# Detecting Patterns in Train Position Data of Trains in Shunting Yards

## Analysis of Arrival Time Distributions and Delays

**Amanda Krudde**[1]

**Supervisors: Prof. dr. Mathijs de Weerdt**[1]**, ir. Issa Hanou**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Amanda Krudde
Final project course: CSE3000 Research Project
Thesis committee: Mathijs de Weerdt, Issa Hanou, Jing Sun

## Abstract

Shunting yards are locations next to train stations that serve as parking places for trains when they are not in operation and often contain facilities for maintenance and cleaning for passenger trains. Planning of the tasks regarding shunting trains involves routing, assignment of tracks, and scheduling tasks. This is done manually and requires a lot of effort, making it inefficient. Identifying patterns specific in the arrival times of trains at shunting yards can help to predict future train arrivals and potential delays throughout the year more accurately. This enables the alignment of staff and equipment with train arrivals, minimizing idle time and optimizing cost efficiency.

This research focuses on extracting and analyzing the arrival times of trains at shunting yards using a dataset consisting of GPS data. It conducts two algorithms to cluster the given data for each train unit within and across days to identify the same train across different days. Distributions and heatmaps of the arrival times and delays are created based on the identified train series. They are analyzed to identify patterns in train arrival times and delays across different months.


## 1 Introduction

An important part of daily travel in the Netherlands is the train system. Each day, around 4.800 passenger trains operate in the country for more than one million travelers [9]. Most of these trains run during peak hours, which are Monday to Friday from 06:30 to 09:00 and 16:00 to 18:30. When trains are not in use, they are placed in parking areas called shunting yards. These locations contain many tracks, are placed close to train stations and often also contain facilities for maintenance and cleaning. Figure 1 shows all the tracks in the area of Amersfoort in gray, highlighting the shunting yard with blue tracks.

Figure 1: Area Amersfoort with a shunting yard (blue tracks).

The planning of the tasks regarding shunting yards is also known as the Train Unit Shunting Problem [3]. It contains among other things; routing trains between stations and shunting yards, assigning tracks to trains, and scheduling servicing tasks. Planning and coordinating these tasks are still done manually, making it a laborious and time-consuming process.

To help the people creating the planning of tasks, it is useful to detect patterns in the operations regarding shunting yards that happen frequently. Patterns can help identify recurring issues, which will optimize scheduling. Humans can recognize certain features and patterns which can simplify the Train Unit Shunting Problem while current models cannot do this [15].

ProRail, the owner and maintainer of the Dutch rail network, is currently looking into different patterns within the shunting yards to reduce the Train Unit Shunting Problem. Next to ProRail, other researchers have proposed optimization methods for sub-components of the problem. For example, the feasibility of shunting movements with respect to the layout of the shunting yard [5] and the assignment of tracks to trains in shunting yards [4]. There remains a big gap in the scientific literature regarding the analysis of arrival times of trains, resulting in the motivation of this research.

This research analyzes the arrival times and delays of trains in shunting yards to uncover patterns for predicting future train arrivals and delays more accurately. Identifying peak moments in train arrivals allows for early scheduling of additional staff and equipment when needed. Next to this, knowing potential delays in arrival times can help allocate more resources when needed to reduce delays in the future. This allows for a more reliable and consistent train schedule.

Data is needed before it is possible to find patterns in the arrival times of trains in the shunting yards. Currently, realization data exist that reflect the actual occurrences rather than the planned movements of trains. It contains GPS coordinates of each train unit around seven shunting yards in the Netherlands over 10 months. It could be that a train consists of more than one train unit, which means it could contain multiple GPS coordinates for the same train, see Figure 2. Before this data can be used to find patterns, the data is processed to combine train units as trains.

Figure 2: One train consisting of two train units with a GPS coordinate.

This paper aims to answer the following research question:

**What patterns can be identified in arrival time distributions of trains in shunting yards using train position data?**

This question is further divided into sub-questions, providing a structured approach to answer the main research question:

**RQ 1**: What features from the given dataset are relevant for extracting the arrival times of the trains in shunting yards?
**RQ 2**: Which clustering method can be used to group train units of the same train?
**RQ 3**: What methods can be used to group trains that represent the same train across different days?

**RQ 4**: How can the grouped data be used to derive a distribution of arrival times for each train series?
**RQ 5**: What conclusions can be drawn from the derived distributions of the arrival times?

Throughout this research, it is hypothesized that patterns will be found. It is expected that almost all trains arrive in the shunting yards directly after peak hours and not within since shunting yards mainly exist as parking places. Train series are expected to be the same throughout the year, assuming the train schedule stays the same. Next to this, weather conditions could influence the tracks and therefore possible delays. Therefore, it is expected that there will be more delays during the winter months in comparison to the summer months.

In addition to being useful for ProRail and trains in the Netherlands, this research can also be placed in a broader context. It can be applied to train systems in other countries and to other transportation systems involving grouping instances. For instance, it can be valuable for analyzing the arrival times of ships in ports or airplanes at airports, as both ships and airplanes are equipped with GPS coordinates for safety and navigation.

## 2 Background

This section contains background information about the Train Unit Shunting Problem and existing clustering algorithms that are used in this research. It also contains related scientific work and a clear description of the received dataset.

### 2.1 Background

**Train Unit Shunting Problem**
Passenger trains are parked in parking areas called shunting yards which often contain facilities for maintenance and cleaning. The related shunting processes include the matching of arriving and departing shunt units, the routing of shunt units over the station infrastructure, cleaning and short-term maintenance of shunt units, and crew planning for the crew that carries out the shunting processes [3]. Planning and coordinating these tasks are still done manually, making it laborious and time-consuming. Planning these tasks is also known as the Train Unit Shunting Problem. Automated solutions are currently being researched to improve the efficiency of the process, aiming to reduce manual workload.

