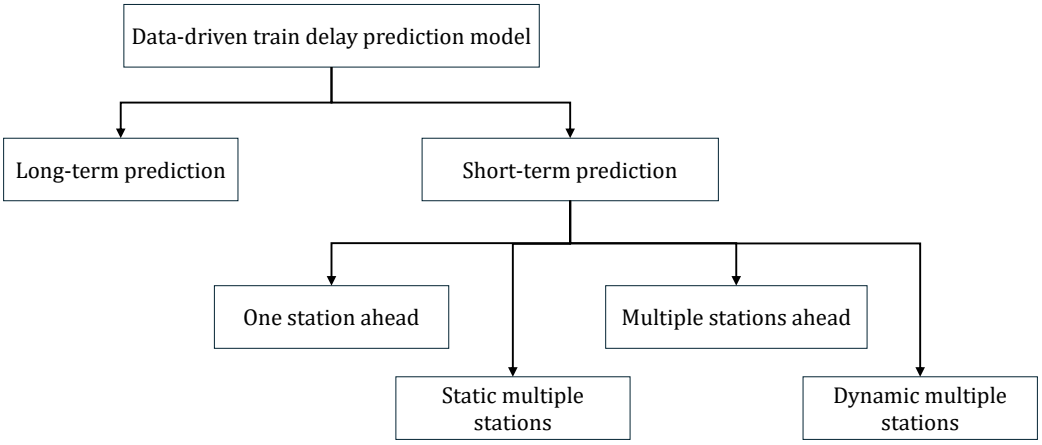


Figure 3.1: Hierarchies of data-driven train delay prediction model in terms of prediction horizon [24].

This thesis focuses on short-term prediction. Short-term, in the sense that the focus is on using real-time data to make predictions for the next station and a few stations ahead.

Different models are used in the literature. The main ones are presented in the subsequent subsections.

3.2.1 Machine Learning models

Machine learning models are popular in the literature as they are powerful tools to capture nonlinear relationships in high-dimensional space. The most widely used one for train delay prediction is the random forest regression [24].

There is no definitive evidence that certain machine learning algorithms consistently outperform others. Each machine learning model requires feature engineering to accurately capture spatial and temporal flow patterns. These models are robust in handling noisy data. However, their primary drawback is their lack of interpretability. Understanding the interactions among stations is essential for improving railway infrastructure and scheduling.

3.2.2 Markov chain model

In ŞAHIN İsmail article [21], the time allowance for trains is analyzed. Time allowances are used to avoid deviating from the schedule. On top of the minimum time required for the operation, they include extra time to absorb small delays.

To do so, the author has studied normal train movement with train movement subject to disturbances. By defining states as discrete delay levels (state no1: ≤ -1min, ..., state no6: > 11min) and proper transition matrices, the author applied his Markov chain model to the railway line between Istanbul and Ankara. The data was provided by the Turkish State Railways. The goal was to compute the steady state (i.e., the long-term probability that the system will be in each state) under the two scenarios. The following numbers represent a train's probability of being in each state.

1. *Without disturbance*: **0.475**, 0.317, 0.111, 0.078, 0.017, 0.001 (in state order).
2. *With disturbance*: 0.004, 0.015, 0.008, 0.022, 0.113, **0.825** (in state order).

Another use case for Markov Chain is the prediction case. In a paper by Gaurav, R., & Srivastava, B [9], a Markov chain model is trained on a dataset of "known train" and evaluated on an "unknown train" dataset.

Unknown in the sense that the trains are not drawn from the same data set. The goal is to assess the knowledge transfer to unseen trains. In their paper, a train must be understood as a physical train that has made several journeys in the considered time period.

Both random forest regressions (RFRs) and ridge regressions (RRs) were used to predict the delay at the next station. The model was given the four previous stations with their respective delays, along with the current month, the weekday, the train type, and the zone (which zone the train belongs to). In case a station had never been seen before in the training data, a K-nearest neighbors approach was used to find the most similar station among known ones.

RFR models outperformed RR models. The authors admit that the results on the unknown trains are not promising because a significant amount of data is unavailable, with a large portion of the train being "unknown." However, the paper has still been able to determine experimentally that most of the trains follow a 1-order Markov Process. This means that for most trains, the delay at the next station only depends on the delay from the previous one.

3.2.3 Bayesian network model

By definition, Bayesian networks (BN) are directed acyclic graphs, meaning they cannot be directly applied to a national railway graph where bidirectional arcs are the norm. The railway graph is composed of nodes corresponding to stations, and a connection between two nodes exists if a train between the two associated stations exists. The literature offers two approaches: consider a subset of the graph or assume an underlying structure (a station influences a neighbor but not vice versa).

Several papers have decided to analyze corridors, such as Corman F. & Kecman P., who analyzed the Swedish southern mainline between Stockholm and Malmö [6]. A second paper investigates two Chinese high-speed lines [15]. Both papers do not simply assign a delay variable to each station to construct the structure but rather break down what is happening at each station level.

The first paper separates the departure and arrival variables and includes the relationship with the following train. The authors have de-

fined several building blocks to match the different scenarios: the following train is a connection, or the current or following train does not stop at the station. After determining the structure, the parameters were fitted. The model was tested by predicting on the test set with a horizon of 60 min. The model produces reliable predictions for horizons of up to 30 min.

The second article [15] uses different building blocks. It does not aim at predicting future delays directly but rather to analyze the consequences of disruptions. The building blocks corresponding to the random variables for each station are the number of delayed trains at the given station, the total delay time at a given station, and the primary delay (disruption). The Bayesian structures of each building block are learned for every station, and then the general Bayesian structure representing the railway line is finally learned. Compared to a regression model which takes as input the parent nodes, the Bayesian performs better in general for each of the three random variables.

The following paper aims to represent the delay dependencies between transit network stops in Long Island [25], located just off the coast of New York. Because the data used by the authors was gathered through a crowdsourced real-time transit information app, it is incomplete, and some missing data had to be inferred. The BN structure was learned from the data and by blacklisting all impossible interactions. It is impossible in the sense that if no train connects two stations, then no direct interaction should exist between them.

Using their Bayesian network model, the authors introduced two scoring metrics: *inducer* and *susceptible*. The *inducer* score aims at evaluating the station impact as a potential source of delay that propagates in the network. On the other hand, the *susceptible* score evaluates how a station is influenced by its neighbors.

A track expansion project was used as a case study. The results support the Bayesian Network (BN) approach to understanding interdependencies. Some dependencies observed in the BN before the expansion disappeared after the expansion due to congestion relief. Also, the metric scores indicate decreased susceptibility to delays. Therefore, the model has successfully identified the network dependencies and indicated candidate rail links/corridors for subsequent improvement investments.

3.3 Other transport means

The problem of public transport delay is not limited to trains. The closest public transport in terms of properties and constraints is the subway, followed by flights. Public buses are different, as they do not operate on their own route, making them dependent on traffic jams. Furthermore, buses are not constrained by the limited number of platforms in a station or terminal gate at an airport.

The problem of delays in subways is not the same as for trains. Subways could be viewed as trains with much higher frequency and much shorter travel time between stations. As explained by the Belgian national railway operator, SNCB [1], many of the delay are caused by the infrastructure in itself: work, technical breakdown, weather impact on the equipment etc. But also the flow of passengers and accidents happening on a track. In comparison, for the Belgian subway [2] (no English version was available), the source of delay is mainly caused by passengers. The infrastructure of a subway is more dense, making it easier to maintain and more protected because of its underground location. The impact of a delay can have a ripple effect, but the consequences are less severe thanks to the high frequency of journeys. The literature focuses on optimizing timetables rather than dealing with delays of subways [28].

The models used to predict delays for flights resemble the ones for train stations. Xu et al. [30] and Wu et al. [29] built a Bayesian network to model delay propagation. Graph theory is used to model the schedule of an airline. Other more traditional techniques such as Machine learning (KNN, SVM, random forest, and others) are used in the literature, but also probabilistic models using common distributions such as Poisson and the Normal distribution [23]. What sets the problem of delay in aviation apart from train delays is the fact that the delays are coming from the airport themselves, and not the route in between. Meanwhile, train tracks in between stations can be a source of delays, impacting neighboring stations directly.

[1] https://www.belgiantrain.be/en/support/faq/faq-routes-schedules/faq-delays-canceled-trains

[2] https://stibstories.be/retards-metro-stib/

Chapter 4

Methodology

This chapter will start by explaining a few concepts needed to understand our work. Then, how data are gathered and processed will be discussed. Finally, the methodology used to create and train models will be described.

4.1 Markov chains

The concept of Markov chains was first introduced by AA Markov in 1906 [2]. A Markov chain is a stochastic model in which the future state solely depends on previous states. The order of such a model represents the number of previous states it considers. A first-order Markov chain is one where each subsequent state depends solely on the previous one. In comparison, a N-order chain is one where each event depends on the N preceding events. A transition matrix defines the probabilities of going from one state to another depending on the history of length N [17, 22].

Definition 4.1.1 *Let $\mathcal{X}$ be a state space and $P = (p_{i,j})_{i,j\in\mathcal{X}}$ be a transition matrix such that $p_{i,j} \in [0, 1]$ is the probability of transitioning from state X_i to state X_j. A first-order Markov chain is a sequence of states $(X_n)_{n\in\mathbb{N}}$ evolving according to T, satisfying the Markov property:*

$$\begin{aligned}&\Pr(X_{n+1} = x_{n+1} \mid X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1)\\ &\quad = \Pr(X_{n+1} = x_{n+1} \mid X_n = x_n) = p_{n,n+1}\end{aligned}$$

for all $n \in \mathbb{N}$.

This definition can be adapted for N-order models by changing the dimension of the transition matrix.

4.2 Markov random field

Let $X = X_1, ..., X_n$ be a finite set of random variables, a stochastic process. A Markov random field, or Markov network, is similar to a Markov chain except that the structure linking the random variables in X is a graph instead of a chain. It represents a joint probability distribution of the set of random variables. Let's define $G = (N, E)$ as an undirected graph where $N = n_1, n_2, ..., n_t$ is a set of nodes, and E is a set of edges. Two nodes are considered adjacent if an edge exists between them. Another important concept to define before going further is cliques. A clique is a maximal subgraph in G that is fully connected in E. In this context, maximal means no other nodes can be added to the clique without breaking the fully connected condition. Each node contains a random variable from the set X, and each edge is a link between nodes. The value of a random variable depends only on random variables contained in all adjacent nodes [5]. This property can be expressed as follows:

$$P(x_i \mid x_j, i \neq j) = P(x_i \mid x_j, (i, j) \in E) \tag{4.1}$$

Markov random fields encode dependencies between variables, just like Bayesian networks. However, Bayesian networks are directed and acyclic [19], which is a big problem when representing railway networks. In a railway system, trains circulate both ways to all stations, making the graph cyclic by design. This also means that dependencies between random variables are cyclic. For instance, a delay in station A will cause a delay in station B, which can also cause a delay in station A! This is what makes Markov random fields interesting for this problem.

4.3 Data Collection

SNCB doesn't make its historical data publicly available. However, it offers two API endpoints, one for schedule data and a second for real-time data. Those endpoints require a key, available on demand.[1]

[1] https://www.belgiantrain.be/en/3rd-party-services/mobility-service-providers/public-data

Therefore, it was important to promptly build up our historical data database. To do so, GitHub Actions was the key element. It enables routine tasks, which in our case are retrieving both scheduled and real-time data.

The retrieved data from the SNCB API has different lifetime relevancy. Indeed, according to SNCB, the real-time data is updated every 30 seconds and is available for 6 hours, while the schedule data is updated daily. For this reason, the real-time data is collected much more frequently.

The right routine compromise is to collect the real-time data every 30 minutes and the schedule data twice daily. Even though the SNCB states that its real-time data is available for 6 hours, we noticed this was not true. Hence, the routine is every 30 minutes.

Because our entire analysis is based on gathering those data, we deemed it judicious to save it twice. One is directly available on a GitHub repository, and the second is stored on a Google Drive. A flowchart of the retrieval process is shown in figure 4.1.

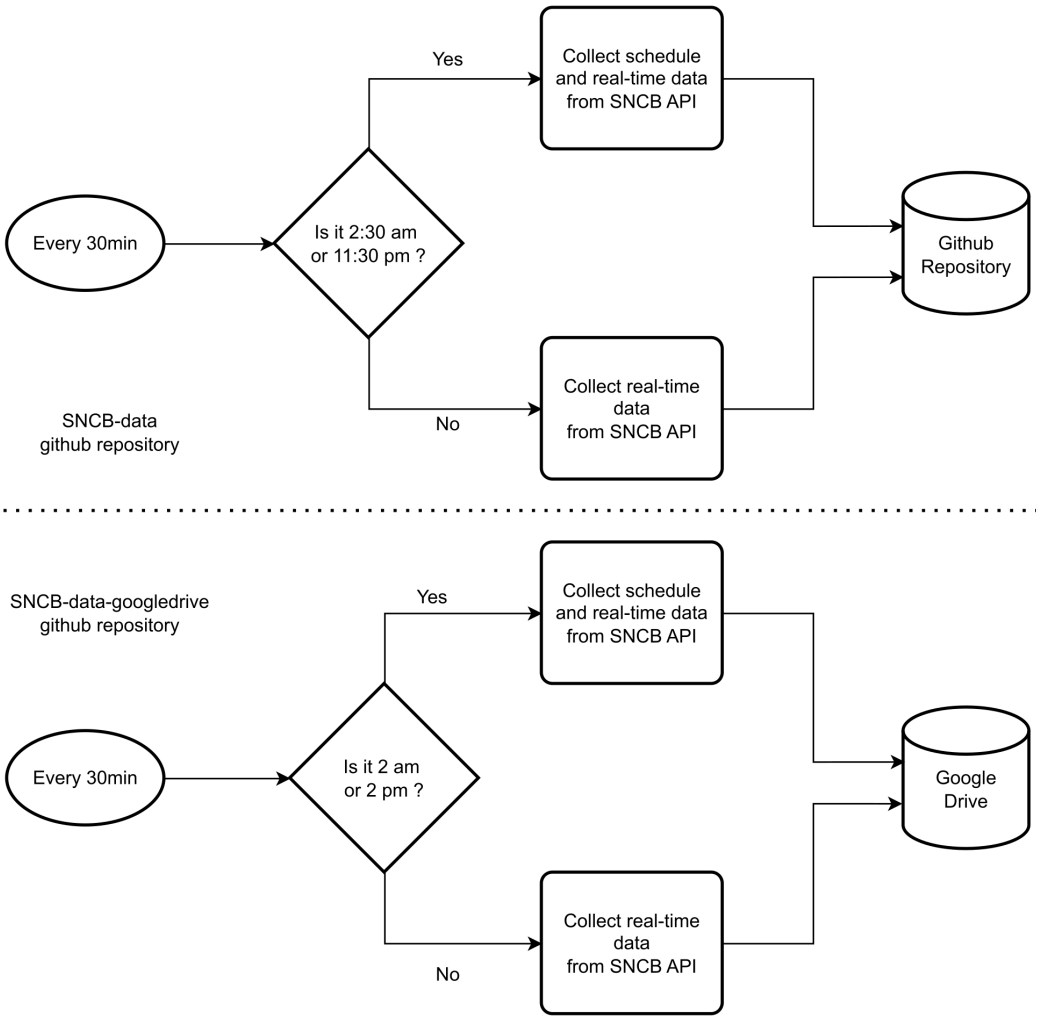


Figure 4.1: Data collection flowchart.

