POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
Master's degree in Mathematical Engineering

Data Analysis in Railway Descriptive Models

Supervisor: Prof. **Daniele Loiacono**
Co-supervisor: Dr. **Boubekeur Merabet**

Master's thesis by:
Alessandro Danesi
Student number: 928304

Academic year 2020-21

Ringraziamenti

Questa tesi rappresenta la fase finale di un percorso in cui si sono susseguiti tanti momenti di gioia e soddisfazione ma anche di ansia e difficoltà che mi hanno permesso di crescere molto sia dal punto di vista personale che accademico e, pertanto, voglio dedicarla alle persone che mi hanno permesso di raggiungere questo traguardo.
Ringrazio innanzitutto la mia famiglia, i miei genitori Ivan e Mirella e mio fratello Simone, i miei zii e cugini, che mi hanno saputo sostenere sempre anche nei momenti più difficili, dopo un esame andato male o un periodo di poca autostima, esortandomi a non arrendermi mai e a darmi la forza necessaria per continuare.
Vorrei ringraziare i miei relatori, Daniele Loiacono al Politecnico di Milano e Boubekeur Merabet alla SNCF Réseau, che mi hanno aiutato alla stesura di questa tesi con critiche costruttive e ottimi consigli.
Infine ringrazio i miei amici di Cesena, Milano e Parigi, con i quali ho vissuto degli anni meravigliosi e senza i quali non sarei arrivato fin qui.

Sommario

La *SNCF Réseau*, la principale compagnia ferroviaria francese che si occupa della gestione e della manutenzione della rete ferroviaria nazionale, lavora ogni giorno con una vastissima mole di dati che vengono analizzati e sfruttati per migliorare costantemente i propri servizi e infrastrutture.
Due dei principali modelli di dati utilizzati dalla compagnia sono *Réseau* e *GAÏA*. Il primo è un modello semantico basato su una visione locale e consiste in un assemblaggio di modelli locali, offrendo l'unicità dell'oggetto solo a livello di stazione o linea ferroviaria. Il secondo è un modello più recente basato su una visione globale, dove per definizione si ha l'unicità dell'oggetto a livello dell'intero modello.
Siccome *GAÏA* risulta essere un modello più completo e informativo, è stato creato il progetto *MGOC* (*Modernisation de la Gestion Opérationnelle des Circulations*, modernizzazione della gestione operativa del traffico) che ha tra gli obiettivi proprio la migrazione del vecchio modello *Réseau* a quello nuovo *GAÏA*.
Lo scopo di questo lavoro si inserisce all'interno del progetto *MGOC* e, in particolare, cerca di risolvere due importanti problematiche: la prima riguarda esclusivamente *GAÏA* e la struttura di dati sulla quale si basa. Esistono, infatti, due versioni di *GAÏA*, ovvero una più vecchia, fissa nel tempo, e una attuale, dinamica, che viene costantemente modificata con i nuovi dati. Lo scopo di questa prima parte è allora quello di creare un confronto tra le due versioni con relative corrispondenze e differenze, in modo da facilitare il passaggio di informazioni da una all'altra. La seconda problematica riguarda invece la necessità di avere un modello chiamato *bouchon* che metta in relazione *Réseau* con *GAÏA* e che permetta di avere le informazioni in modo più preciso ed efficacie a livello della singola via ferroviaria.
Infine, l'ultima parte di questa tesi è un lavoro sperimentale ed esplorativo sul completamento e l'incremento dell'affidabilità del modello *bouchon* attraverso il flusso di dati proveniente da un'altra sorgente, chiamata *X16*, la quale, in connessione con *GAÏA*, ha l'obiettivo di coprire il traffico ferroviario in tempo reale.
Questo lavoro è stato svolto nell'ambito di uno stage alla SNCF Réseau da maggio a novembre 2021.