**Existing clustering algorithms**
There exist several ways to group similar train instances as one. Below are a few possible clustering algorithms considered during this research: K-Means clustering, DBSCAN, and Hierarchical clustering.

K-means clustering is an unsupervised machine learning algorithm that clusters the dataset into a pre-defined number of clusters [14]. With this algorithm, you need to know how many clusters you want to use beforehand. However, when grouping train units as one train, it is not known in advance how many units the train exists. Next to this, it is also not known beforehand how many train series can be identified. The K-means clustering algorithm is therefore not applicable in this research.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that groups points that are closely packed while identifying outliers that lie in low-density regions [13]. It starts by finding core and border points; core points are points that contain at least X amount of points (MinPts) in its neighborhood ($\epsilon$) and border points are the points within that neighborhood that are not a core point itself. All other points are specified as outliers [16]. DBSCAN clusters the points based on their density, starting from core points and expanding to include reachable points within their neighborhood. As a result, DBSCAN does not depend on a pre-defined number for the number of clusters, but determines them based on their density and distribution of the points, making it a useful technique for handling noise and creating clusters with different densities or irregular shapes. DBSCAN has been proved to be significantly more effective in discovering clusters of arbitrary shape than the well-known algorithm CLARANS[1], an adaptation of the k-means algorithm, and DBSCAN outperforms CLARANS by a factor of more than 100 in terms of efficiency [2].

Hierarchical clustering creates a hierarchy of groups, such that objects within a group are similar to each other and different from objects in other groups [7]. It is not needed to pre-define the number of clusters. However, hierarchical clustering algorithms do not actually create clusters but compute only a hierarchical representation of the data set [12]. Hierarchical clustering can be visualized using a dendrogram. A dendrogram shows the distance between all the data points in a tree-like structure [11]. The tree structure can be cut off at any distance and the amount of groups will follow. This clustering algorithm is useful for this research because a train can consist of more than two units and the groups are not set to be a certain size.

### 2.2 Related work

An example where train data was used to improve service management is in the subway system of New York City. In 2020, trains were automatically identified in real-time [8] and matched with the train schedules. They created applications for internal use, e.g. a real-time visualization of slow-speed segments, but did not look into patterns over time.

Another study focusing on New York City Transit's subway system highlights the importance of identifying the root causes of delays. Using historical data to pinpoint specific incidents responsible for delays, resulted in better operational decisions and efficiency strategies [6]. Unlike their approach, this research identifies patterns after delays occur.

A study already applied DBSCAN to identify patterns in temperature data, focusing on anomaly detection. Anomalies are unexpected patterns in a dataset, those that do not conform to the general behavior [17]. They pre-processed time series data before applying DBSCAN and found that DBSCAN can discover subtle anomalies. This finding is useful for this research because it suggests that DBSCAN can also effectively identify patterns in train arrival times, even if they are not immediately obvious.

[1] https://javadoc.io/doc/com.github.haifengl/smile-core/1.0.4/smile/clustering/CLARANS.html

## 2.3 Dataset

**GPS points**

The received dataset is from ProRail and contains realization data from NS trains in the Netherlands. This data reflects the actual occurrences of trains rather than the planned movements of trains. The dataset contains those GPS points every 10 seconds of each train unit whenever it is present in the area. Each train unit has a GPS tracker, so whenever a train consists of more than one train unit, it contains multiple GPS coordinates for the same train, see Figure 2. The data is from seven areas[2] in the Netherlands and each area contains a shunting yard. The data is from a period of 10 months, namely May 2023 through February 2024.

Besides containing the GPS locations of the units, the dataset also contains information about the tracks in each area.

The area of Amersfoort contains the largest shunting yard amongst the shunting yards in the given dataset. Next to this, it also contains most trains moving into the shunting yard, resulting in more relevant data points. Therefore, this research only focuses on the Amersfoort area.

**Values of data**

There does not exist any documentation of the dataset. All statements and interpretations below are logically derived from the data itself. Because of this, there could exist misinterpretations.

The dataset contains a lot of fields, of which most are not relevant to this research. Some features are manually annotated, examples of these are *Track* and *ActivityType*. It could be possible that there are human errors in the labeling of these features. For example, the track can suddenly be a track next to it, or the *ActivityType* is called "Entering" while the train is actually leaving the area. This has been taken into account when extracting the relevant features of trains (section 3.1). Other features, such as *Matnr* (id of the unit) and *TimeStart* (start time of the GPS coordinate) are the true values and are mostly used in this research.

**Storage**

The data is stored in an Azure environment. This means that the data is hosted and managed using Microsoft's cloud computing platform. The data is protected and only accessible through a ProRail account.

# 3 Methodology

In this section, the chosen method and algorithms used in this research are discussed. Each step of data processing and analysis is implemented in Python, using Polars and Pandas libraries.

## 3.1 Data Processing

The data contains a lot of data points that are not relevant to the current research and need to be filtered. This section explains both what the relevant features are and what data points are extracted for the next step in this research.

[2] Amersfoort, Arnhem Goederen, Arnhem West, Carthesiusweg, Dordrecht, Hoofddorp, Watergraafsmeer

**Relevant features of trains**

The raw train data contains a data point for every GPS coordinate of a unit present in the area. The amount of points for a unit depends on the movements and time spent within the area. First, the data is filtered based on the day; only the data points that are on weekdays are kept because this research only focuses on patterns within weekdays.