4.4 Data Description

4.4.1 Input Schedule Data

As a growing number of public transport operators, SNCB uses the GTFS format to store its scheduled information. The format is not strict, and SNCB adapted it. On the next page, figure 4.2 presents the data available from the schedule endpoint.

Figure 4.2: Received data from schedule endpoint API in Crow Foot notation.

4.4.2 Input Real-Time Data

The GTFS specification provides a format for real-time data. SNCB has also adapted it to its usage, and below is presented the general structure:

```
{
  "header": {
    "gtfsRealtimeVersion": "1.0",
    "incrementality": "FULL_DATASET",
    "timestamp": ...
  },
  "entity": [
    {
      "id": ...,
      "tripUpdate": {
        "trip": {
          "tripId": ...,
          "startTime": ...,
          "startDate": ...
        },
        "stopTimeUpdate": [
          {
            "arrival": {  // Not necessarily present
              "delay": 0,
              "time": "1701363120"
            },
            "departure": {  // Not necessarily present
              "delay": 0,
              "time": "1701363120"
            },
            "stopId": "8844545",
            "scheduleRelationship": "SKIPPED"   // Not necessarily present
          },
          ...
        ],
        "timestamp": ...
      },
      ...
    },
    ...
  ]
}
```

4.4.3 Database construction

For our analysis, we're interested in trip planning data. Therefore, data regarding translations or the agency are irrelevant in our case. Additionally, trip schedules can be viewed as a graph, and a better representation of the data has to be implemented to traverse such a graph efficiently.

Therefore, we adopted a new representation of the schedule data, greatly inspired by a previous work [2]. We constructed the following MongoDB Database:

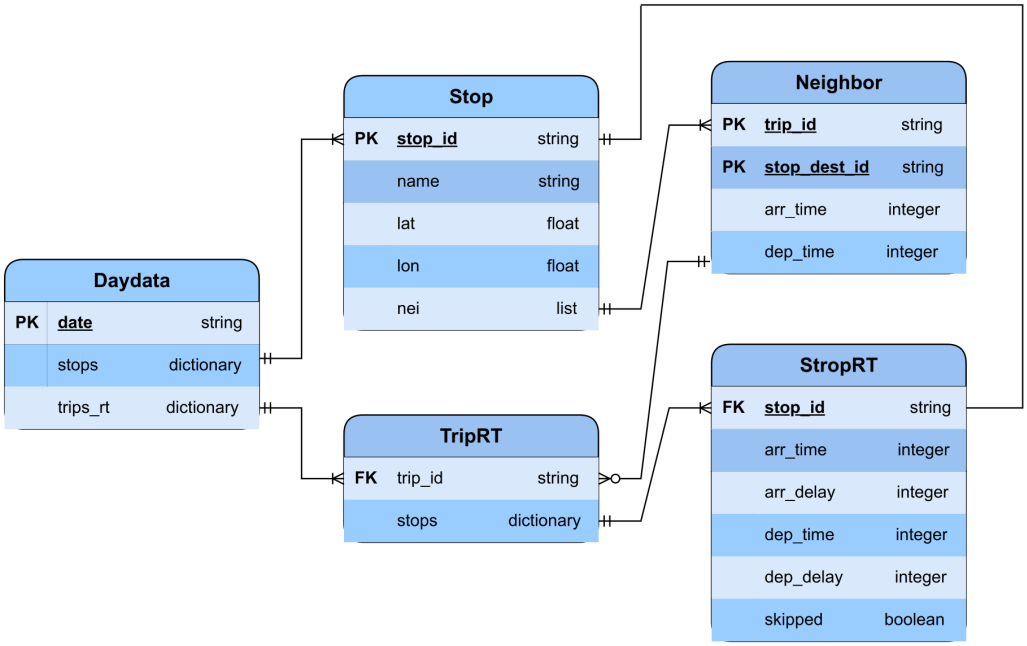


Figure 4.3: MongoDB database - Crow Foot Notation

This representation of the data enables the use of efficient search and traverse algorithms. Indeed, the representation is similar to an adjacency matrix because, from a stop, we can directly iterate over its neighboring stops.

An important remark is how a neighbor is defined. A neighbor is a stop directly reachable in one stop by at least one train from the current station. Therefore, the graph is directed.

[2] https://github.com/GuillaumeDerval/publictransportmap/blob/master/new_method/0bis_parse_gtfs.py

4.4.4 Data cleaning

SNCB's raw data is unsuitable for direct use. Notably, many delays only appear as 'None' values in the data. Another problem is that there is a discrepancy between planned trips, appearing in the `Stop` part of our database, and the trips really happening, for which we have data on their delays and are in the `TripRT` part of the database.

All 'None' values in the delays are considered to be zeroes. This is done for two reasons. First off, we don't want to unfairly worsen delay statistics since it isn't known why there are holes in the data. Secondly, zero is the most ubiquitous value for delays. Considering the discrepancies, trips appearing in the planned set but not the real set make up for about 5% of all trips. Two hypotheses were made to explain this phenomenon. They could either represent canceled trains, which were planned to travel but in the end did not because of various reasons (too much delay, accidents, weather conditions, ...). Or it could be possible that only trips in which there was at least one minute of delay at any point in the trip are present in the `trip_rt` dataset. In that case, all planned trips are considered to be on time by default. There are no trips which are in the `TripRT` dataset, which has a delay of zero for all stops in the data we observed. However, some trips have either a zero or a None value for all their delays. As stated before, we consider 'None' values to be zero. Therefore, the second hypothesis seems less probable. Furthermore, there is no field to represent canceled trips in the provided structure represented in figure 4.2.

Chapter 5

Delay Statistics

Previous chapter 4 discussed the process of gathering, storing, and cleaning data. This section will explore this data to gain further insights into the Belgian railway system. Furthermore, the impact of different delay definitions will be discussed.

5.1 Definition of a late train

Defining what a late train is seems like a simple task, but the answer is not as straightforward as it seems. Intuitively, it could be said that a train is late when it arrives at its next destination later than planned. However, SNCB's definition is different.

5.1.1 Intuitive definition

Before going into the definition, let's remind ourselves of two concepts. Firstly, a trip represents a single train going from a starting station to a terminus station at a specific time. It is represented by a sequence of 5-tuples containing the station ID, and real and expected arrival and leaving time to/from the station. The complete definition is defined in Definition 1. Secondly, a delay happens when there is a difference between the expected arrival or leaving time and the real arrival or leaving time, respectively. Formal definitions of arrival and leaving delays can be found in Definition 3 and Definition 4.

In the intuitive definition, a train is considered late when it arrives at a station delayed. The length of the delay does not matter in this case.

Consequently, a train can be late multiple times during the same trip. For a trip, the set of all stops at which the train is late is defined as follows:

Definition 5 *Let t be a trip as per definition 1 and $d^a_{t,s}$ be the arrival delay as per definition 3. Then, S^I_L is the set of all stops $(s, a^p, a^r, ...)$ at which the train is late under the intuitive definition.*

$$S^I_L = \{(s, a^p, a^r, ...) \mid (s, a^p, a^r, ...) \in t, d^a_{t,s} > 0\}$$

As stated before, only the relevant elements of the 5-tuple are present in the definition. In this case, the station ID and arrival times are used to compute the delay. The previous equation must be applied to all trips to find all the stops at which the train is late.

Definition 6 *Let T be the set of all trips t and S^I_L be the set of all late arrival stops. Then, T^I_L is the set of all late arrivals under the intuitive definition.*

$$T^I_L = \bigcup_{t \in T} S^I_L$$

5.1.2 SNCB's definition

When looking at Infrabel's punctuality statistics, they only consider a train to be late if it arrives 6 minutes or more after the planned time. Furthermore, they only consider train delays at their terminus station or Brussels Nord and Midi stations [1]. Using the notations introduced in chapter 2, a definition of the set of all late arrivals for a trip can be crafted. In this case, most sets will only be one stop long since only the terminus and two other stations are considered.

Definition 7 *Let t be a trip as per definition 1, s_t be the terminus station of said trip, s_m be Brussels-Midi stations, s_n be Brussels-Nord station, and $d^a_{t,s}$ be the arrival delay as per definition 3. Then, S^S_L is the set of all stops $(s, a^p, a^r, ...)$ at which the train arrives late in trip t, using SNCB's definition.*

$$S^S_L = \{(s, a^p, a^r, ...) \mid (s, a^p, a^r, ...) \in t, (d^a_{t,s} \geq 6) \wedge (s = s_t \vee s = s_m \vee s = s_n)\}$$

[1] https://opendata.infrabel.be/page/punctuality/

To find the set of all late arrivals under these conditions, unioning all trip-specific sets is necessary.

Definition 8 *Let T be the set of all trips t and S_L^S be the set of all late arrival stops. Then, T_L^S is the set of all late arrivals under SNCB's definition.*

$$T_L^S = \bigcup_{t \in T} S_L^S$$

Understandably, a train that is only one or two minutes late can hardly be considered late. It should not impact connections between trains or other public transportation. Moreover, most people won't even realize their train is technically a bit late. Infrabel's logic seems to be that a train is only considered late if it causes an inconvenience to passengers. On the other hand, it can also be questioned whether these little delays tend to get worse or are mitigated by reduced waiting times in the station or by a train traveling faster than usual.

An argument can be made that a delay happens when a train arrives at the station so late that it impacts connections with other trains, public transport, or passengers' planning. Considering this point of view, let's analyze the pertinence of SNCB's criterion.

5.1.3 Creating a new definition

We can begin with the limitation of looking only at delays at the terminus and at Brussels'Midi and Nord stations. It could be reasonable only to count delays at the last station of a trip if it was always or nearly always the worst part of the trip. However, this is often not the case. About 63.5% of all trips have a higher delay at some point during the trip than at their terminus. This means that by only counting late trains at the terminus, we do not consider many trains that could be inconvenient at the beginning of their travels. Next, we can see whether the limit of a minimum of six minutes of delay is reasonable. As stated before, a train could be considered late when its delay causes a passenger to miss their next train or bus. As of May 2024, if you want to go to Ostende from Brussels-Midi, you will have to change trains at Gand-Saint-Pierre station. You will only have five minutes to switch trains. If your train were five minutes late, you would miss the correspondence, but it wouldn't count as a delayed train in official statistics because the

delay is less than six minutes. Even a slight delay can cause you to miss your train in that situation. A study from 2016 on Beijing's subway system considers that passengers need at least ninety seconds to change trains [16]. This number can be rounded to two minutes since SNCB's data only deals with minutes. Since a minimum of five minutes is given to change trains and a minimum of two minutes is required to change trains, considering trains with more than three minutes of delays as late is a better representation than six.

A new definition of the late train set can now be created, taking into account the previous remarks. The delay at each stop is considered, and the train is considered late if it is at or higher than three minutes of delay. This means a train is considered late or on time at each stop instead of for each trip, as per the previous definition.

Definition 9 *Let t be a trip as per definition 1 and $d^a_{t,s}$ be the arrival delay as per definition 3. Then, S^N_L is the set of all stops $(s, a^p, a^r, ...)$ at which the train is late under the intuitive definition.*

$$S^N_L = \{(s, a^p, a^r, ...) \mid (s, a^p, a^r, ...) \in t, d^a_{t,s} \geq 3\}$$

To define the set of all late arrivals from the previous definition, a union of all sets of late stops at each trip t can be used.

Definition 10 *Let T be the set of all trips t and S^N_L be the set of all late arrival stops. Then, T^N_L is the set of all late arrivals under the intuitive definition.*

$$T^N_L = \bigcup_{t \in T} S^N_L$$

Now that all conditions have been defined and formalized, they can be compared to see their influence on delay statistics.

5.2 Comparison between definitions

First, it is important to consider the official numbers for train delays in Belgium. In March 2024, Infrabel, the Belgian railway infrastructure manager company, considered that 9.98% of trains were late [2].

[2] https://opendata.infrabel.be/page/punctuality/

This number is computed using SNCB's conditions defined in 8. When this statistic is computed using the data at our disposition for the same month, we obtain a result of 7.40% of late trains. As mentioned in section 4.4.4, there are holes in the data available to us, which explains the observed difference. When calculating these statistics, None values were considered as zeroes for the reasons mentioned in section 4.4.4. In that same section, it is also mentioned that there are some trips for which we don't have real-time data. It was hypothesized that these were canceled trains, but they might just be holes in the data. Or a mix of both. Those trips are not considered in our statistics or our models' training, but there might be delays in them, which the SNCB counts but we don't have access to. While this should be kept in mind when reading the following analysis, the percentages we obtain are still roughly the same as those of Infrabel. Instead of relying solely on data from March, our statistics utilize all available data from trips between December 2023 and March 2024. During this period, Infrabel's set of rules considers 8.36% of all trains late. On the other hand, using the second set of rules defined in definition 10, 26.34% of trains are classified as late. Using the intuitive rules defined in definition 6, a percentage of 61.72% of late trains is obtained. This shows how many small one to two-minute delays happen in the network.