Abstract

The *SNCF Réseau*, the main French railway company responsible for the management and maintenance of the national rail network, works every day with a huge amount of data that is analysed and exploited to constantly improve its services and infrastructure.
Two of the main data models used by the company are *Réseau* and *GAÏA*. The first is a semantic model based on a local view and consists of an assembly of local models, offering the uniqueness of the object only at the level of the station or railway line. The second is a more recent model based on a global view, where by definition there is uniqueness of the object at the level of the whole model.
Since *GAÏA* is a more complete and informative model, the *MGOC* project (*Modernisation de la Gestion Opérationnelle des Circulations*, modernisation of operational traffic management) was created and one of the objectives of it is the migration of the old *Réseau* model to the new *GAÏA* model.
The aim of this work is part of the *MGOC* project and, in particular, seeks to resolve two important issues: the first concerns only *GAÏA* and the data structure on which it is based. There are, in fact, two versions of *GAÏA*, namely an older one, fixed in time, and a current one, dynamic, which is constantly modified with new data. The purpose of this first part is therefore to create a comparison between the two versions with their correspondences and differences, in order to facilitate the transfer of information from one to the other. The second issue concerns the need to have a model called *bouchon* which relates *Réseau* to *GAÏA* and which allows to have the information in a more precise and efficient way at the level of the single railway track. Finally, the last part of this thesis is an experimental and exploratory work on the completion and increase of the reliability of the *bouchon* model through the flow of data coming from another source, called *X16*, which, in connection with *GAÏA*, has the objective of covering the railway traffic in real time.
This work has been carried out as part of an internship at *SNCF Réseau* from may to november 2021.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The railway network is composed by different "objects" (e.g. stations, tracks, etc..) described according to a data model called *Ariane* in each "data pool". The model is thus used to describe infrastructure elements as well as traffic, works, etc..
GAÏA is the repository that describes the railway infrastructure and all its components: rails, signalling elements, switches, other "track devices", etc.. Moreover, *GAÏA* includes metadata and links to external data to precisely qualify each object in the network. These include the railway location (initial and final kilometre points on the track diagram) and geo-location (spatial coordinates) of the objects, but also descriptions and documentation, traffic, status (wear and tear, repairs, scheduled or completed maintenance operations) and many other useful information.
There are two *GAÏA* versions. A first version, called Data Preparation (in short, Data Prep), was created in January 2020 and is fixed in time, i.e. it has not been updated since then. The second version, on the other hand, called Data Production (in short, Data Prod), is dynamic, i.e. it is periodically updated with the new structures and data created. However, the fact that many tools still use the first version makes it necessary to create a translation model between the two versions. This will be the objective of the first part of our work. In particular, we will try to compare between Data Prep and Data Prod a particular *GAÏA* object called **SRV** (*Système de Repérage Voie*, Track Position System), the system which describes all the tracks.
The SRV is one of the most used objects in *GAÏA* and will be constantly exploited in all our work. In particular, the SRV will also be one of the protagonist of the second part of this thesis together with the *Localisateurs* (trackers or locators). These are sensors that are physically attached to the tracks and detect useful information related to the passage of trains. Originally, the tracks identified by the locators are those associated to the *Réseau* repository. Since the *MGOC* project to which this work belongs is creating a migration operation from *GAÏA* to *Réseau*, we want to couple the trackers with the SRVs. In fact, *GAÏA* is definitely simpler than *Réseau*: it offers some new perspectives for railway export since the observations on trains are done in a more precise way at the level of tracks.
In order to do this we will create the *bouchon* file by implementing an algorithm based on a semantic comparison between the tracks from *GAÏA* and *Réseau.* the term *bouchon* (cap or cover in english) comes from the caps that are used in electronic systems. In fact, we want to cover all the associations between trackers and SRVs.

Following the creation of this model, since it is difficult to achieve 100% of correspondence between trackers and SRVs due to various problems that we are going to analyse, we need a way of completing it and making it more reliable. For this purpose we will exploit the dynamic flow of data coming from the *X16* model (see diagram in Fig. 3.1 of Chapter 3.2) which provides real time information about the train itineraries. In the same time, since this flow of data is often inaccurate or presents some missing values, we will use the *bouchon* file to adjust it. This part is just an exploratory and experimental analysis in which only a small percentage of data from X16 will be used in order to show a possible methodology that can be exploited. We will see how this method already shows some good results that will be improved in future when, in a few years, a new tool that makes use of a higher frequency of the GPS of the trains will be introduced to get more accurate data.