The units are grouped based on their *GroupIdHash*, where each grouped unit contains a *Matnr*, a *UnitType* and a list of movements, which contains the *TimeStart*, *TimeEnd*, *Track* and *ActivityType* of each data point of the unit.

Next, the grouped units are filtered, based on if the unit is present in a shunting yard in at least three data points. The filtering is done based on the tracks in its movements. Three data points are chosen because there can be a human error in the allocation of the tracks.

The first timestamps as soon as the unit enters the shunting yard are retrieved from the units that are present in the shunting yards for at least three data points. The resulting dataset contains a *Matnr*, a *UnitType*, and three *StartTimes* for each unit.

**Features that are used:**

| Feature | Description |
|---|---|
| GroupIdHash | Hash of the unit containing the Matnr and the date/time the unit enters the area |
| Matnr | ID of the unit ("Materiaalnummer") |
| UnitType: | Type [10] and length of the unit with sub-types: Intercity (ICMm, DDZ or VIRM) & Sprinter (SLT or SNG) |
| TimeStart | Start time of the GPS coordinate |
| TimeEnd | End time of the GPS coordinate |
| Track | Name of the track |
| ActivityType | Activity type of the train with options: Short stop, Opgesteld, Entering/Exiting, Exit, Rangeren (Shunting) |

**Relevant features of tracks**

The dataset contains information about the tracks, e.g. track name, electric traction, voltage class, GPS location, and area. Most areas can be divided into smaller sub-areas such as shunting yards, these are called "Geocode" locations. An example can be found in Figure 1 where the shunting yard has Geocode "Amersfoort Bokkenduinen". The shunting yards were found by manually visualizing the different sub-areas of areas and the tracks of a sub-area were retrieved.

**Features that are used:**

| Feature | Description |
|---|---|
| GEOCODE_NAAM_BEGIN and GEOCODE_NAAM_EIND | "Begin" and "Eind" indicate the geocode location at beginning/end of an track (they can differ) |
| Naam | Name of the track (corresponding to "Track" from features of trains) |

The resulting dataset contains all relevant information for this research. The amount of features is reduced from 28 to 3 during processing. The amount of data points after processing is reduced by 99%. This allows for faster processing and interpretation of the next steps.

## 3.2 Grouping similar train instances

With the clean dataset, similar train instances have to be matched. Train units are clustered as one train using hierarchical clustering. In the next step, these trains are matched across days to identify train series that represent the same train. Two algorithms are introduced and explained.

### Cluster units as one train

Each day in the dataset is clustered separately. The clustering process for each day is as follows:

1. Create a distance matrix for the lists of timestamps when entering the shunting yard using fastdtw[3] which is an approximate Dynamic Time Warping (DTW) algorithm able to calculate distances between time instances.
2. Normalize the distance matrix.
3. Perform hierarchical clustering using scipy[4] module.
4. Create clusters with a very low threshold (distance of max. 0.1%). Figure 3 shows the hierarchical clustering as a dendrogram. Each cluster that has a distance below the threshold (horizontal line) is accepted as a cluster. It is visible in the figure that clusters of three units also can be created. This dendrogram is zoomed on the y-axis, and the full dendrogram is visible in Figure 9.
5. It could be the case that two units are driving very close to each other as they go into the shunting yard and are being grouped by hierarchical clustering. However, this should not be counted as one train. Therefore, each cluster will be filtered based on the sub-type. The most common sub-type within the cluster is found first. Then, all units that do not match the most common sub-type become separate clusters.

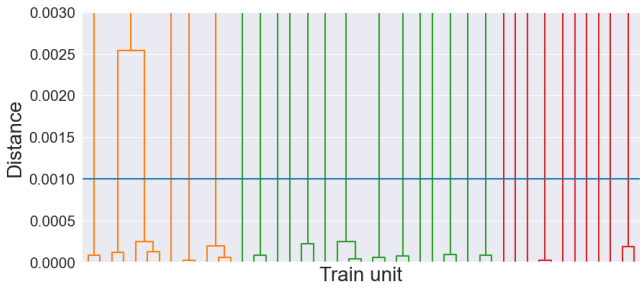



Figure 3: Hierarchical clustering shown in a dendrogram plot of a random day (20-06-2023), including the horizontal threshold.

The final clusters contain for each arriving train: a list of all *Matnr* of the units, the *UnitType*, and the *ArrivalTime* (the time the first unit arrives in the shunting yard). Figure 4 and Figure 8a show the timestamps of arriving trains across a random week and ten months, respectively. It can be seen in Figure 8a that there are several days without any trains arriving at the shunting yard in August, this is due to missing data. The figures show that certain trains can be linked vertically, meaning their daily arrival times are similar. The next step is to cluster those trains accordingly.

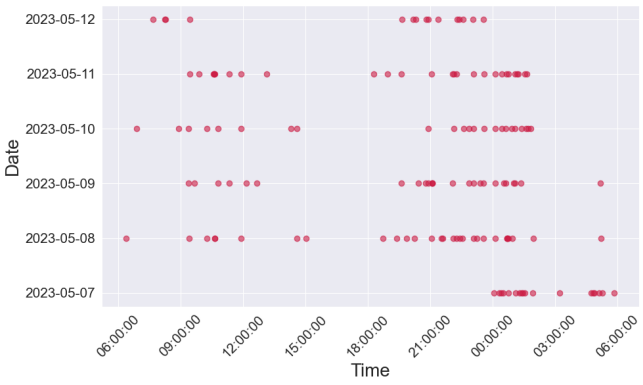



Figure 4: Scatterplot of the arrival times in a random week. Each dot represents a train arriving in shunting yard Amersfoort on the given date and time.