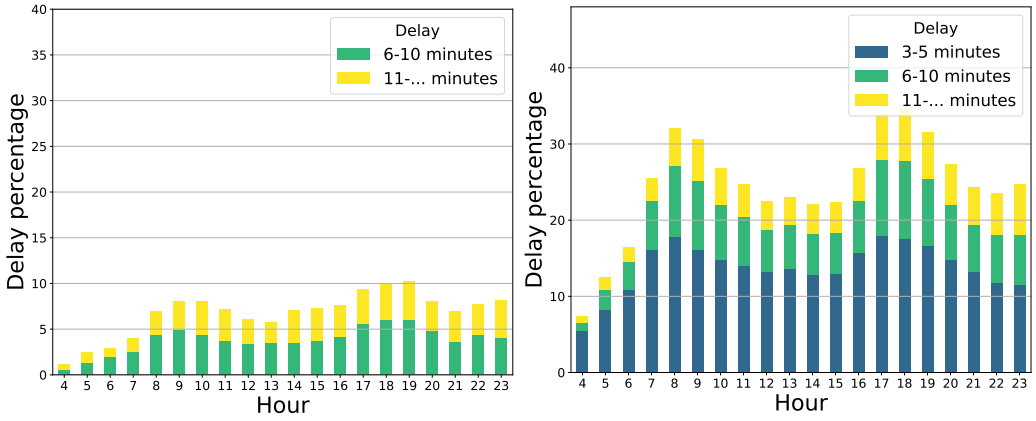


Figure 5.1: Late trains per hour according to SNCB's rules (left) and our rules (right). Data from 01/12/2023 to 31/03/2024. Divided into categories of delay.

Figure 5.1 compares the percentage of delayed trains per hour between SNCB's definition and ours. Even though the percentages are different, the tendencies are the same, with a peak between 7 and 9 am and between 5 and 7 pm.

5.3 Overview of the Belgian railway situation

The whole Belgian territory is not affected by train delays in the same way. Figure 5.2 shows delays of more than 5 minutes to better fit with the user impression of delays. Brussels, the north of the Hainaut area, the Limburg area, and, to a lesser extent, the Liège area, experiences larger delays for the four-month period from December 2023 to March 2024. Unless explicitly noted, the intuitive definition of a late train as defined by definition 6 is used for the following statistics. Therefore, trains are late when they arrive at a station later than planned, regardless of the delay.

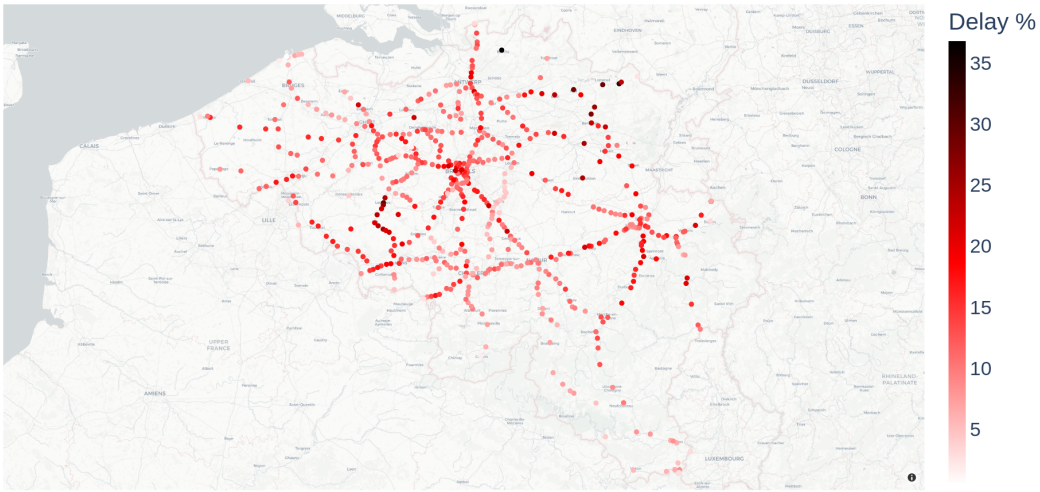


Figure 5.2: Average number of delays for the period December 2023 - March 2024.

When looking at all delays in figure 5.3, delay distribution is rather homogeneous. There isn't any correlation between the average delay time and the percentage of late trains. This means longer delays are not necessarily observed in stations with high delay rates. Stops with less

than 40% of trains delayed are as many as stops with over 60% of trains delayed. This is an interesting observation for the user experience. If most stations had low delay rates with a few exceptions, then most people would have a positive experience. But this is not the case; delays impact everyone. Most delays are between two and six minutes.

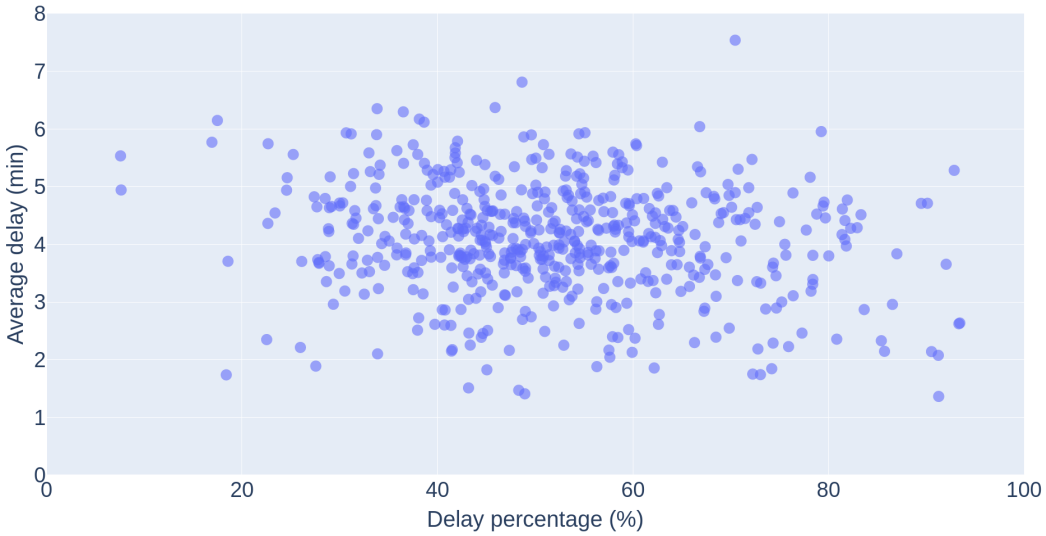


Figure 5.3: Average number of delays VS the average time delay for the period December 2023-March 2024.

Figure 5.4 offers a day-by-day delay percentage comparison between delays over and under five minutes. It can be seen that delay distribution has been periodic, with a period length of 7 days, over the four-month period spanning from December 2023 to March 2024. Around the middle of January, a rise in the delay rate can be observed. This corresponds to exceptional snowfalls. It is also remarkable that the percentage of delayed trains goes up during the week and down around the weekend. Indeed, fewer trains are late during the weekend.

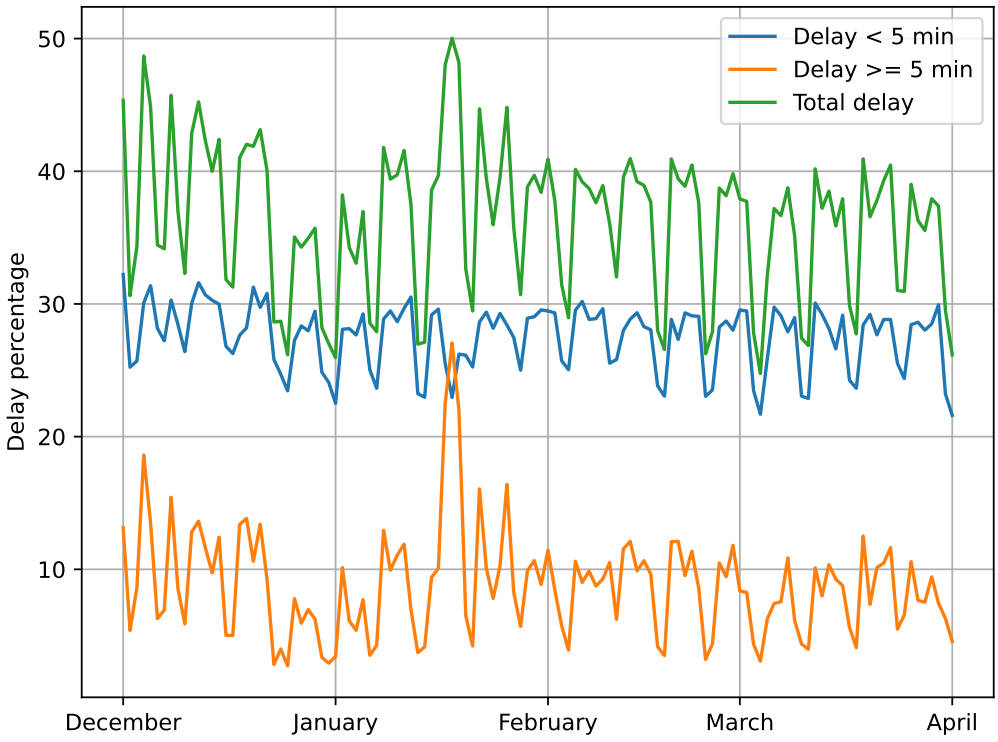


Figure 5.4: Evolution of the percentage of the delay in the Belgian railway per day between December 2023 and March 2024 included.

After train delays per day have been compared, they can be compared per hour. Figure 5.5 shows the time of day greatly influences the delay percentage. Peak hours range from 7 to 9 am and 5 to 7 pm. They can be correlated with the traditional working/teaching hours. It is reasonable to think that a higher passenger count translates to more delays under those circumstances. When only considering delays of at least 3 minutes, like in figure 5.1 on the right, the variance is more marked between hours. This is because the lower delay ratio tends to diminish during peak hours, which is especially visible when looking at delays of one and two minutes in figure 5.5. While the proportion of other delays gets higher, theirs gets lower. Thus, it can be said that during peak hours, not only does the percentage of delay increase, but a part of the shorter delays worsen.

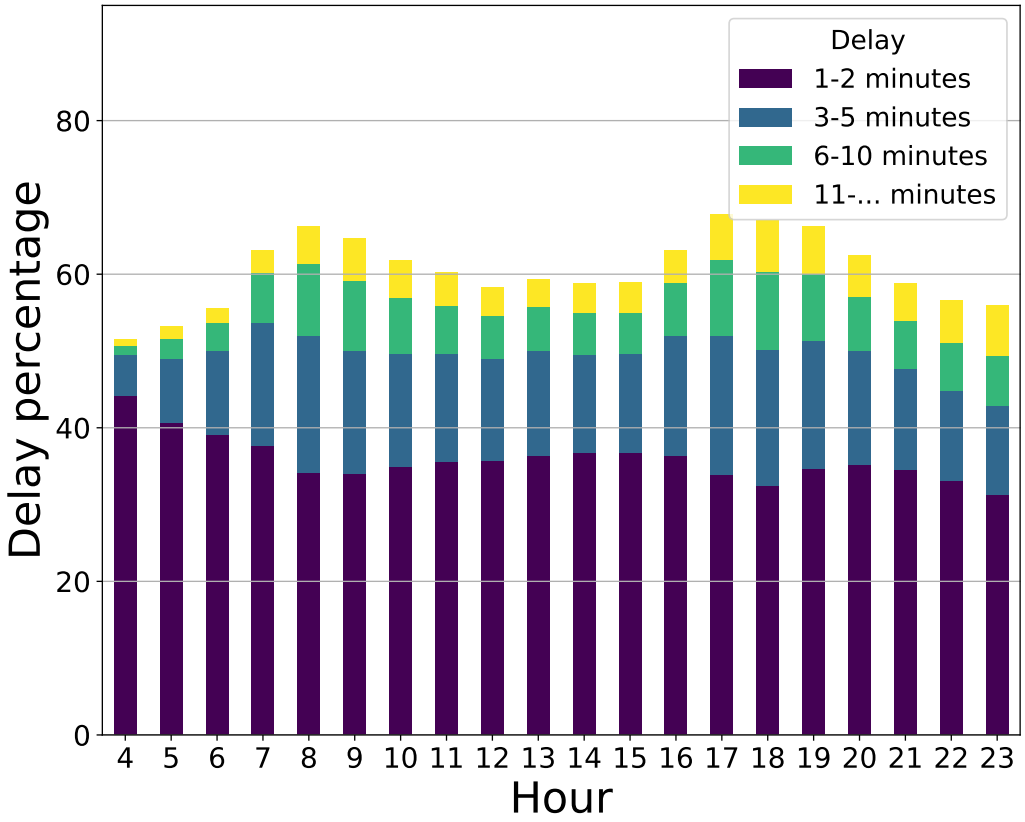


Figure 5.5: Late trains per hour according to intuitive rules divided into categories of delay. Data from 01/12/2023 to 31/03/2024.

5.3.1 Large and small stations

Some trains skip small stations to make the journey between large stations quicker. Before trying to train models, it is interesting to know if large stations (LS) and small stations (SS) influence delay differently. It might be interesting to consider only large stations to simplify the network.

Table 5.1 shows that smaller stations have lower average delays but a larger impact on increasing delays than larger stations. However, this difference in delay change comes from large stations being better balanced in their influence on delay change. When comparing strictly positive changes only to the delay in the same table delays tend to increase more in large stations than in small ones. It is also true that delays are reduced in large stations when only delay reduction is considered.

Mean	Arrival delay	Departure delay	Change in delay
SS all delays	2.21 (min)	2.40 (min)	0.16 (min)
LS all delays	2.47 (min)	2.53 (min)	0.11 (min)
SS positive delays	3.59 (min)	3.47 (min)	1.22 (min)
LS positive delays	4.27 (min)	3.80 (min)	1.53 (min)

Table 5.1: Comparison between small and large stations and between all delays and only positive delays in minutes during February 2024.

Furthermore, delay doesn't change between arrival and departure in small stations 64% of the time versus 51% for large stations. Therefore, large stations are more likely to experience delays and have a bigger impact on increasing delays.

5.4 Consequences of delay

Now that delays and late trains have been extensively discussed, what they are, and how often they happen, we can discuss their consequences. As seen before, most delays are less than ten minutes long. This can still be a manageable delay if you just have one train to take. However, even a shorter delay can be problematic when reaching a destination requires taking multiple trains or switching public transport. Fourteen thousand journeys were tested, all between faraway stations, with either one or two train changes happening each time. As a reminder, a journey is a person's path through the railway network. It is represented by a sequence of segments of trips (i.e. trains boarded on). For each of those journeys, at least two minutes are required between train changes, or there isn't enough time to switch trains. Under those conditions, at least one train change was missed for about **46%** of all those travels. The percentage of missed connections increases when journeys with more train changes are considered. It is at **58%** for journeys containing four train changes. Missing a connection greatly impacts the final journey length since the passenger needs to wait for the next train in the same direction.

Chapter 6

Predicting delays on the fly

In this chapter, previously collected data will be used and treated to explore different features and models for train delay prediction. Different network architectures are used for prediction. The first one is a Markov chain model, where we use previous results to predict further into the future. The second one is a Markov random field model.