The thesis is structured as follows: in Chapter 2 we will focus on the first of our two tasks, namely the SRV comparison analysis between Data Prep and Data Prod. After a short introduction on the main objects of *GAÏA* and, in particular, on the SRV class (Section 2.1), we will conduct a statistical analysis of the two datasets (Section 2.2) and then we will move on the concrete phase of comparison (Sections 2.3 and 2.4). In order to have a more accurate comparison, we will deal with the classification problem of the type of track in "junctions" and "non-junctions", where we will analyse some supervised machine learning techniques (Section 2.5), and we will take advantage of unsupervised methods for clustering the tracks in groups such as the comparison can be made at the level of them (Section 2.6). In Section 2.7 we will take into account a concrete example of a railway line, specifically that one from Creil to Jeumont, two small french cities, where we will adopt all the techniques we have developed before for associating SRVs between Data Prep and Data Prod. Finally, the last part of this chapter will be devoted to the analysis of TIVs (*Tronçon d'Itinéraire Voie*, Way Section Track) that have been used during the classification problem.
In Chapter 3 we will focus on the second of our tasks: the creation of *bouchon* model. After an overview of the data structures that we will rely on, we will move firstly on the detection of the origin track (Section 3.1.1), that is the railroad way to which the tracker is attached and used for constructing the *bouchon* file, and then we will move on the detection of the destination track (Section 3.1.2), although this part is a bit out of our scope. Finally, in Section 3.2 we will tackle the completeness and reliability problem of the *bouchon* model by using some of the data coming from X16. In particular, we will analyse a dataset where we will find some anomalies (Section 3.2.1) and a dataset where we will have some missing data (Section 3.2.2). At the end, in Chapter 4 we will draw the conclusions.

Chapter 2

SRV Comparison Analysis

There are 3 main systems that describe the entire French railway network. Proceeding in order from largest to smallest we have:

- **SRL** (*Système de Repérage Ligne*, Line Tracking System): it is the system that describes all French railway lines. By definition, a line is "a railway track or a set of railway tracks with characteristics of alignment, operation and installation between a point of origin and a point of destination and on which a transport service can be provided". There are 953 lines in total and each one is identified by a series of attributes describing its name, length, identification codes etc..

- **SRV** (*Système de Repérage Voie*, Track Position System): it is the system that describes all French tracks. By definition, a track is "a physically continuous element of the network that allows for rail traffic, made up of two lines of rails". They are divided into 2 main groups: **Voies Principales** (Main Tracks), which are the tracks used for trains circulation between stations (VPL, *Voies Principales de Ligne*, Main Lines Tracks) or inside them (VPA, *Voies Principales Autres*, Other Main Tracks) and the **Voies de Service** (Service Tracks), which include all pathways that are not main tracks; these include, in particular, marshalling tracks excluding traffic ones, overflow tracks in stations, second part tracks of particular sidings, etc..
  A SRL is composed by several SRVs, which do not form the topological continuity of the line as we will see later, but they cover its entire path. A SRV belongs to an unique SRL. Like SRL, a SRV is described by several attributes that identify it (see 2.1).

- **TIV**(*Tronçon d'Itinéraire Voie*, Itinerary Section Track): it is the smallest element of the 3 described and belongs to an unique SRV. In particular, TIVs forms the topological continuity of the SRV. Thanks to TIVs we have the possibility to have more local and precise information about the network system.

In this chapter we will focus on the second element, the **SRV**. In particular, we want to make a comparison analysis between the old data system structure, the Data Prep, that is frozen in time, and the new one, the Data Prod., that is continuously updated.
In particular, our aim is to understand which are the new SRVs comparing to the old ones.

2.1 SRV: Système de Repérage Voie (Track Position System)

One of the main classes in the *GAÏA* database is the SRV class and and it will be our main protagonist in the rest of our work. Basically, it is the device that defines the location of any point on the railway network and through which we can trace which tracks make up the railway lines, where they begin and where they end, which stations connect and many other useful information.
This is one of the most important class since the new corporate repository is based *Ariane* modeling composed by attributes that allow each object to be identified unambiguously across the railway network.
The first step is to understand how this class is composed.
After a deep analysis over all the attributes that define an SRV object, the most useful ones are the following:

- **Id**: long and unique alphanumeric value identifying the SRV.
- **Libelle**: SRV name.
- **DateDebutActivite**: starting activity date.
- **DateFinActivite**: ending activity date.
- **PkDebut**: starting PK (*Point Kilométrique*, kilometric point). It is written as **XXXX00<sign>YYY**, where:
  - XXXX: number of kilometres.
  - 00: separation digits.
  - sign: empty or "-" for negative values.
  - YYY: number of metres.
- **PkFin**: final PK.
- **SrPkLigneId**: Unique SRL ID to which the SRV belongs to.