### Match train instances across days

There are several constraints when creating a train series of trains across days that represent the same train:

- Each day can only be present once.
- The type of the train (sprinter/intercity) is the same.
- There are at least twice the amount of trains as the number of weeks in the time frame. This avoids trains being added if it has a repeating pattern on one day of the week.

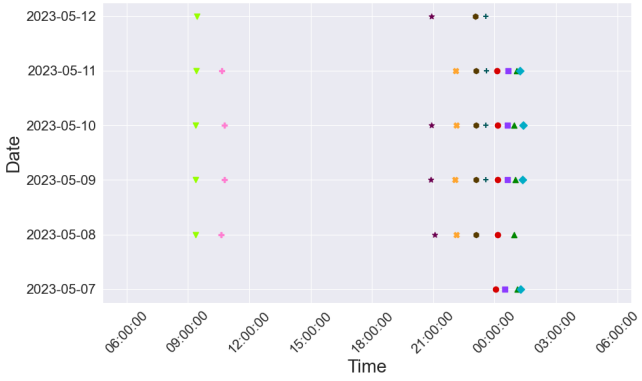
Adding constraints to clustering significantly increases its complexity. Due to time constraints, this research evaluates two clustering algorithms where constraints are filtered afterward, as explained below. Visualizations of the results of the two algorithms across a random week (8th-12th of May) are visible in Figure 5. Each created time series by the corresponding algorithm is represented as a combination of symbol and color. The symbols/colors are picked randomly so they do not match across different figures.

- **Created algorithm**
  This algorithm loops over the arriving trains and adds the train to an existing cluster if the time within the cluster is less than 10 minutes apart, the trains have the same type and the days are different. Otherwise, the arriving train will be added to a new cluster. The pseudocode of the algorithm is shown in Algorithm 1. Figure 5a shows a small example of the clustering algorithm.

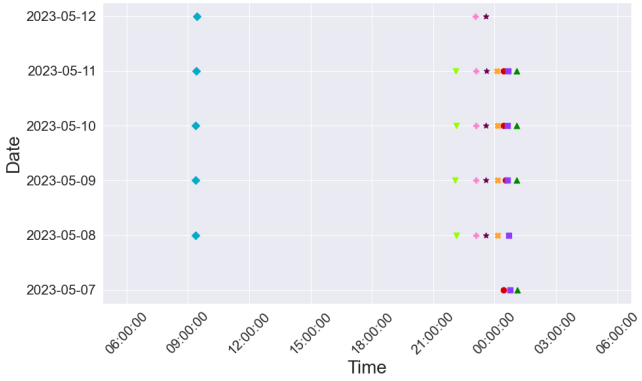
[3] https://github.com/slaypni/fastdtw

[4] https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html

- **DBSCAN**
  DBSCAN from sklearn[5] is applied to the dataset of the arriving trains. The input contains the arriving time in seconds (not taking the date into account) and an encoded list of the types of trains (to ensure clusters have the same train type). After applying DBSCAN on the arriving trains, the created clusters are filtered, such that whenever a date is present multiple times in the cluster, only the first timestamp of that date is saved in the series. In Figure 5b a small example of DBSCAN is shown.




(a) Identified train series using the created clustering algorithm.




(b) Identified train series using DBSCAN.

Figure 5: Scatterplot of the arrival times of the identified train series in a random week (8th-12th of May).

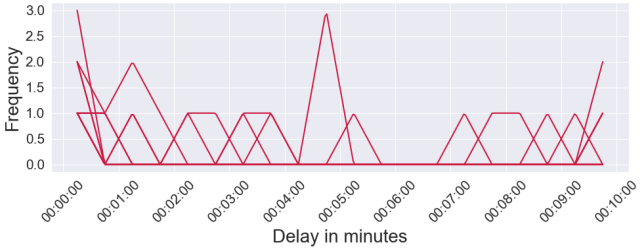
As can be seen in Figure 5, there is a very subtle difference in using the created algorithm and DBSCAN. There are a few more train series identified in the created algorithm, but the train series derived from DBSCAN are denser. Figure 10 shows another example of the two algorithms, across one month of data, where the same observations can be made.

## 3.3 Deriving distributions

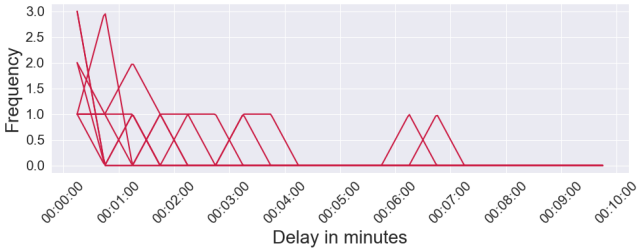
Using the grouped data from both the created algorithm and DBSCAN, clear visualizations of the distributions are created. For smaller ranges of data, a line graph is used to provide detailed insights into the delays over time. To compare data across months, a heatmap is used, which offers an accessible way to detect temporal patterns.

### Line graph visualization

The trains within each train series are counted by occurrences in a time interval of 30 seconds and mapped to start at 00:00:00. It is expected that the line graph of train arrivals will decrease over time, where most trains within the series arrive at 00:00:00 and fewer trains arrive at a later time. Each line in Figure 6 represents a train series corresponding to a combination of symbol and color from Figure 5.




(a) Line graph of identified train series using the created clustering algorithm.




(b) Line graph of identified train series using DBSCAN.

Figure 6: Line graph visualization of arrival times in a random week (8th-12th of May).