6.1 Problem description

The objective is, given past departure delays of a running trip, to predict future departure delays of the studied trip. Given a running partial trip $\tilde{t}$:

$$\tilde{t} = \langle (s_1, a_1^p, l_1^p, a_1^r, l_1^r), ..., (s_m, a_m^p, l_m^p, a_m^r, l_m^r) \rangle$$

where a_j^p and l_j^p are the planned arrival and leave time at the station s_j, and a_j^r and l_j^r the real arrival and leave time at that station. The leaving delay can be defined as follows:

$$d_{t,s}^l = l_{t,s}^r - l_{t,s}^p$$

where t is the trip such that $\tilde{t} \subseteq t$. Let s_n be the terminus station of trip t. The models aim at predicting $d_{t,s_{m+i}}^l$ with $m + i \leq n$, $i \in \mathbb{N}$. Therefore, only partial trips, i.e. incomplete trips where the terminus has not been reached, are considered for predictions.

6.2 Feature selection

Studies have already been conducted on train delay prediction. In a 2018 work by Gaurav and Srivastava [9], a Markov chain model is used to predict delays of trains passing by a particular station in the Indian railway network. Our models are inspired by theirs but are built upon the entirety of SNCB's railway system. As such, an attempt was made to replicate the features used in the paper. These features can be classified into multiple categories.

First off, some of them refer to the train itself. Those are the type of train, the zone where the train operates, and the speed category of the train. However, we don't use any of those because, although they exist in the Indian network, they aren't present in the data available to us.

Then, some features are on the trip itself. Among these, are the distances between stations and the number of neighboring stations. The distance between two stations is computed from their geographic coordinates, which are present in the dataset. The geodesic distance is used to take the earth's curvature into account and be more precise. In our case, a station is a neighbor to another station if a railway path exists between them that does not cross any other station. In that category, we also find the stations' traffic strength and unique identifiers. The traffic strength is represented by the number of people passing through the station. It is not something that is available in SNCB's scheduling data. However, every year SNCB releases a document containing an estimation of the number of passengers in each station. This data is obtained by visual counting and is only done for a period of about ten days in early October. Although this method is prone to potentially large errors, we think it is a reasonable metric for evaluating traffic at a station. Indeed, the most important thing for the model is to be able to compare stations between themselves. Thus, even if the counting is imprecise as long as the proportions are relatively correct, it can be used.[1] Using identifiers to train a model can make sense if their numbers is low which was the case in `GAURAV & SRIVASTAVA`. In our case, with about 600 stations, it wasn't very interesting, so it was decided to remove it.

[1] https://www.belgiantrain.be/fr/about-sncb/enterprise/publications/travellers-counts

The last category of features concerns the ones on the trip itself. The month and the weekday are included in them. Including the month makes sense when available data spans two years, but with only four months of data, it was decided against because of the risk of overfitting the model. After choosing our features, the data needs a bit more cleaning.

6.2.1 Outlier removal

Train delays and their cause can vary greatly. Our models use the current and past state of the network to make predictions. Therefore, they cannot predict accidents, which can provoke large delays and are provoked by external factors outside the network. Considering this, the upper outliers in terms of delay were removed. It was done because they are most likely to be caused by unpredictable events. Outliers are delays higher than the mean delay plus three times the standard deviation over all delays in a station.

This means that the removal threshold is station-dependent. This is necessary to accommodate stations with wildly different average delays. For instance, in Bruxelles-Central, more than one of every ten late trains (following the intuitive definition 6) is late by ten or more minutes. In comparison, it's nearly one of every twenty trains in Jambes station. So the threshold needs to be adapted.

Tukey's rule was also considered for outlier removal. It works by finding out the interquartile range (IQR) defined as the difference between the first (Q_1) and the third (Q_3) quartile. A threshold can then be computed by adding the third quartile to the product of the IQR and a coefficient (k) [12]. Unfortunately, this method had a tendency to either remove every delay over ten minutes or not remove anything at all, depending on the coefficient.

6.3 Markov chain-based models

Markov chain-based models are structured as follows :

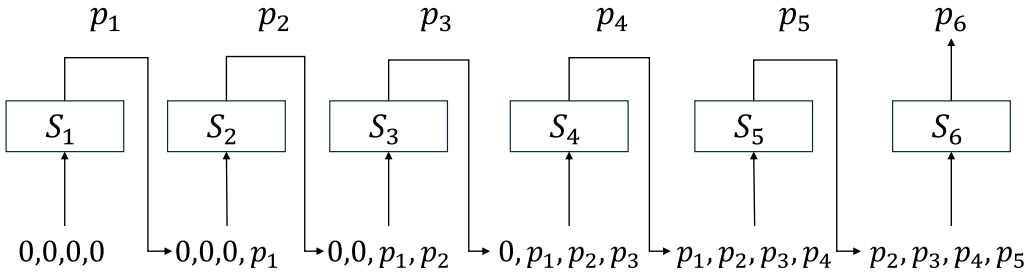


Figure 6.1: Structure of a Markov chain-based model

The S_i rectangles each represent a station on a journey. A model is associated with each of them. Each model predicts p_i, the delay at their attached station, using an input composed of the four previous delays and the other features mentioned in section 6.2. Those features are not shown in figure 6.1 for conciseness and clarity reasons. It follows the same architecture as `GAURAV & SRIVASTAVA` [9], where the delay at each station in a traveler's journey is modeled as a random variable in the chain. Remember that a journey is the path taken by a traveler through the railway network as defined in definition 2. Our goal is to find the delay at station 6. To do this, the model goes from station to station, each time predicting the delay at that station and using it to predict the delays at the next four stations. In a traditional Markov chain, a transition matrix would be used to find the next station's delay (see section 4.1). However, instead of building a transition matrix of five dimensions, a machine-learning model is built for every station. They predict the delay at their station using the four previous delays as well as the features described above for better performance. Predicted delays are used only if necessary. If real data is available it is used instead. When taking the train during the first few stations of a trip the previous delays don't exist since there aren't enough previous stations. So, in that case, they are considered null.

Now the question is what kind of models should be used. The previously mentioned paper uses a random forest and a ridge model, but more options could be explored. It was decided to test, in addition, a linear model as an alternative to the ridge model, a neural network, and decision and optimal trees. Optimal trees, as well as decision trees, are tested both as standalone and in a random forest. This allows us to compare the influence of a random forest for both types of trees. All models are trained using `sklearn` python library for better training performance.

This paper doesn't discuss fine-tuning the different models used inside Markov chains. Its goal is only to give a brief overview of the different possibilities and a first impression of their performances. Linear and ridge models use the default parameters from `sklearn.linear_model`. Neural networks are trained via the `MLPRegressor` class from `sklearn.neural_network`. They have been tested with various numbers of layers (2, 4, 7, 9, 11), with 50 neurons for each layer every time. Since there are about 600 stations in Belgium, there also are 600 models to train. To limit the training time, a max iteration limit has been set to 600 (unrelated to the number of models). The decision tree, from `sklearn.tree`, uses a mean absolute error criterion instead of a mean squared error one. This helps since the dataset is very unbalanced; the low delays far outnumber the big ones. A mean squared error criterion would give too much importance to the big delays. The random forest, from `sklearn.ensemble`, uses a 1000 tree with the same criterion as the decision tree. That number was chosen because the reference article implementation uses that number too [9].

PYDL8.5's implementation of optimal trees is used with the `DL85Predictor` model [1]. Optimal trees require binary input to work. Distances between stations and traffic strength are both binarized into five categories each. The thresholds have been chosen to separate data into quintiles. Delays are separated into categories depending on their importance: no delay, up to two minutes, up to five, up to ten, up to twenty, and above twenty. Finally, the degree strength containing values mostly between zero and seven has been separated into eight categories. Ranging from zero to seven and ending with everything above. Other features were already in binary form. For the same reasons as explained above, optimal trees use mean absolute error as an optimizing criterion. Moreover, for training time constraints, their maximum

depth is limited to three and their training time to five minutes. Random forests with optimal trees use the same configuration for their trees, except the time limit has been reduced to 15 seconds since more needs to be trained. Thirty trees are used in the forest, once again, training time constraints are a limiting factor.

6.4 Markov random field

The idea of using a Markov random field as a prediction model came in the process of finding an explanatory model. The first intuition was to use a Bayesian model, inspired by the article [25]. This Bayesian network offers a compelling representation of the network and intuitive metrics that can be drawn from it, such as the later explained *inducer* and *susceptible* metrics 6.5.

Unfortunately, the SNCB railway network structure is much more complex than the networks presented in the literature for Bayesian models. The SNCB network comprises numerous cycles, making it a poor fit for a directed graph. The objective is to make delay predictions for any possible trip, not only for a given corridor with no cycle, as observed in the literature. This makes the Markov random field a more appealing choice than a Bayesian network. In addition, no relevant paper applying Markov random field to the train delay problem could be found, making it interesting to explore.

6.4.1 Model construction

The Markov random field (MRF) is defined as follows: Let $G = (N, E)$ be an undirected graph, where $N = n_1, n_2, ..., n_i$ is the set of node, $E = e_1, e_2, ..., e_j$ is the set of edges, $i, j \in \mathbb{N}$. Each node represents a random variable corresponding to the leaving (departure) delay at the associated train station. The leaving delay at a given station s and for a specific trip t is defined as:

$$d^l_{t,s} = l^r_{t,s} - l^p_{t,s}$$

With $l^r_{t,s}$ be the real leaving time and $l^p_{t,s}$ be the planned leaving time 4. In order to keep the model simple and interpretable, only adjacent pairs of stations are considered. This way, each pair of stations has its own

function. Markov random field theory allows for more general functions covering all cliques, i.e. fully connected subgraphs, but we have decided only to consider pairwise cliques. This enables us to model the impact of a train leaving station A to join station B. In such a scenario, a relationship, i.e., a function, exists between the two stations. In a clique of three stations, the influence of a station on the two others does not correspond to the physical reality. Each station influences its neighboring stations because of the train linking them. Before defining the function, the domain has to be defined. The dataset only contains discrete values, so it was decided to give the solver discrete delays. Therefore, using natural numbers from 0 to 60 included, corresponding to minutes. This range of values covers the vast majority of delays. Any delay above 60 will be capped at 60. As a result, each function can be defined as follows:

$$\phi : \{0, 1, 2, \ldots, 60\} \times \{0, 1, 2, \ldots, 60\} \to [0, 1]$$

where $\phi(x, y)$ represents a probability value and $x, y \in \mathbb{N} \cap [0, 60]$.

Every function $\phi(x, y)$ can be represented as a 2D table, where each entry i, j corresponds to a pair of value (x, y). Therefore, constructing this table based on observed values is possible. For a pair of stations s_1, s_2, no matter the order, as the graph is undirected, both $d^l_{t,s1}$ and $d^l_{t,s2}$ are taken into account for the associated function.

It is easy to compute the probability associated with each pair of values from empirical data. However, before concluding this subsection, one last problem we should address is the case when the probability is null. In the data, it is possible that delays (5, 50) do not appear as it would imply that either a delay of 45 minutes has been created or absorbed. It is very unlikely but probably not impossible. The presence of null values makes a lot of "potential worlds" unfeasible.

In order to work this out, for a probability of an outcome x_i, Laplace smoothing was used with a constant equal to 1:

$$P(x_i) = \frac{\text{count}(x_i) + 1}{N + n}$$

where count(x_i) is the number of occurrences of the outcome x_i, N is the total number of observations, and n is the number of possible outcomes (61 in our case).

6.4.2 Maximum A Posteriori (MAP) Inference in MRF

Given a Markov Random Field (MRF), the MAP Inference problem is to find the most likely assignment of values to the unknown random variables given a set of evidence.

Let $G = (N, E)$ be the MRF associated undirected graph. Joint probabilities $\phi_{ij}(N_i, N_j)$ for each edge $(i, j) \in E$ are defined as:

$$\phi_{ij} : \{0, 1, 2, \ldots, 60\} \times \{0, 1, 2, \ldots, 60\} \to [0, 1]$$

In order to make an inference, a set of evidence has to be provided. Let a set of evidence $E = \{(N_i = n_i)\}$, and each n_i corresponds to an evidence value in $\{0, 1, 2, \ldots, 60\}$.

The objective is to find the assignment of $X = N \setminus E$, which maximizes the marginal maximal a posterior. More formally:

$$N^* = \arg\max_X P(X \mid E)$$

where N^* is the assignment that maximizes the posterior distribution. The choice of the evidence plays a crucial role. Different approaches are explored in the following subsection.

6.4.3 Constructing evidence

Given a running trip, the model's goal is to predict the departure delay for the upcoming stations. So given a running trip $\tilde{t}$:

$$\tilde{t} = \langle (s_1, a^p_1, l^p_1, a^r_1, l^r_1), ..., (s_m, a^p_m, l^p_m, a^r_m, l^r_m) \rangle$$

where $\tilde{t} \subseteq t$, s_j is the j^{th} station in that trip, a^p_j and l^p_j the planned arrival and leave time at that station, and a^r_j and l^r_j the real arrival and leave time at that station. Let n be the number of stations in that trip with $m < n$.

The model will predict $d^l_{t,s_{m+1}} = (l^r_{t,s_{m+1}} - l^p_{t,s_{m+1}}), d^l_{t,s_{m+2}}, d^l_{t,s_{m+3}}$, and $d^l_{t,s_{m+4}}$. A prediction is only made if $m + j \leq n$, with $j \in \{0, 1, 2, 3\}$.

The Markov random field model is flexible in the sense that it only requires one assignment to perform an inference on the entire graph. The past delays for the running trips $\tilde{t}$ are given as evidence. The other stations that do not belong to the trip t are dealt with according to three different scenarios.

Scenario 1

The first scenario assigns every station that is not part of the trip t a delay value of zero. The scenario is meant to evaluate whether the other station variables impact the quality of the prediction.

Scenario 2

The second scenario assigns each station that is not part of the trip t its median value. A median value is neutral, and the goal again is to assess whether this improves the quality of the prediction.

Scenario 3

In the last scenario, the objective is to construct a model as close to reality as possible. Every station that is not part of the trip t is assigned its average leaving delay computed on the last integer hour. For instance, if the inference is for 9:40, the average delays between 8 and 9 will be assigned to each station that is not part of t.

The objective is to analyze whether data close to real-time data improves the model's performance. And, if this is the case, are they significant?

6.5 Inducer and Susceptible metrics

Two interesting metrics have been developed for Bayesian networks: *inducer* and *susceptible* metrics[25].

The delay-*inducer* score captures a stop impact on the general network. The impact in terms of delay, assessing how this particular stop can propagate its delay through the entire network. Each delay propagated is weighted by the number of trains impacted by the prediction.