We remark that it may happen that a negative sign appears before the value of a PK: indeed, sometimes the tracks or the kilometre system of these undergo changes. For example, suppose a pathway has an initial PK of 20 metres. If, following a change of the kilometre system, the original 0 is moved 50 meters further, then the initial PK of the same track will be -30 metres.

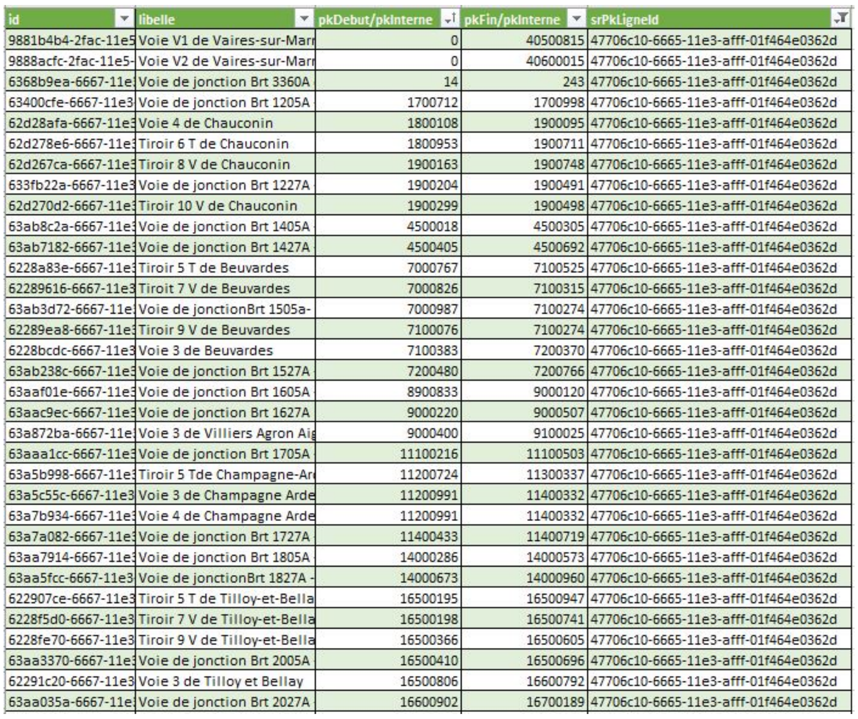
2.2 Statistical Analysis of Data Prep. and Data Prod.

Before diving into the main topic of this chapter, that is the SRV comparison between Data Prep and Data Prod, it is necessary to explore deeper our datasets. In order to deal with valid data, the first step is a **pre-processing phase**: to begin with, we will consider only SRVs with an end of activity date that is not already expired and therefore still used. Then, we remove unnecessary NaN (Not-A-Number) values and we correct values written in a wrong format.

After this first preliminary phase, we want to deepen our knowledge of the datasets by giving an answer to some questions:

1. What type of tracks we are dealing with?
2. How long are SRVs?
3. How many SRVs are located along the different railway lines?

We start by taking a random railway line and collect all SRVs composing it, ordered in ascending track for initial PK. Figure 2.1 displays part of the SRVs that compose