Figure 11 and Figure 12 show two more examples of two random months: May and October. The difference between the two algorithms becomes clear when looking at multiple line graphs across weeks and months. The line graphs after using DBSCAN clustering are as expected; the number of train arrivals decreases over time, where most trains arrive at 00:00:00 and fewer trains arrive at a later time.

The created algorithm fails when looking at larger ranges of data. This happens because it adds trains to a cluster if the arrival time is less than a 10-minute difference from the other trains in the cluster. The algorithm does not give priority to trains that have a similar arrival time; it therefore does not take density into account. This results in more spread distributions and is less useful for this research.

Since the created algorithm fails here, only the results from DBSCAN are further analyzed. Choosing parameters for DBSCAN in larger ranges of data (e.g. multiple months) resulted in more cluttered train series and choosing parameters for smaller ranges of data (e.g. days) resulted in train series too small to detect delays. Therefore, it is decided to cluster the data with monthly intervals which also makes it easier to compare delays across months.

[5]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

**Heatmap visualization**

When comparing the data across several months, line graphs tend to become more crowded and less readable. A heatmap is a better way to easily visualize the differences between trains across months because it reveals temporal patterns using color intensity.

The data of one month can be visualized using a heatmap. Figure 7 shows a heatmap of the line graph in Figure 11b. It is visible that each dark spot in the heatmap represents a peak in the line graph. The dark spot on the first row around 2.5-3 minutes means that from that train series (where the earliest train arrives around midnight at 00:05:10), most trains arrive 3 minutes late. This means that most trains that should arrive at 00:05:10, are arriving at around 00:08:10.

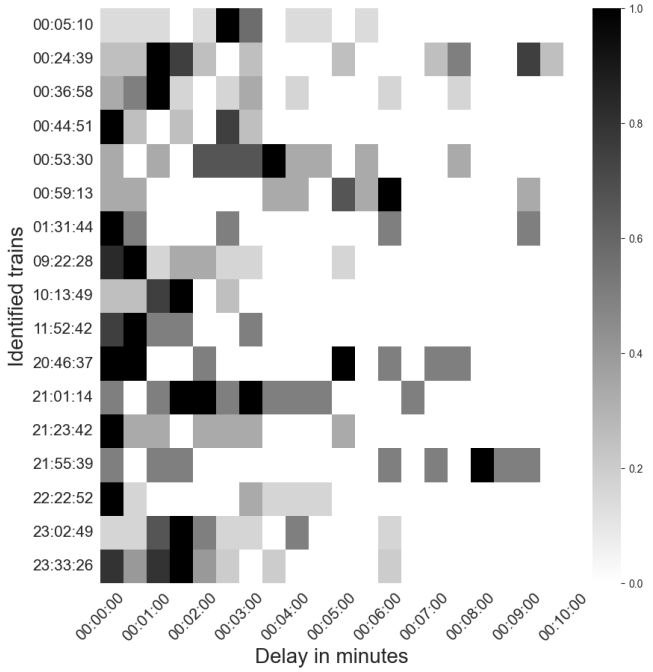
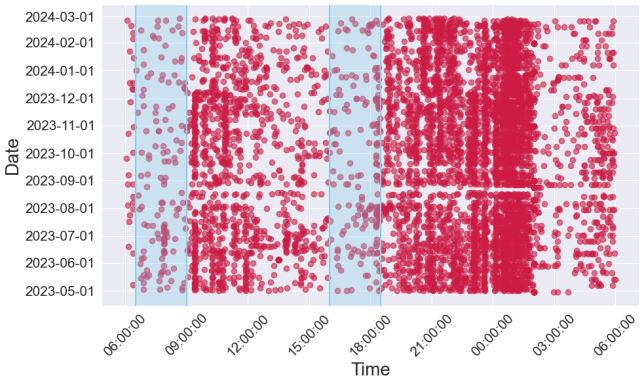


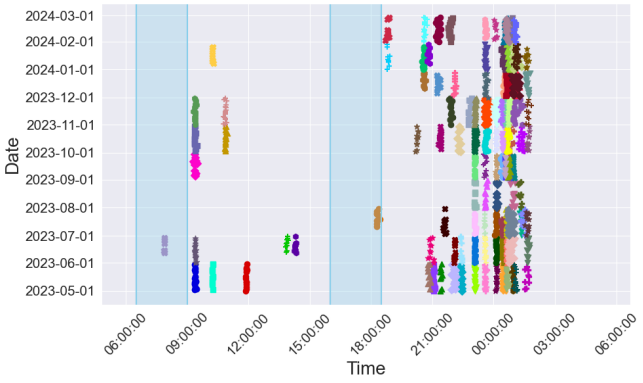

Figure 7: Heatmap visualization of the identified train series in May. Each row represents a train series where the y-axis shows the arrival time of the earliest train and the x-axis shows the delays of all the trains within the train series.

Comparison between months can be done if the delays within each month are summed and then normalized. Merging those heatmaps of each month in the data (10 months) results in Figure 8c. Here, the dark spot on the first row at 0 minutes means that in the 5th month (May) most trains arrive within 30 seconds. This can be confirmed by the heatmap in Figure 7 and the line graph in Figure 11b. It does not mean that each dark spot in the heatmap represents the same amount of trains, since each month has been normalized.