On the other hand, the delay-*susceptible* score measures how sensitive a stop is to the delay from its neighboring stops. The formula for the *susceptible* score has been modified compared to the original one from the article to fit an undirected graph better. In the original formula, the score is computed by not only considering neighbors, but every station in the graph too. It is reasonable in a Bayesian network to compute only the impact of parent stations. However, this parent-child relationship

disappears in an undirected graph. Hence, the choice of only considering the impact of the neighbors. The score of a station is weighted by the number of trains that go through that station.

$$\textbf{Inducer}_s = \sum_{j \neq s}^{n} \{E(X_j \mid X_s = x_s) * V_j\} / n \text{ where } \forall X_{i \neq s} \in \mathbf{N}\ i = 1, 2, \ldots, n$$

$$\textbf{Susceptible}_s = \sum_{j \neq s}^{m} \{E(X_s \mid X_j = x_j)\} * V_s / n \text{ where } i = 1, 2, \ldots, n,$$
$$\text{and } X_j \in \text{neighbors(s)} = \{X_1, X_2, \ldots, X_m\} \subset \mathbf{N}$$

where X_i is the delay of stop i, V_i is the delay number of trips to/from stop i, n is the total number of stops, x is the median delay value of variable x. **N** corresponds to the node variable of the MRF, i.e., the station delays.

Both metrics have been applied to a Markov chain-1 and Markov random field. For the Markov chain-1, the *inducer* score definition is ambiguous. A delay can be propagated through the network, using each station predicting model to propagate the delay to the unexplored neighbors, but some ambiguity occurs in the following situation:

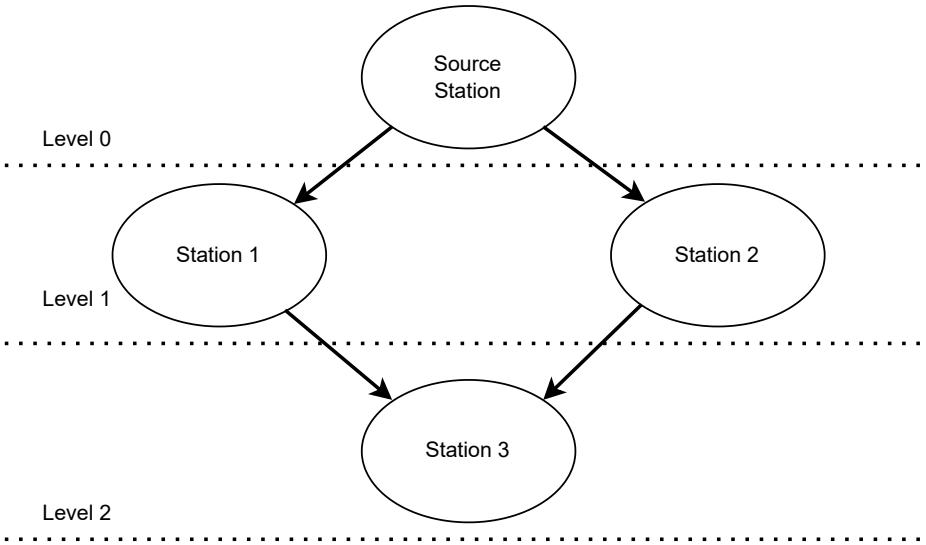


Figure 6.2: Delay propagation from the source

In that scenario, Station 1 and 2 will propagate their delay using their own predicting model to Station 3, and Station 3 will take the average as input. It can be generalized to more stations. A station will take as input the average of the propagated delay from the previous level, where a level corresponds to the distance of a station from the source station.

Regarding the Markov random field, the metrics are derived more naturally. With a median delay assigned to the source station, the propagation is inferred by the *argmax* inference. In order to retrieve the *inducer* score, every prediction has to be weighed by the number of passing through trains. For the *susceptible* score, one at a time, a neighboring station is assigned its median delay, and the most likely value for the source station is derived. The sum result is then weighted by the number of passing trains through the source station.

Chapter 7

Model Evaluation

Before being used for training models, a part of the data must also be set apart and not used in any way before testing. This is the test set. It will be used to evaluate the performance of the models on unseen data. It is important that the test set is representative of real data. For instance, as explained in section 5, days of the week and hours influence delays. Thus, trips must be well distributed over time. To do that, 15% of all trips are put aside randomly from the dataset. Since there is a large number of trips in the dataset (about 3.5 million) that are well distributed over four months, the law of large numbers should guarantee that the 15% are well distributed too.

7.1 Models performance

To test Markov chains-based models, each trip from the test set is taken, and five delay predictions are made. The first one uses real previous delays as inputs. Then, the second prediction is made on the station right after the first one, and the first prediction and the three next real delays are used as input. This cycle continues with each prediction using one less real delay and one more predicted delay each time. Until the last prediction is made only using predicted delays. This test suite aims to estimate how well the models predict far into the future. A second performance test compares the inference speed of the different implementations. Since one of the objectives of the models is to be used in a web application open to the public, the inference speed is important.

The solver used to perform inferences on the Markov random field is called *Toulbar* and is available on GitHub [1]. It is an open-source black-box C++ optimizer for cost function networks and discrete additive graphical models. As of May 2024, the latest binary executable was used for this work.

For Markov random fields, a unique stop, $s0$, for each trip in the test set is randomly picked to be the first stop used for prediction. The following stops $s1$, $s2$, and $s3$ are also used to predict the train leaving time delay, i.e., $d^l_{t,s}$ with t being the selected trip and s the associated stop.

7.1.1 Performance metrics

The main performance metric will be the mean absolute error (MAE). Since the data is very unbalanced and contains many more low delays (0 to 2 minutes), using the mean squared error (MSE) favors models that overestimate the delay to reduce their error in case of a very large delay. Even though all used models are regression models, not all errors are equivalent. This mean absolute error is computed based on predicted delays in minutes. As stated in section 5.1, under the new definition, a train is only considered late if it has three or more minutes of delay. From this basis, four categories of delays are established. From **zero to two** minutes are the ones that don't matter much from a traveler's viewpoint. Delays from **three to five** minutes are problematic if you have to take a connection but won't be a great annoyance if you don't. From **six to ten** minutes are noticeable delays even if you don't take a connection. Finally, **above ten** minutes are long delays that will always be inconvenient. All predicted and real delays are classified into those four categories, and classification metrics can then be used. In particular, the accuracy, which is the fraction of correct classification.

7.1.2 Prediction performance

The following tables compare each model using the metrics mentioned above. The best score for each metric is indicated in bold. Markov random fields are named "MRF" followed by a number indicating the

[1] https://github.com/toulbar2/toulbar2

scenario used for the model. For Markov chain-based models, only the type of model used to approximate the transition matrices is written. For those models, all four previous delays used as input are taken from the real delays in the dataset.

Score	Linear	Ridge	Neural(2)	Neural(4)	Neural(7)
MAE	0.725	0.725	2.559	1.193	0.926
accuracy	0.759	0.759	0.481	0.669	0.748

Score	Neural(9)	Neural(11)	Decision Tree	Random Forest
MAE	0.910	1.043	**0.699**	0.741
accuracy	0.779	0.748	**0.859**	0.840

Score	Optimal tree	Optimal forest	MRF 1	MRF 2	MRF 3
MAE	1.791	1.841	1.721	1.743	1.774
accuracy	0.489	0.536	0.777	0.776	0.749

Table 7.1: Models performance. The four previous delays are taken from the dataset for Markov chain-based models.

From table 7.1, it can be seen that overall the Markov chain-based models perform better than the Markov random fields models. In the Markov chain models, the simple ones perform better when the previous delays are the real ones. The linear, ridge, and decision tree models have the lowest MAEs. However, the accuracy of the random forest and neural network with nine layers is better than that of the linear and ridge models. Overall, the decision tree is the best model for this situation. The following table breaks down the accuracy for each class on the decision tree Markov chain-based model and the three Markov random field models.

Category	**MC - DT (%)**	**MRF 1 (%)**	**MRF 2 (%)**	**MRF 3 (%)**
0-2	93.3	99.2	99.2	93.7
3-5	65.2	9.1	9.1	18.9
6-10	70.3	8.2	7.6	12.3
11+	72.3	5.5	5.3	5.7
Aggr. Acc.	**85.7**	**77.7**	**77.6**	**74.9**

Table 7.2: Comparison of Model Accuracy

Table 7.2 shows the problem Markov random fields face quite well. The model predicts a delay of zero way too much. It has a significant bias, which is only mitigated a little in MRF 3. But overall, the accuracy of MFR is quite disastrous compared to what the Markov Chain can achieve.

7.2 Markov Chains

This section will focus on the performances specific to the Markv chain-based model. The following table shows the same Markov chain-based models as in the previous section but with all four previous delays being predicted.

Score	Linear	Ridge	Neural(2)	Neural(4)	Neural(7)
MAE	1.647	1.542	4.551	1.784	1.525
accuracy	0.564	0.564	0.323	0.594	0.659

Score	Neural(9)	Neural(11)	Decision Tree	Random Forest
MAE	1.510	1.542	1.667	**1.505**
accuracy	0.679	0.678	0.698	**0.708**

Score	Optimal tree	Optimal forest
MAE	1.934	1.955
accuracy	0.501	0.502

Table 7.3: Markov chains-based models performance when the four previous delays are previous outputs.

As seen in table 7.3, when the models use four predicted delays in their input, their overall performance decreases. An increase of about 62% in combined MAE is observed between the two tables, ignoring the MRF models. This can be explained by the fact that predicted delays can be incorrect, and those errors will spread through the chain and mislead the next models. In this situation, simpler models that were better in the previous situation fall off in favor of the random forest and the nine-layered neural network. These models seem more robust to errors and generalize better than the simpler ones. In the case of the random forests, this is not surprising as they have been created to solve this very problem [11].

Using inducer and susceptible scores

The following section 7.3 presents a model based on Markov random fields. In that context, two new scores are produced. Each station gets an inducer and a susceptible score. These scores could be added as features to the models. This adds ten features, two scores for each of the stations in the history and two for the target station. The following table shows the performance of models trained with these new features.

Score	Linear	Ridge	Neural(2)	Neural(4)	Neural(7)
MAE	1.652	1.543	3.288	1.864	1.553
accuracy	0.553	0.555	0.325	0.545	0.666

Score	Neural(9)	Neural(11)	Decision Tree	Random Forest
MAE	1.526	**1.505**	1.705	1.595
accuracy	**0.697**	0.686	0.690	0.673

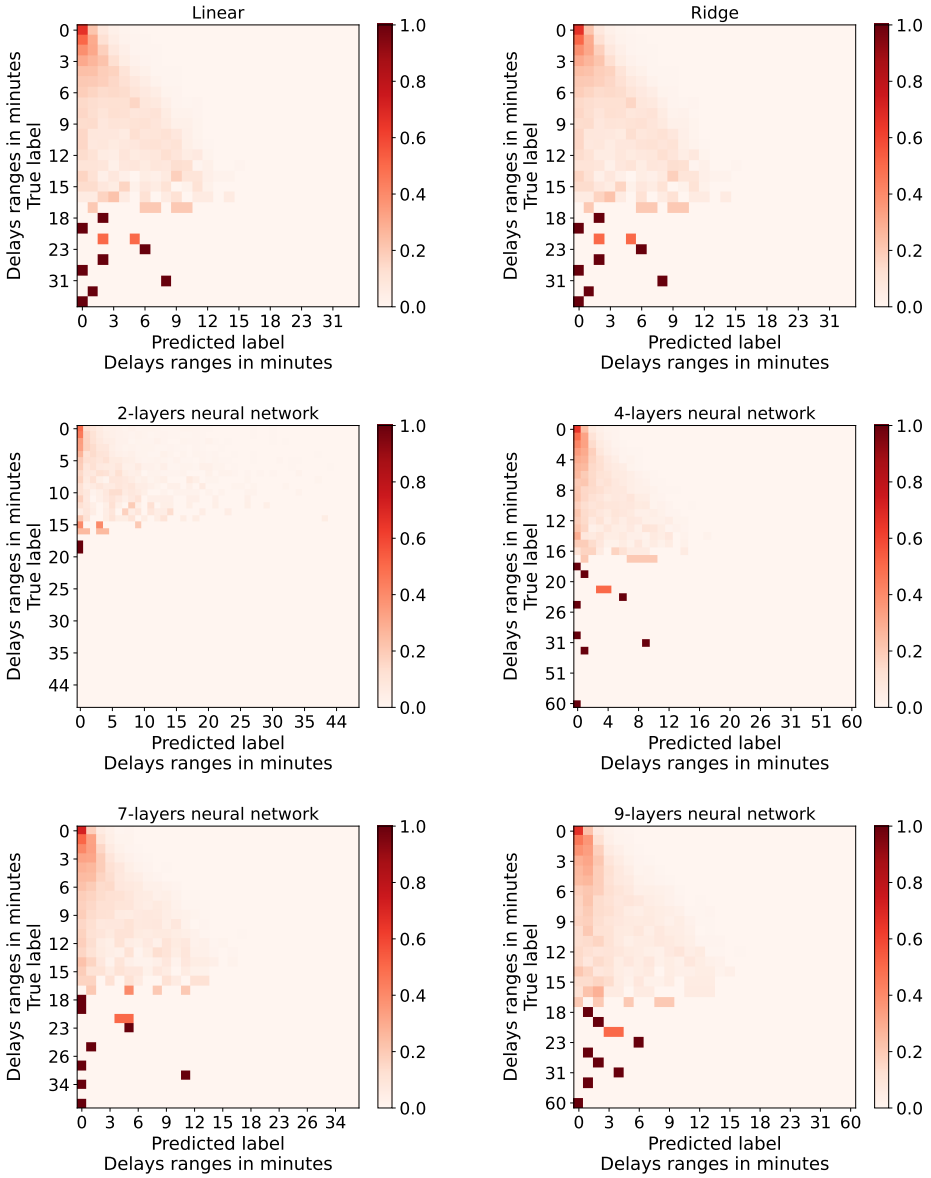
Score	Optimal Tree	Optimal forest
MAE	1.903	1.915
accuracy	0.502	0.522

Table 7.4: Markov chains-based models performance with new features, when the four previous delays come from the predictions.

Table 7.4 shows that these new features impact the different models differently. In particular, the neural networks with two and eleven layers have a better MAE. The optimal tree and optimal forest also perform better overall. However, the other models don't perform as well as before. The improvement in MAE but not in accuracy of the two-layered neural model happens because this new iteration of the model predicts a delay of zero much more often. While it tended to predict too high values before. However, it is still the worst model available by a margin. The model that most benefits from the change is the 11-layered neural network. This seems to indicate that the dependence between the new metrics and the delay is complex and necessitates a more complex model to be useful. For the other models, adding ten new features can cause overfitting and reduce performance. This last problem could potentially be solved with more data to help the model discover dependencies and allow the construction of larger models.