id	libelle	pkDebut/pkInterne	pkFin/pkInterne	srPkLigneId
9881b4b4-2fac-11e5	Voie V1 de Vaires-sur-Marr	0	40500815	47706c10-6665-11e3-afff-01f464e0362d
9888acfc-2fac-11e5-	Voie V2 de Vaires-sur-Marr	0	40600015	47706c10-6665-11e3-afff-01f464e0362d
6368b9ea-6667-11e	Voie de jonction Brt 3360A	14	243	47706c10-6665-11e3-afff-01f464e0362d
63400cfe-6667-11e3	Voie de jonction Brt 1205A	1700712	1700998	47706c10-6665-11e3-afff-01f464e0362d
62d28afa-6667-11e	Voie 4 de Chauconin	1800108	1900095	47706c10-6665-11e3-afff-01f464e0362d
62d278e6-6667-11e	Tiroir 6 T de Chauconin	1800953	1900711	47706c10-6665-11e3-afff-01f464e0362d
62d267ca-6667-11e	Tiroir 8 V de Chauconin	1900163	1900748	47706c10-6665-11e3-afff-01f464e0362d
633fb22a-6667-11e3	Voie de jonction Brt 1227A	1900204	1900491	47706c10-6665-11e3-afff-01f464e0362d
62d270d2-6667-11e	Tiroir 10 V de Chauconin	1900299	1900498	47706c10-6665-11e3-afff-01f464e0362d
63ab8c2a-6667-11e	Voie de jonction Brt 1405A	4500018	4500305	47706c10-6665-11e3-afff-01f464e0362d
63ab7182-6667-11e	Voie de jonction Brt 1427A	4500405	4500692	47706c10-6665-11e3-afff-01f464e0362d
6228a83e-6667-11e	Tiroir 5 T de Beuvardes	7000767	7100525	47706c10-6665-11e3-afff-01f464e0362d
62289616-6667-11e3	Tiroit 7 V de Beuvardes	7000826	7100315	47706c10-6665-11e3-afff-01f464e0362d
63ab3d72-6667-11e	Voie de jonctionBrt 1505a-	7000987	7100274	47706c10-6665-11e3-afff-01f464e0362d
62289ea8-6667-11e	Tiroir 9 V de Beuvardes	7100076	7100274	47706c10-6665-11e3-afff-01f464e0362d
6228bcdc-6667-11e3	Voie 3 de Beuvardes	7100383	7200370	47706c10-6665-11e3-afff-01f464e0362d
63ab238c-6667-11e	Voie de jonction Brt 1527A	7200480	7200766	47706c10-6665-11e3-afff-01f464e0362d
63aaf01e-6667-11e	Voie de jonction Brt 1605A	8900833	9000120	47706c10-6665-11e3-afff-01f464e0362d
63aac9ec-6667-11e	Voie de jonction Brt 1627A	9000220	9000507	47706c10-6665-11e3-afff-01f464e0362d
63a872ba-6667-11e	Voie 3 de Villiers Agron Aig	9000400	9100025	47706c10-6665-11e3-afff-01f464e0362d
63aaa1cc-6667-11e	Voie de jonction Brt 1705A	11100216	11100503	47706c10-6665-11e3-afff-01f464e0362d
63a5b998-6667-11e	Tiroir 5 Tde Champagne-Ar	11200724	11300337	47706c10-6665-11e3-afff-01f464e0362d
63a5c55c-6667-11e3	Voie 3 de Champagne Arde	11200991	11400332	47706c10-6665-11e3-afff-01f464e0362d
63a7b934-6667-11e	Voie 4 de Champagne Arde	11200991	11400332	47706c10-6665-11e3-afff-01f464e0362d
63a7a082-6667-11e	Voie de jonction Brt 1727A	11400433	11400719	47706c10-6665-11e3-afff-01f464e0362d
63aa7914-6667-11e	Voie de jonction Brt 1805A	14000286	14000573	47706c10-6665-11e3-afff-01f464e0362d
63aa5fcc-6667-11e3	Voie de jonctionBrt 1827A -	14000673	14000960	47706c10-6665-11e3-afff-01f464e0362d
622907ce-6667-11e3	Tiroir 5 T de Tilloy-et-Bella	16500195	16500947	47706c10-6665-11e3-afff-01f464e0362d
6228f5d0-6667-11e3	Tiroir 7 V de Tilloy-et-Bella	16500198	16500741	47706c10-6665-11e3-afff-01f464e0362d
6228fe70-6667-11e3	Tiroir 9 V de Tilloy-et-Bella	16500366	16500605	47706c10-6665-11e3-afff-01f464e0362d
63aa3370-6667-11e	Voie de jonction Brt 2005A	16500410	16500696	47706c10-6665-11e3-afff-01f464e0362d
62291c20-6667-11e3	Voie 3 de Tilloy et Bellay	16500806	16600792	47706c10-6665-11e3-afff-01f464e0362d
63aa035a-6667-11e	Voie de jonction Brt 2027A	16600902	16700189	47706c10-6665-11e3-afff-01f464e0362d

Figure 2.1: Part of the SRVs of the Data Prep. forming the railway line from Paris to Strasbourg.

the railway line that goes from Paris to Strasbourg.