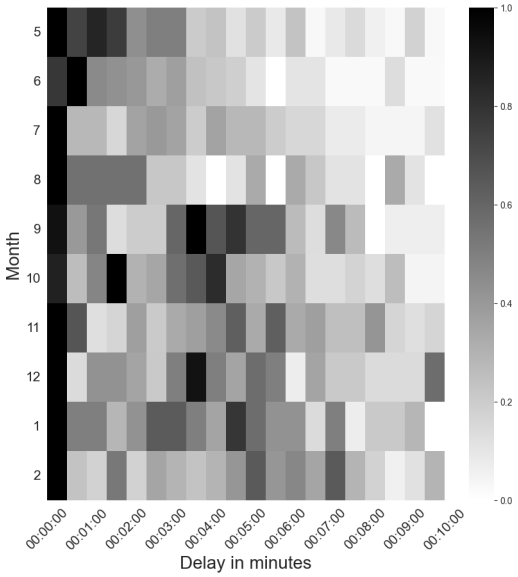
For more clarification, Figure 8 shows an overview of the scatterplot (Figure 8a), found train series (Figure 8b) and the resulting heatmap (Figure 8c) across all data.




(a) Scatterplot of the arrival times.




(b) Identified train series created by DBSCAN.




(c) Each row represents the normalized distribution of delays within each month where the x-axis shows the delay in minutes.

Figure 8: Scatterplot, identified train series and heatmap in ten months (May 2023 - February 2024). Peak hours are highlighted in blue areas.

# 4 Results

The arriving train series and heatmap from Figure 8 are analyzed to see if patterns can be identified. Both patterns are detected in the arrival times of the trains in shunting yards and in the delays of trains.

## 4.1 Patterns in arrival times

Figure 8a shows two visible busy moments in trains arriving at the shunting yard, right after the morning and afternoon peak hours (after 09:00:00 and 18:00:00). Even during peak hours, trains arrive in the shunting yard.

Recurring trains become visible and can be linked vertically, meaning they have similar arrival times. These train series are identified by DBSCAN in Figure 8b. It also becomes visible that almost all train series are after peak hours.

Fewer train series are identified in August, likely due to the missing data, but there are also fewer train series identified after the morning peak in December and February.

Next to this, there are no train series identified during midnight at 02:00:00 and 06:00:00.

## 4.2 Patterns in delays

In Figure 8c the darker areas correspond to more trains arriving and white areas mean that fewer trains arrive. This corresponds to the amount of delay trains in each month have.

It can be seen that in May, June, July, and August, most trains arrive within 1 minute, having almost no delay. Whereas in the months following, most trains arrive 3+ minutes later than expected. Mostly in September, a lot of trains arrive around 4 minutes late.

# 5 Responsible Research

## 5.1 Data security

The dataset received for this research is owned by ProRail and contains GPS points from actual trains driving in the Netherlands. These data points could potentially reveal high-traffic areas. Given the critical role of train transport in the Netherlands, the data is highly sensitive. Therefore, the data is securely stored in an Azure environment, accessible only by ProRail employees.

## 5.2 Data validity

Some features in the data are manually annotated, as mentioned in subsection 2.3. While this research has this taken into account, there could still be errors in the annotation of tracks of shunting yards or in what was assumed to be "true" values. This research assumes that the given data is correctly labeled for the used features. It could therefore be the case that misinterpretations exist, but this is unavoidable unless the data has been self-generated.

## 5.3 Reproducibility

This research is fully reproducible if the same data as described in subsection 2.3 is provided by ProRail. Since the algorithm does not involve deep learning, it will produce the same result each time.

Additionally, the algorithms can be used to analyze different months or areas outside the current dataset, as long as the data is structured the same way as described in subsection 3.1.

## 5.4 Reliability

The used dataset only contains data from NS passenger trains while in reality, the tracks in shunting yards are also used by trains from other companies (e.g. freight trains). The derived distributions are therefore not an accurate representation of the train arrival times, but only an indication. It is not possible to make a definite conclusion based on the given dataset.

# 6 Discussion

The busiest moment in the shunting yard is after the afternoon peak and the second busiest moment is after the morning peak. This is expected since one of the uses of a shunting yard is to facilitate parking places for trains outside the peak hours. More trains arrive in the shunting yard after the afternoon peak hour, which is also expected since most trains are driving throughout the day and are parked at night.

It was expected that no trains would arrive at the shunting yard during peak hours. However, as shown in Figure 8a, trains still arrive quite frequently during these times. The reason for this is unknown.

Almost all trains that are part of a train series (see Figure 8b) are trains directly after peak hours. This could be related to the fact that trains follow a specific schedule and end up in a shunting yard afterward.

Next to this, it can also be concluded that trains that are not part of a train series (arriving between 03:00:00-09:00:00 and 12:00:00-18:00:00) are less predictable and thus harder to consider with planning. Nothing can be concluded about train series leaving Amersfoort throughout the night.

In August, the train series from May to November after the morning peak is missing and fewer trains arrive. This could be due to the missing data, but could also be due to the summer break. Fewer people traveling to work leads to less maintenance and cleaning, thus fewer trains going into shunting yards. Similarly, in December and February, fewer trains arrive after the morning peak which could be due to the Christmas and spring break.

The delay difference in the summer and winter months is as expected, where trains have more delay during winter. However, the delays already start during the fall.

In May, June, July, and August, most trains have almost no delay. This could be because of various reasons; better weather conditions, fewer regular travelers due to holidays, or recent maintenance. Holidays are a popular moment for ProRail for maintenance of trains[6] leading to a more efficient infrastructure during the summer. This results in fewer disruptions and smoother train operations in shunting yards in these months.

In comparison to the summer months, in the months following, trains more often arrive 3+ minutes late. Especially in September, a lot of trains arrive around 4 minutes late. September marks the beginning of fall resulting in weather changes; more rainfall, temperature changes, and falling leaves, which could all influence the general delay of trains. This general delay continues until February.