Source of errors

It is interesting to know where the models make mistakes to gather hints on how to improve them. Confusion matrices will be used to compare the predicted delays against the real delays to find the source of errors. For better clarity, the confusion matrices will be represented as heat maps. All compared predictions have been made using four previous predicted delays because there are more errors in that situation.



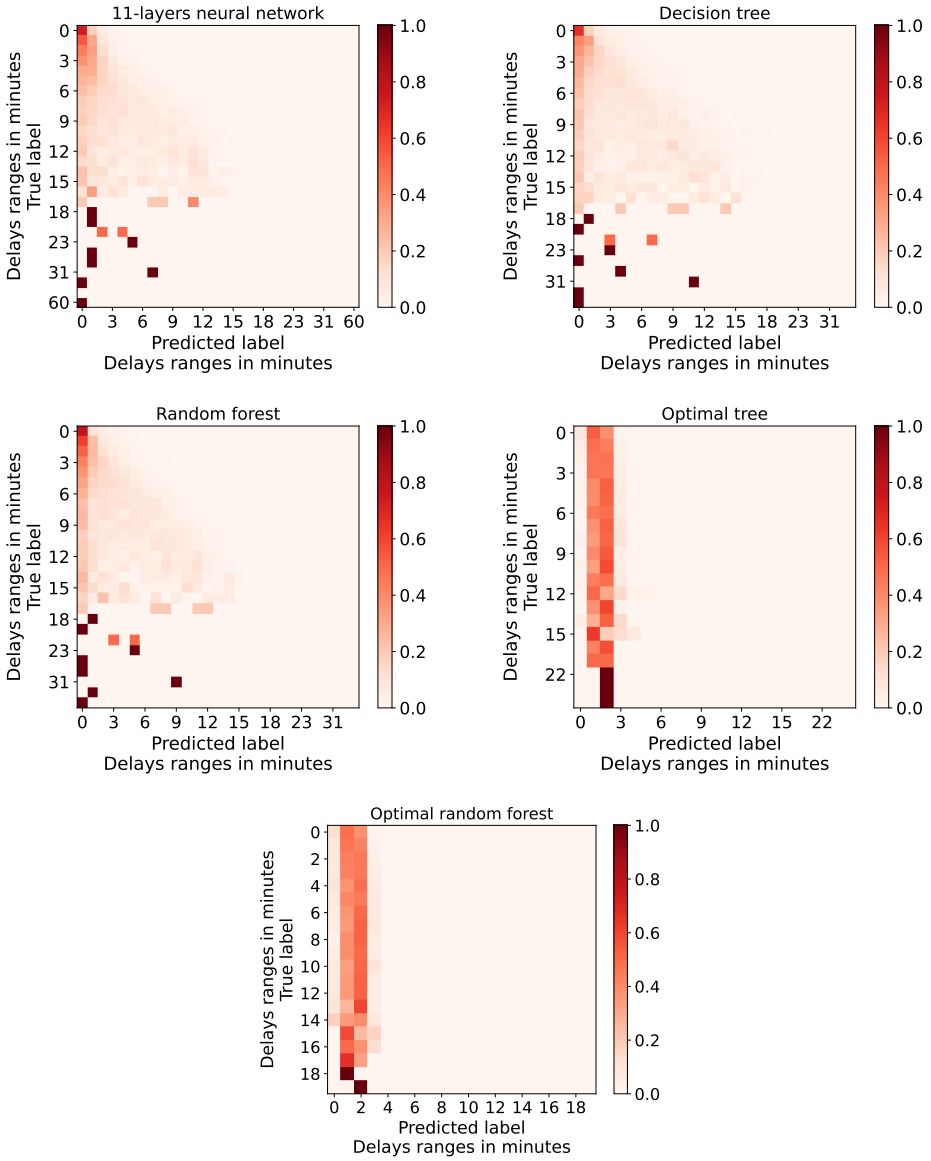


Figure 7.1: Heat map representation of confusion matrices for all Markov chains-based model.

Figure 7.1 shows that all models tend to predict values that are lower than the real values. This is a consequence of the very unbalanced dataset. Since there isn't a lot of data on larger delays, the models have trouble recognizing them. Another visible consequence is that every graph has much more concentrated values for large delays. For instance, it seems that the linear model always predicts 8 minutes when

the real delay is 31. The reason is that there are very few examples of large delays. So, there is only one instance of a 31-minute delay in the test set.

In both situations, the optimal tree model and optimal forest perform poorly. The delay prediction distribution explains why. Both the optimal tree and optimal forest nearly always predict low delays, whatever the real delay is. Two factors play a part in this: the depth of trees is limited to three, so the number of possible outputs is limited to eight, and the dataset is very imbalanced. Together, those factors make it so there is a risk that all eight possible outputs will be very low. To counter this problem, a simple solution that does not require increasing the depth and thus greatly increase the training time is to keep outliers in the dataset. The idea is that having higher values will help the model understand that it needs to make higher predictions. When training and testing on unfiltered datasets, the MAE drops from 1.791 to 1.363 with previous delays taken from the dataset. The accuracy also jumps to 0.793 from 0.489. When it takes predicted delays as input, the MAE becomes 2.306, worse than the trees trained on outlier-free data. However, the accuracy is better at 0.687 against 0.501. Similarly, the performance is much better if the optimal random forest is trained on data containing outliers. It performs better than the single tree with an MAE of 2.139 and an accuracy of 0.706 when taking predicted delays as input. In both cases, the MAE is worse when considering outliers, while the accuracy is better. This difference between accuracy and MAE can be explained by looking at the new confusion matrices.

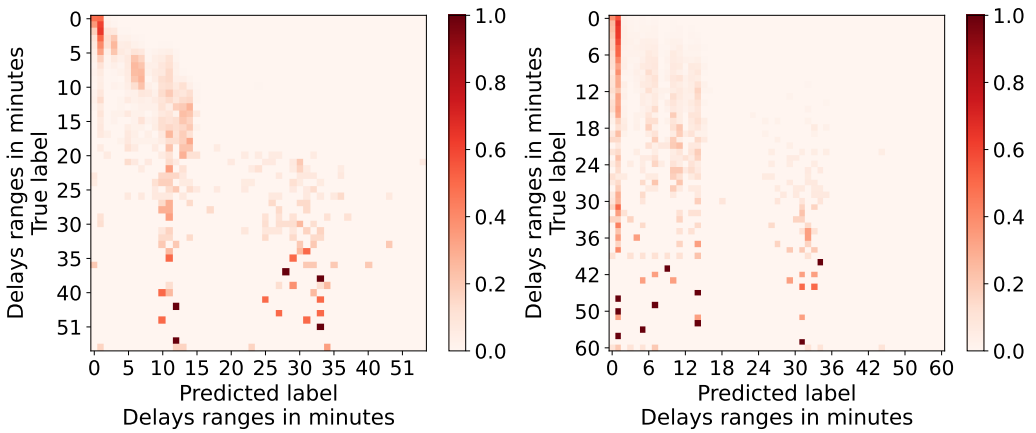


Figure 7.2: Confusion matrices of the optimal tree Markov chain-based model trained without removing outliers. Using 4 real delays on the left and 4 predicted delays on the right.

When outliers are not removed, the models predict high values much more often, see figure 7.2. This helps the accuracy by predicting large delays correctly more often. However, it also produces larger errors by overestimating low delays. Another reason is that the unfiltered datasets contain large, unpredictable delays, increasing the overall error. This is the trade-off of using outliers for training and why they were removed for other models.

7.2.1 Inference speed

The inference time is the time a model takes to make a prediction. All the models were tested on the same workload as in the previous section. They were executed on a 12-core Intel SkyLake 5118 processor at 2.3 GHz.

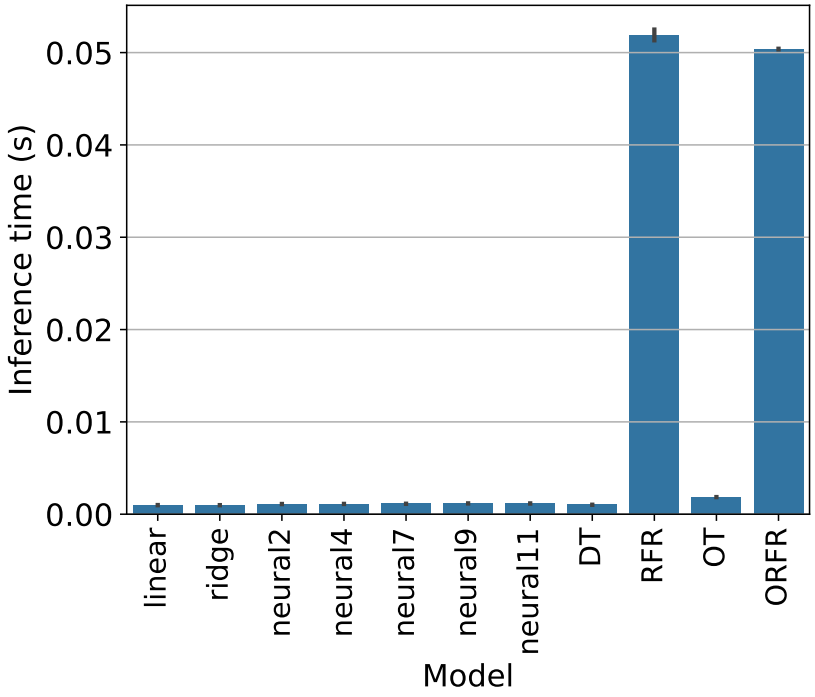


Figure 7.3: Inference time in seconds for each model.

Figure 7.3 shows that both random forest (RFR and ORFR) models have a much larger inference time than their competitors. This is explained by how random forests, and ensemble models in general, work. They train many smaller models on subsets of the training data and features and then poll them to make predictions. This necessitates inferring as many times as there are models. In our case, 1000 times for the RFR and 30 times for the ORFR. The optimal tree (OT) is also slower than the other non-forest models. Our explanation is that to do regression on optimal trees implemented in PYDL8.5 (see section 2.2) two functions used to calculate regression error and leaf score need to be implemented. Our implementation was done in Python, which slows down PYDL8.5, which was implemented in C++ [8].

7.3 Markov random field

Continuing with the Markov random field (MRF). The training process consists of learning the different pairwise joint distributions as explained

in section 6.4.1. The outliers have not been discarded as they are capped at 60 minutes. There is no learning cost function to optimize which could suffer from the outliers. As many of the trips are not late, the model would be penalized not to learn from those delays.

In the following figure 7.4, the model prediction MAE (Mean Absolute Error) is not consistent over time. There are some clear outliers in the middle of January, which are caused by exceptional weather conditions due to heavy snowfalls.

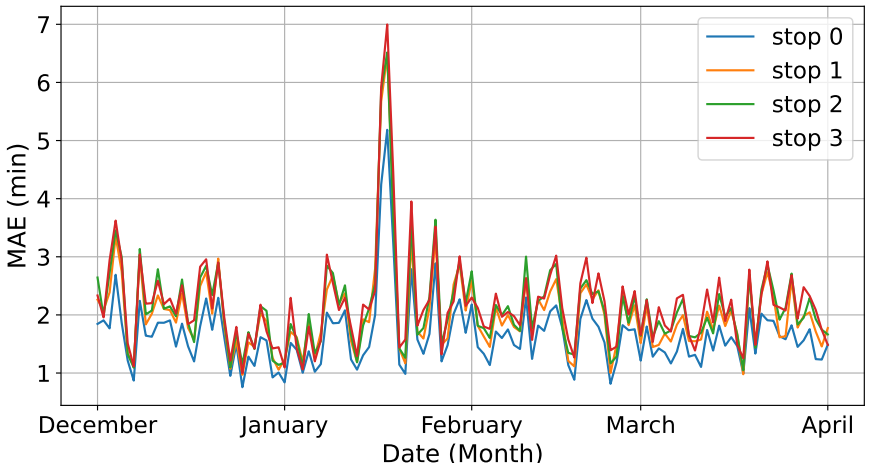


Figure 7.4: MAE over time (Scenario 1)

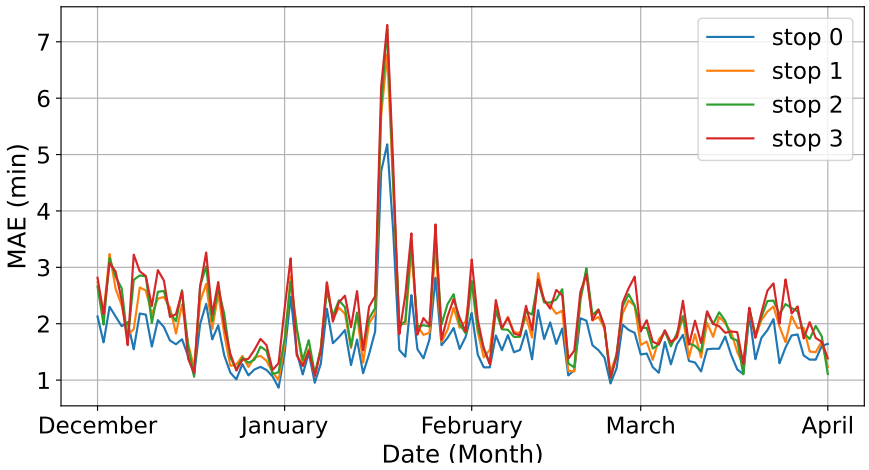


Figure 7.5: MAE over time (Scenario 2)

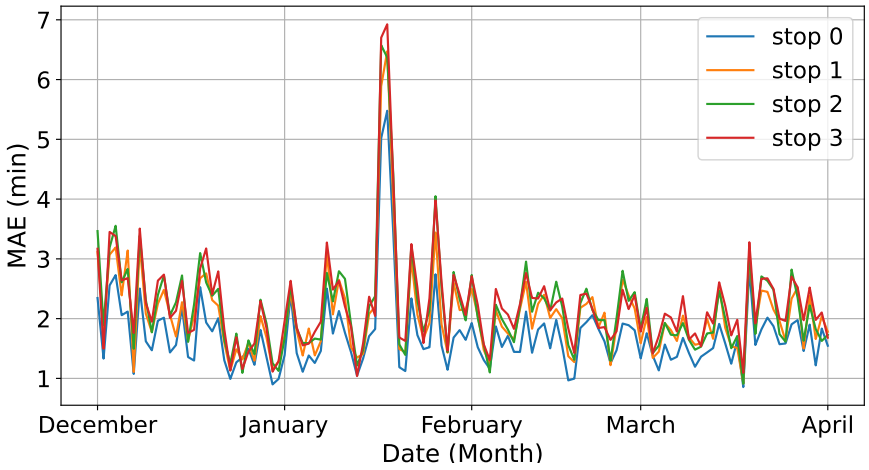


Figure 7.6: MAE over time (Scenario 3)

When looking at the following table 7.5, we have confirmation that the results are not that compelling compared to the Markov chain model. The best scenario is the first and falls short of the performance achieved by Markov chains regarding MAE 7.1. Unlike in the previous section, the accuracy in the following tables corresponds to accurately predicting the exact integer delay and not the categories defined in section 7.1.1.

Surprisingly, there are no significant differences among the three proposed scenarios. This suggests that the impact of the stations that do not belong to the studied trip is limited. A positive note is that the model performance only slightly decreases when used to predict more stations ahead.

Score	Station 0	Station 1	Station 2	Station 3
accuracy	**0.470**	0.433	0.430	0.434
MSE	**21.10**	27.73	29.27	30.67
MAE	**1.721**	2.138	2.236	2.302

Table 7.5: Results with scenario 1

Score	Station 0	Station 1	Station 2	Station 3
accuracy	0.468	0.428	0.427	0.434
MSE (min)	21.21	27.15	28.87	30.86
MAE (min)	1.743	2.121	2.220	2.296

Table 7.6: Results with scenario 2

Score	Station 0	Station 1	Station 2	Station 3
accuracy	0.445	0.392	0.389	0.390
MSE (min)	21.51	26.85	28.89	29.71
MAE (min)	1.774	2.154	2.279	2.350

Table 7.7: Results with scenario 3

When looking in greater detail at the result on the confusion matrices 7.7, it becomes clearer why the first scenario has the upper hand on the others. Especially when looking at the prediction for station 3 (third prediction ahead), it can be observed that the model is biased toward predicting zero delay. This bias comes from the scenario 1 assumption that assigns zero delays to stations not part of the trip under scrutiny. As most trains have zero delay or a delay close to zero, this bias explains the slightly better results of scenario 1.

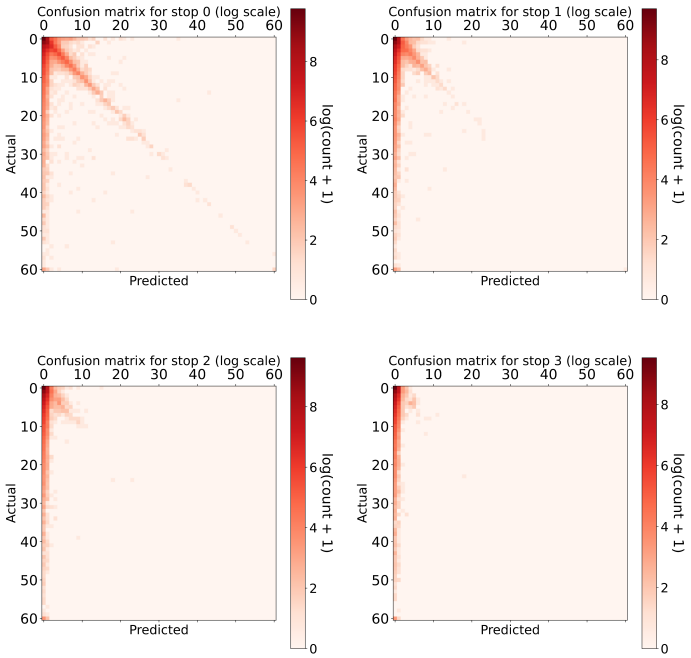


Figure 7.7: Confusion matrix for scenario 1

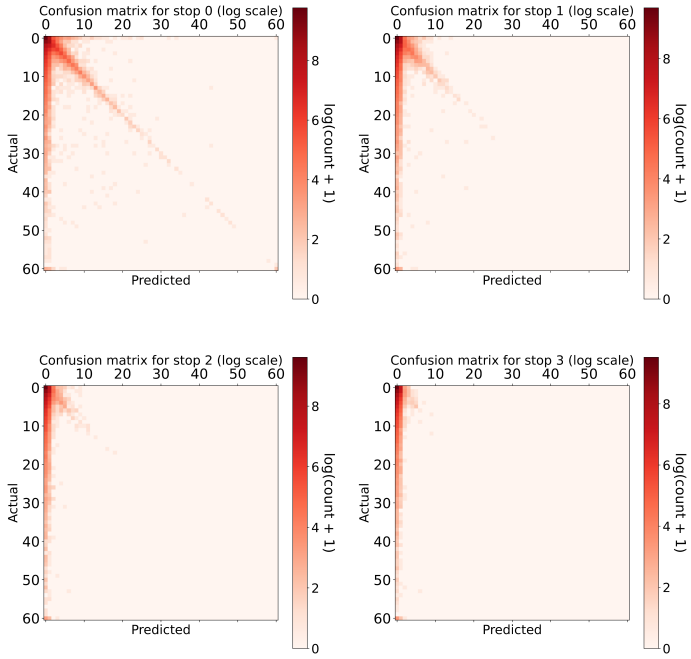


Figure 7.8: Confusion matrix for scenario 2

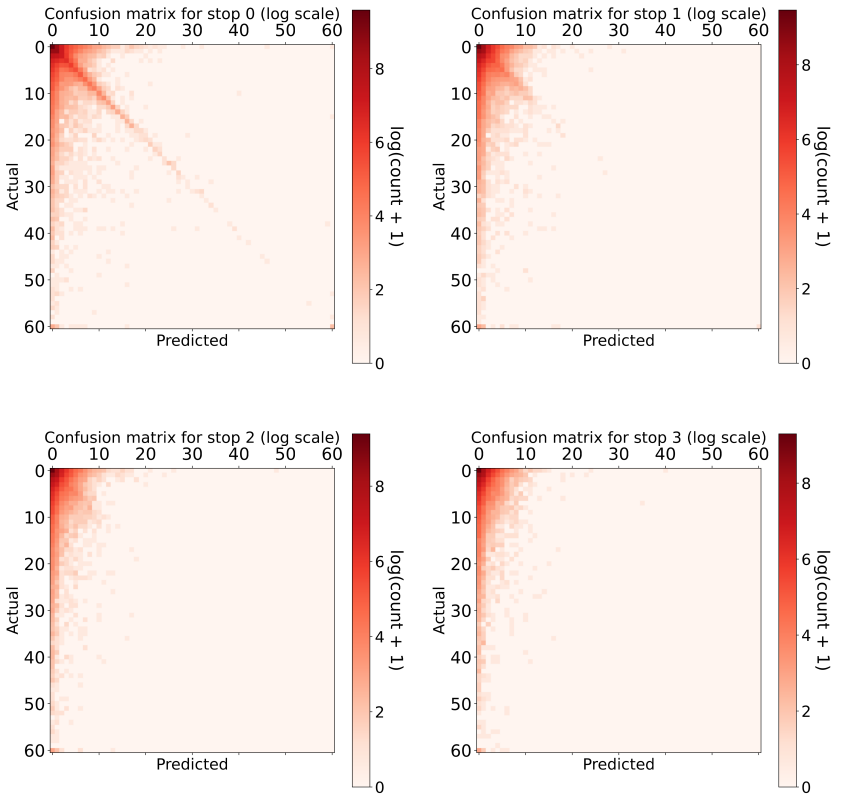


Figure 7.9: Confusion matrix for scenario 3

In the following graphs 7.10, median absolute error (MAE) is plotted against minimum actual delay. So, when x in the x-axis equals 0, y corresponds to the MAE on the entire tested values, i.e. actual delay >= 0. For $x = 10$, the MAE is only computed on the subset of the tested data where actual delay >= 10. Therefore, the plots give an idea of the performance loss when predicting higher delays.

The graphs 7.10 confirm what was initially observed on the confusion matrix. The predictions made under the third scenario are slightly better for higher values. This can be explained by the fact that this model is not biased toward zero delay or median delay, as for scenario 2. The third scenario uses data close to real-time for the station non-part of the studied trip. Those observations counterbalance the first impression that stations non-part of the trip could be discarded, but this should be taken with a grain of salt as the differences are non-significant.

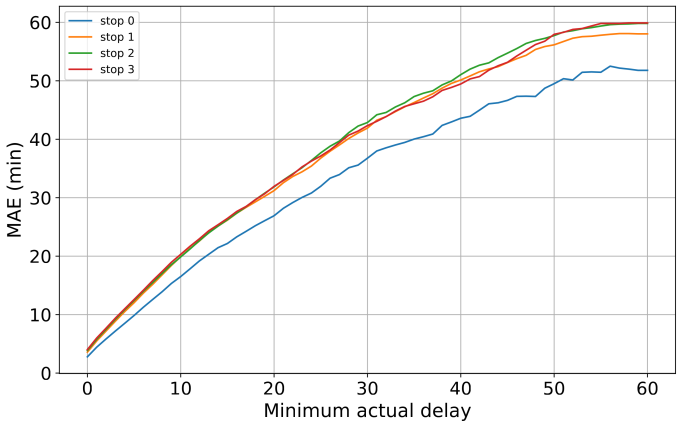


Figure 7.10: MAE for increasing increasing actual delay (Scenario 1)

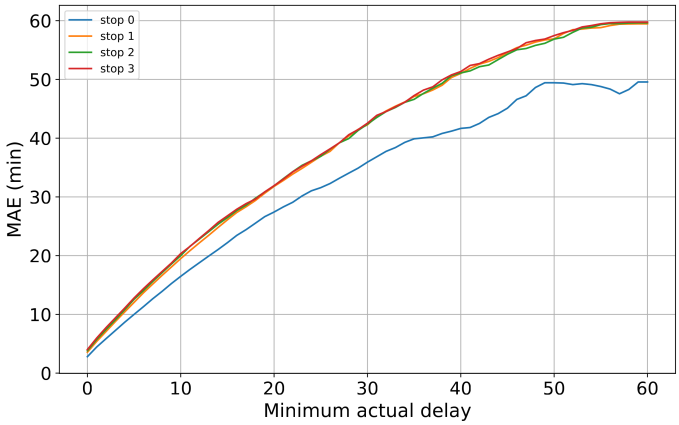


Figure 7.11: MAE for increasing increasing actual delay (Scenario 2)

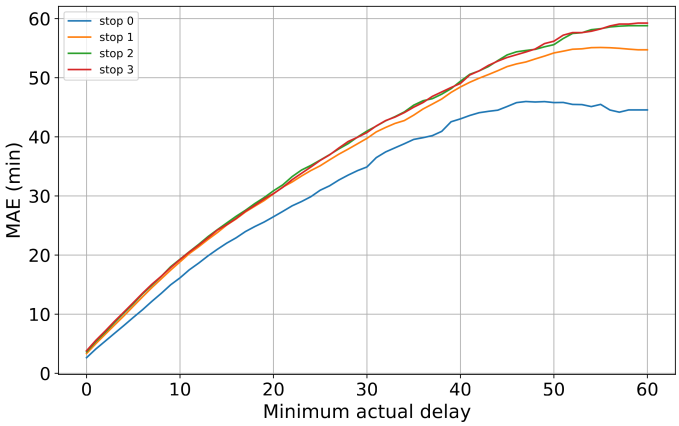


Figure 7.12: MAE for increasing increasing actual delay (Scenario 3)

7.4 Inducer-Susceptible

This section explores the insight behind the *inducer* and *susceptible* metrics. Both metrics have been scaled using the following formula:

$$\text{scaled\_value} = \frac{x - \min}{\max - \min}$$

where x is the input, and min and max correspond respectively to the lowest and highest score.

7.4.1 Markov chain first order

Stop name	Inducer score
Bruxelles-Chappelle	1.0
Bruxelles-Central	0.810
Bruxelles-Congrès	0.592
Bruxelles-Nord	0.412
Bruxelles-Midi	0.317
Lège-Guillemins	0.286
Anvers-Central	0.228
Anvers-Berchem	0.216
Braine-l'Alleud	0.173
Ottignies	0.172

Stop name	Susceptible score
Bruxelles-Central	1.0
Bruxelles-Midi	0.507
Bruxelles-Nord	0.304
Ottignies	0.196
Liège-Guillemins	0.180
Malines	0.180
Anvers-Berchem	0.163
Bruxelles-Luxembourg	0.136
Gand-Dampoort	0.119
Gand-Saint-Pierre	0.117

Table 7.8: Top 10 Inducer and susceptible scores.

Three urban centers stand out with high *inducer* scores: Bruxelles, Anvers, and Liège. The five Bruxelles stations follow each other, but surprisingly they have significant score differences. That being said, the *inducer* score is coherent with the reality: Bruxelles is in the middle of the country, so it "contaminates" the other train stations easily with a large volume of passing through. This same reasoning can be applied to Anvers and Liège. Braine-l'Alleud and Ottignies, on the other hand, are not big cities, but both are major stations of two important tracks connecting Wallonia to Bruxelles 7.13.

The *susceptible* score drop is significant from the first position to the fourth. The second station on the ranking has half the score of the first, and the third has less than a third of the score of the first station. Stations around Gand and Malines only appear on the *susceptible* ranking, but not on the inducer one. Interestingly Bruxelles-Chappelle and Bruxelles-Congrès do not appear on the *susceptible* score. This can be explained because they are minor stations compared to their neighboring ones: Bruxelles-Midi, Bruxelles-Central, and Bruxelles-Nord. They are smaller with fewer passages, and trains running late might just simply skip the station, influencing the data.