[6] https://www.prorail.nl/veelgestelde-vragen/treinvrije-periodes/waarom-zijn-treinvrije-periodes-vaak-tijdens-de-vakantie

## 7 Conclusions and Future Work

Most trains arrive in the shunting yards after peak hours, where the busiest moment is after the afternoon peak hour. Almost all trains part of a train series were identified here as well, meaning it is easier to align the staff and equipment with the busyness of these train arrivals. The recurring train series can also be assigned to certain tracks, separating expected trains from unpredictable trains.

Fewer delays were detected from May to August and more delays were detected from September to February. This could be due to the seasons changing or variations in the number of regular travelers. Knowing this in advance, ProRail could schedule additional maintenance and deploy more staff during the months when delays are more likely.

The hypothesis aligns with the findings of this research. The outcomes are validated by actual train data, confirming that the results are factual and can be applied in real life.

This research can be expanded by including data from other (freight) trains passing the shunting yards. By expanding the dataset, the results will become more reliable, making it possible to draw definite conclusions.

In addition to expanding the dataset, future research can also be based on making the algorithm more robust for larger datasets. The current research uses a density-based clustering algorithm to identify train series based on the arrival time of trains. Future work could explore the use of deep learning for comparing the time sequences of arriving trains. For example, Recurrent Neural Networks (RNNs), which is a model that can be utilized for time series analysis while preserving information from previous time steps [1], could be applied. Using deep learning instead of a clustering algorithm would result in more accurate pattern recognition and improved train series identification.

## A Plots

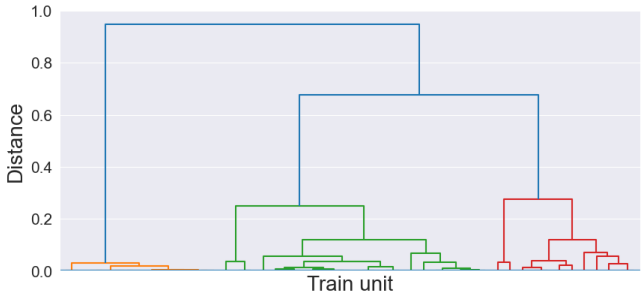
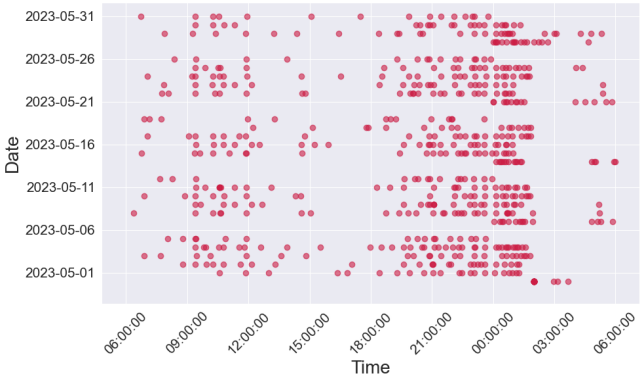
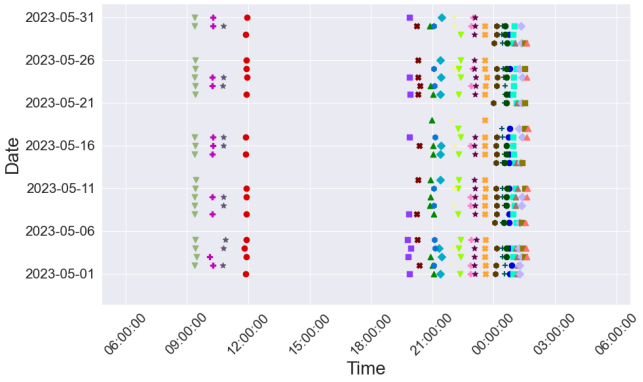


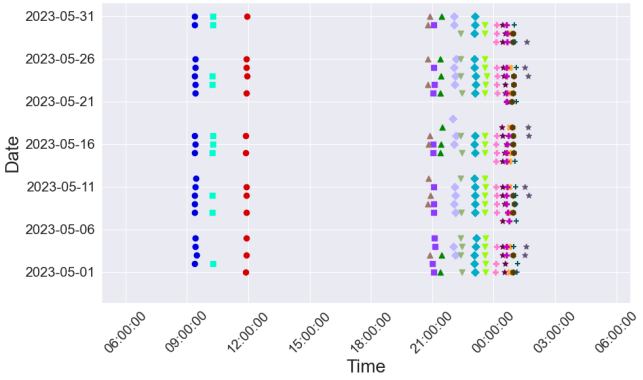

Figure 9: Hierarchical clustering shown in a full dendrogram plot of a random day (20-06-2023).




(a) Scatterplot of the arrival times.

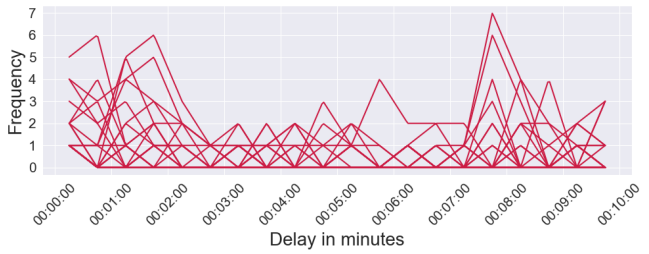



(b) Identified trains using the created clustering algorithm.