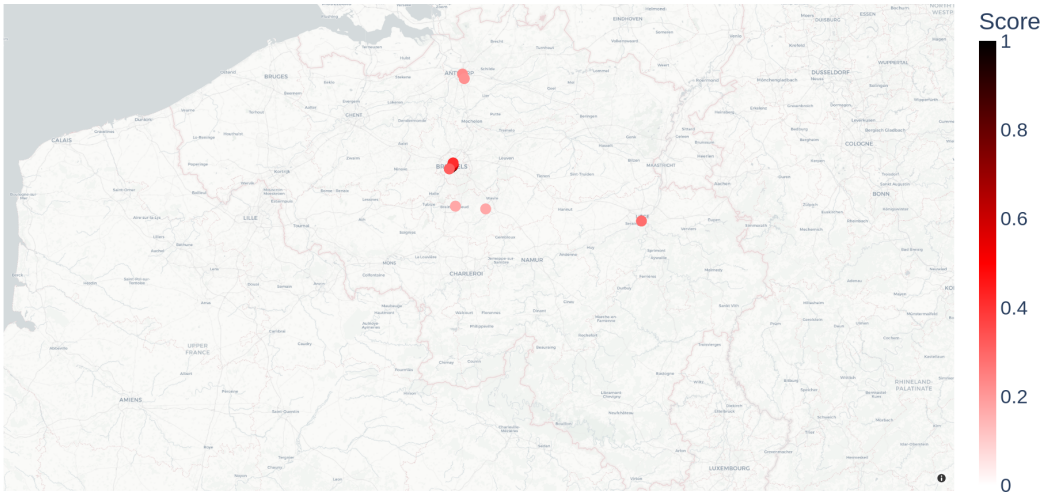


Figure 7.13: Ten highest inducer scores

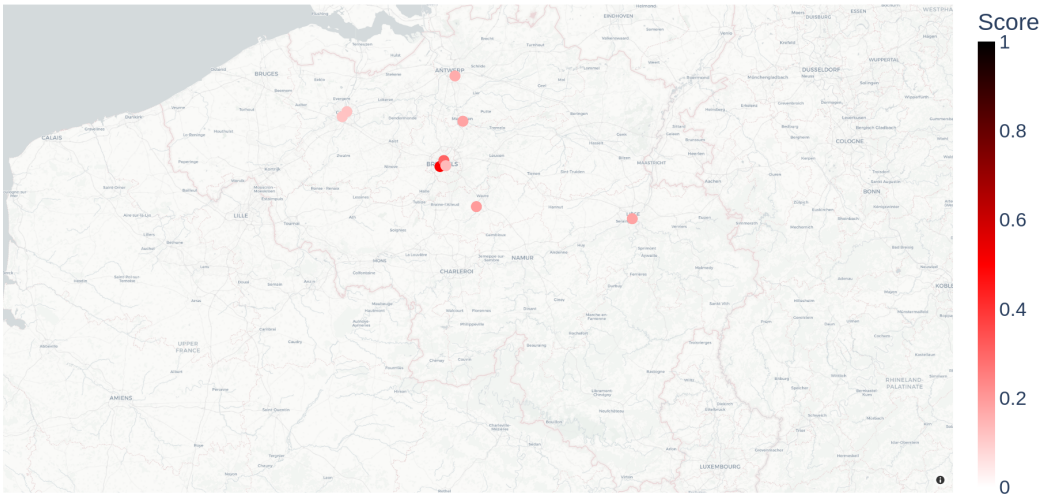


Figure 7.14: Ten highest susceptible scores

7.4.2 Markov random field

Stop name	Inducer score
Wondelgem	1.0
Tour-et-Taxis	0.988
Glons	0.972
Houyet	0.921
Athus-Frontière	0.913
Hamont	0.909
Okegem	0.908
Bertrix	0.907
La Louvière-Sud	0.903
Messancy	0.903

Stop name	Susceptible score
Bruxelles-Midi	1.0
Liège-Guillemins	0.404
Bruxelles-Central	0.213
Ottignies	0.161
Hal	0.128
Bruges	0.113
Anvers-Central	0.110
Diest	0.059
Verviers-Central	0.048
Anvers-Luchtbal	0.04

Table 7.9: Top 10 Inducer and susceptible scores.

In contrast to previous results 7.4.1, the stations having the highest *inducer* scores are minor stations spread evenly on the national railway, see figure 7.15. The different scores are more stacked together. The difference of stations in the ranking can be understood because of how the Markov random field works. For the *inducer* score, the selected station is assigned its median delay, and based on this value, the delays for every station are predicted. The question answered by the model is "Given a median delay x at station s, what are the most likely delays in every other station?". In other words, the model outputs the most likely combination of delays. So, the stations appearing on the list are

correlated with large delays on the network but are not necessarily the source of delay.

Regarding the *susceptible* scores, a steep decrease in the score can be observed. Comparing with results from Markov chain 7.8, Bruxelles-Midi and Bruxelles-Central share both top 3. The newly appearing stations are Hal, Diest, and Verviers-Central. Diest and Vervier-Central have too low scores to be considered. Hal is at the junction of two tracks joining Bruxelles, making the station a sensitive station and, therefore, more likely to be stressed by neighboring stop delays.

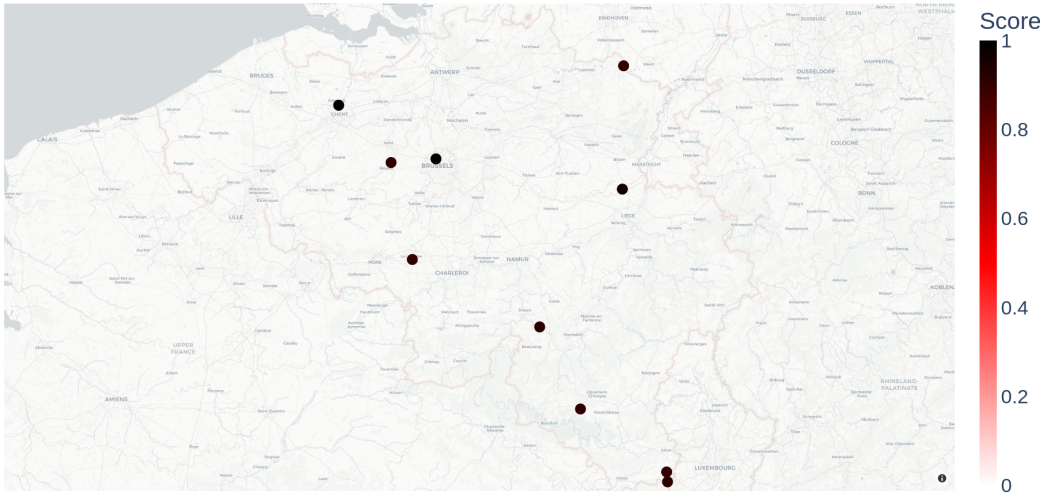


Figure 7.15: Ten highest inducer scores

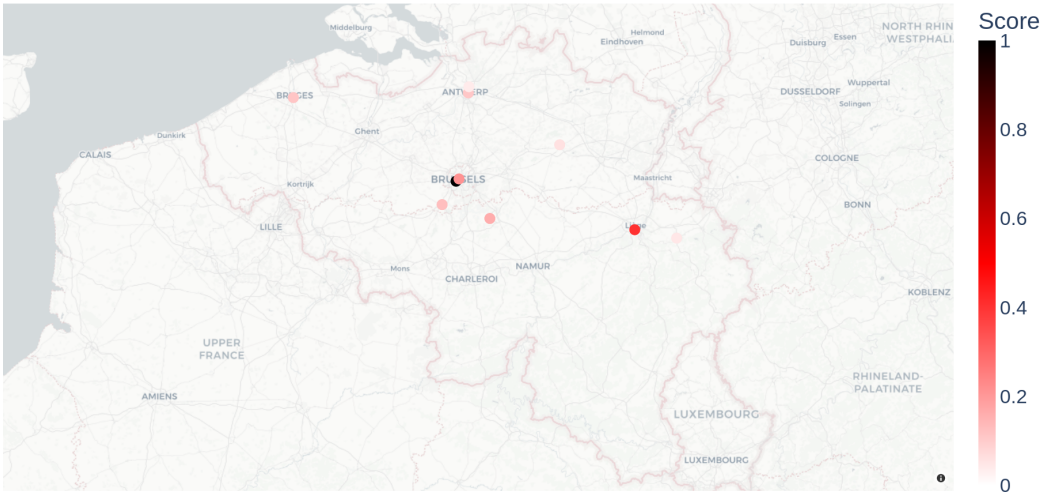


Figure 7.16: Ten highest susceptible scores

In general, *inducer* and *susceptible* score metrics capture sensible information. However, the model used to produce such metrics has to be carefully chosen. As for the example with the inducer scores produced by the Markov random field which outputs control samples correlated to delay levels in the overall network. The delays in the network are produced by the main urban centers. Major stations connecting Bruxelles are sensitive to delays; they do not act as buffers but rather as catalysts, possibly because of the waiting time for connections.

Chapter 8

Conclusion and future work

In this master thesis, we analyzed delays in the Belgian railway network and evaluated multiple solutions to predict them. We also established a new formal definition of a late train in that network. Two base architectures were defined to solve the inference problem.

The first one was a Markov chain. The idea was to use delays and characteristics from previous stations in the trip to predict the delay in the next station. Using this model it was possible to go further into the future by using previously predicted delays as input for the next stations. This architecture necessitated the training of a regression model for every station in the network to act as transition matrices in the chains. Several kinds of models were evaluated on their prediction performance in mean absolute error and accuracy. Using those metrics, the winner, when only real delays were used as input, was the decision tree model. But when the input delays came from prediction, the clear winner was the random forest model with regular decision trees. However, it has been shown that this model was also the slowest at making predictions.

The second architecture was a Markov random field. This method considered the entire railway network as a random field where stations were nodes and rails were edges between nodes. This model was trained and tested under three scenarios. Giving stations outside of the trip on which predictions were made different delay values. The performances were not conclusive compared to the Markov chain model using a random forest, as a strong bias toward predicting zero delays hindered its performances.

Two metrics have been employed to identify the stations with the greatest impact on the network: the inducer metric, which determines the most influential stations overall, and the susceptible metric, which identifies stations that act as delay catalysts. While the results are coherent, it is important to consider the model used when interpreting them. Future work involving simulations could further assess a station's impact on the broader SNCB network.

As this work didn't focus on fine-tuning models used with Markov chain architecture, future works could explore this possibility to improve performance. Another way to improve on the Markov chain architecture would be to use self-attention networks able to establish the relevance of previous results for the current station. A paper published in 2017 presented graph attention networks [26]. This model combines the previous idea of the attention mechanism with the network architecture of the railway. It looks very promising and might be very good at solving delay prediction problems.

Bibliography

[1] Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. “Learning optimal decision trees using caching branch-and-bound search”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 3146–3153.

[2] Gely P. Basharin, Amy N. Langville, and Valeriy A. Naumov. “The life and work of A.A. Markov”. In: *Linear Algebra and its Applications* 386 (2004). Special Issue on the Conference on the Numerical Solution of Markov Chains 2003, pp. 3–26. ISSN: 0024-3795.

[3] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[4] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32.

[5] Peter Clifford. “Markov random fields in statistics”. In: *Disorder in physical systems: A volume in honour of John M. Hammersley* (1990), pp. 19–32.

[6] Francesco Corman and Pavle Kecman. “Stochastic prediction of train delays in real-time using Bayesian networks”. In: *Transportation Research Part C: Emerging Technologies* 95 (2018), pp. 599–615.

[7] Mark Dingler et al. “Determining the causes of train delay”. In: *AREMA Annual Conference Proceedings*. The American Railway Engineering and Maintenance-of-Way Association Lanham, MD. 2010.

[8] Mathieu Fourment and Michael R Gillings. “A comparison of common programming languages used in bioinformatics”. In: *BMC bioinformatics* 9 (2008), pp. 1–9.

[9] Ramashish Gaurav and Biplav Srivastava. “Estimating train delays in a large rail network using a zero shot markov model”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 1221–1226.

[10] Jürgen Groß. *Linear regression*. Vol. 175. Springer Science & Business Media, 2003.

[11] Tin Kam Ho. “Random decision forests”. In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282.

[12] David C Hoaglin, Boris Iglewicz, and John W Tukey. “Performance of some resistant rules for outlier labeling”. In: *Journal of the American Statistical Association* 81.396 (1986), pp. 991–999.

[13] Arthur E Hoerl and Robert W Kennard. “Ridge regression: Biased estimation for nonorthogonal problems”. In: *Technometrics* 12.1 (1970), pp. 55–67.

[14] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[15] Ping Huang et al. “A Bayesian network model to predict the effects of interruptions on train operations”. In: *Transportation Research Part C: Emerging Technologies* 114 (2020), pp. 338–358.

[16] Liujiang Kang and Xiaoning Zhu. “A simulated annealing algorithm for first train transfer problem in urban railway networks”. In: *Applied Mathematical Modelling* 40.1 (2016), pp. 419–435.

[17] Andrei Andreevich Markov. “Rasprostranenie zakona bol’shih chisel na velichiny, zavisyaschie drug ot druga”. In: *Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete* 15.135-156 (1906), p. 18.

[18] Nils OE Olsson and Hans Haugland. “Influencing factors on train punctuality—results from some Norwegian studies”. In: *Transport policy* 11.4 (2004), pp. 387–397.

[19] Judea Pearl. “Fusion, propagation, and structuring in belief networks”. In: *Probabilistic and Causal Inference: The Works of Judea Pearl*. 2022, pp. 139–188.

[20] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[21] İsmail Şahin. “Markov chain model for delay distribution in train schedules: Assessing the effectiveness of time allowances”. In: *Journal of rail transport planning & management* 7.3 (2017), pp. 101–113.

[22] Ahmad Shamshad et al. “First and second order Markov chain models for synthetic generation of wind speed time series”. In: *Energy* 30.5 (2005), pp. 693–708.

[23] Alice Sternberg et al. “A review on flight delay prediction”. In: *arXiv preprint arXiv:1703.06118* (2017).

[24] Kah Yong Tiong, Zhenliang Ma, and Carl-William Palmqvist. “A review of data-driven approaches to predict train delays”. In: *Transportation Research Part C: Emerging Technologies* 148 (2023), p. 104027.

[25] Mehmet Baran Ulak, Anil Yazici, and Yun Zhang. “Analyzing network-wide patterns of rail transit delays using Bayesian network learning”. In: *Transportation Research Part C: Emerging Technologies* 119 (2020), p. 102749.

[26] Petar Veličković et al. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).

[27] Wessel N van Wieringen. “Lecture notes on ridge regression”. In: *arXiv preprint arXiv:1509.09169* (2015).

[28] Jianjun Wu et al. “Equity-based timetable synchronization optimization in urban subway network”. In: *Transportation Research Part C: Emerging Technologies* 51 (2015), pp. 1–18.

[29] MENG WUWei-wei and ZHANG Hao-yu. “Flight plan optimization based on airport delay prediction”. In: *Journal of Transportation Systems Engineering and Information Technology* 16.6 (2016), p. 189.

[30] Ning Xu et al. “Estimation of delay propagation in the national aviation system using Bayesian networks”. In: *6th USA/Europe Air Traffic Management Research and Development Seminar*. FAA and Eurocontrol Baltimore. 2005.

[31] Jianxin Yuan and Ingo A Hansen. “Optimizing capacity utilization of stations by estimating knock-on train delays”. In: *Transportation Research Part B: Methodological* 41.2 (2007), pp. 202–217.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**