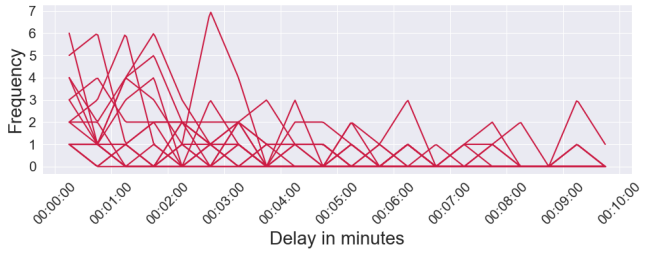



(c) Identified trains using DBSCAN.

Figure 10: Scatterplot and the identified trains for a period of one month: May.

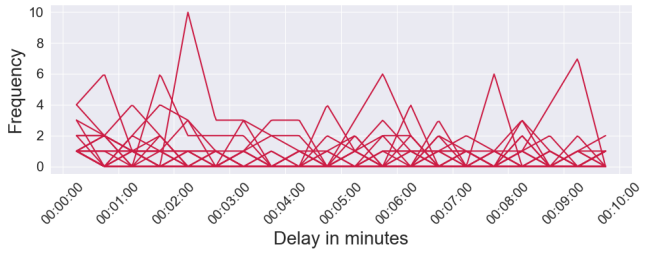



(a) Line graph distribution of the identified trains using the created clustering algorithm.

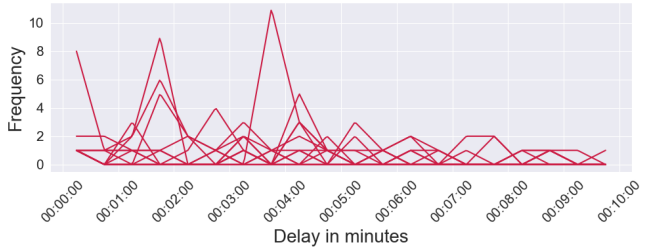



(b) Line graph distribution of the identified trains using DBSCAN.

Figure 11: Line graph distribution of the identified trains for a period of one month: May.




(a) Line graph distribution of the identified trains using the created clustering algorithm.




(b) Line graph distribution of the identified trains using DBSCAN.

Figure 12: Line graph distribution of the identified trains for a period of one month: October.

**Algorithm 1** Pseudocode of the created algorithm.

```
function FIND_CLUSTER(clusters, train)
    for each cluster in clusters do
        if train.date is not in cluster.dates
            and cluster.type is train.type then
            for each cluster_time in cluster.times do
                time_diff ← |new_time - cluster_time|
                if time_diff > 10 minutes then
                    return None
            return cluster
    return None

clusters ← []
for each arriving_train do
    current_train ← arriving_train's 'time', 'type', 'date'
    cluster ← FIND_CLUSTER(clusters, current_train)
    if cluster is None then
        cluster ← current_train
    clusters.append(cluster)
return clusters
```

## References


[1] N. Buhl. Time series predictions with RNNs. https://encord.com/blog/time-series-predictions-with-recurrent-neural-networks/, 2023.

[2] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD-96 Proceedings*, number 9, pages 226–231, 1996.

[3] R. Freling, R. M. Lentink, L. G. Kroon, and D. Huisman. Shunting of passenger train units in a railway station. *Transportation Science*, 39(2):261–272, 2005.

[4] J. T. Haahr and R. M. Lusby. Integrating rolling stock scheduling with train unit shunting. In *European Journal of Operational Research*, volume 259, pages 452–468, 2017.

[5] J. T. Haahr, R. M. Lusby, and J. C.Wagenaar. Optimization methods for the train unit shunting problem. In *European Journal of Operational Research*, volume 262, pages 981–995, 2017.

[6] A. Halvorsen, D. Jefferson, T. Stasko, and A. Reddy. Algorithm for tracing train delays to incident causes. In *Transportation Research Record*, volume 2674, pages 264–273, 2020. Publisher: SAGE Publications Inc.

[7] F. Karabiber. Hierarchical clustering. https://www.learndatasci.com/glossary/hierarchical-clustering/.

[8] S. Lehmann, A. Reddy, C. Samsundar, and T. Huynh. Automated train identification and train position monitoring at new york city transit. In *Transportation Research Record*, volume 2674, pages 843–854, 2020. Publisher: SAGE Publications Inc.

[9] NS. Over NS | Verantwoordelijkheden. https://www.ns.nl/over-ns/de-spoorsector/verantwoordelijkheden.html.

[10] NS. Thema 4 – Soorten treinen. https://www.ns.nl/binaries/_ht_1533716328163/content/assets/ns-nl/over-ons/onderwijs/thema-4-soorten-treinen.pdf.

[11] P. Pai. Hierarchical clustering explained. https://towardsdatascience.com/hierarchical-clustering-explained-e59b13846da8, 2021.

[12] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky. Automatic extraction of clusters from hierarchical clustering representations. In *Advances in Knowledge Discovery and Data Mining*, pages 75–87. Springer, 2003.

[13] A. Sharma. How to master the popular DBSCAN clustering algorithm for machine learning. https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/, 2020.

[14] P. Sharma. Introduction to k-means clustering algorithm. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/, 2024.

[15] L. C. M. van de Gevel. How human knowledge can support algorithmic decision-making in the train unit shunting problem: an exemplary study. 2022.

[16] A. Verma. Unveiling the power of DBSCAN: A comprehensive guide. https://medium.com/@ajayverma23/unveiling-the-power-of-dbscan-a-comprehensive-guide-bc954d742611, 2024.

[17] M. Çelik, F. Dadaşer-Çelik, and A. S. Dokuz. Anomaly detection in temperature data using dbscan algorithm. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 91–95, 2011.