Firstly, we can notice that there are 2 principal railroad ways "Voie V1 de Vaires-sur-Marne" and "Voie V2 de Vaires-sur-Marne" that are long almost 400 km (that is the distance between the 2 cities). This is not so surprising: it is normal for a railway line to be composed of main and secondary tracks. In support of this argument, it can be observed that a large part of the SRVs composing this line are junctions (see the "jonction" word in the feature "libellé"). In general, as we can see from this table, junctions are very short tracks connecting other railroad ways.

Another remark one can state is that there are some repeated track names, like, for instance, "Voie 3" or "Tiroir 5 T". This is an important consideration to take in mind when we will attempt to compare SRVs from Data Prep and Data Prod, in order not to associate SRV with the same name but representatives of different tracks.

Finally, but not less important, notice that SRVs do not form the kilometre continuity of the railway line; this means that there are overlap between SRVs, mainly

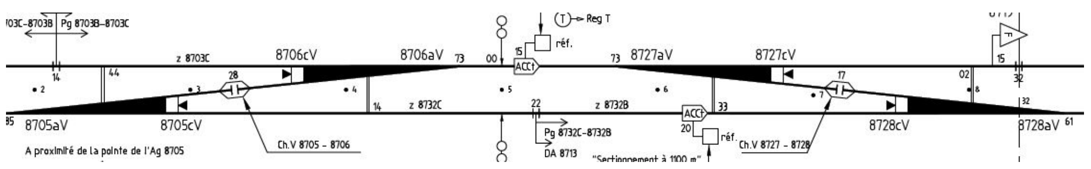


Figure 2.2: Real scheme of 2 junctions and 2 tracks at Dijon control station.

between tracks and junctions or between the 2 main pathways and the others.
To better understand this concept, consider the Figure 2.2.

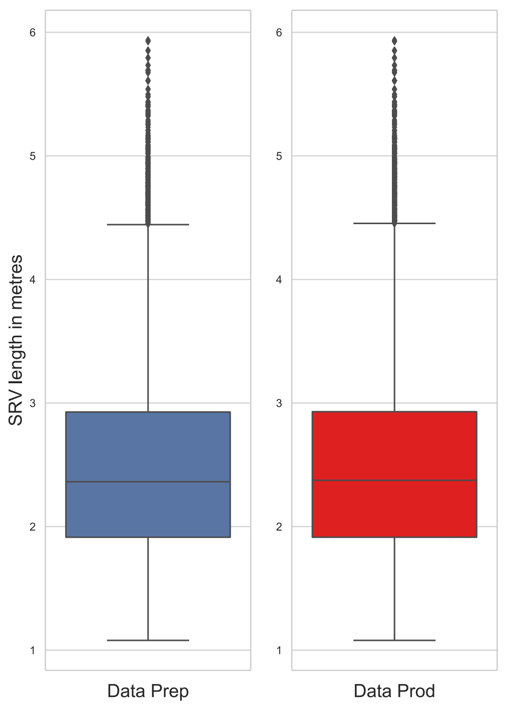
In this figure there are 2 junctions, represented with oblique lines, that connect 2 tracks, represented with horizontal lines. At the center there are some black dots with numbers on the side, indicating the number of meters (we are at PK 48, which is omitted in this figure due to size issues). It is evident then that the PK of SRVs can overlap, depending on the structure of the railway line.

After the over mentioned considerations, we can distinguish 3 type of tracks: junctions, medium length tracks and long length tracks. However, this is generally true for medium-long railway lines, whereas most of the lines (as we will see at the end of the chapter) are provincial lines consisting of a few SRVs and therefore the distinction into the three classes becomes more subtle.
For this reason in the following we will consider a binary classification problem: **junctions** vs **non-junctions**.
In order to analyze their distribution and try giving an answer to the second of our questions, we created the 2 classes in a deterministic way, following a simple algorithm (Alg. 1) where junctions are detected by looking for the word *jonction* (or similar) in the SRV name. In the Chapter 2.5 we will address the supervised

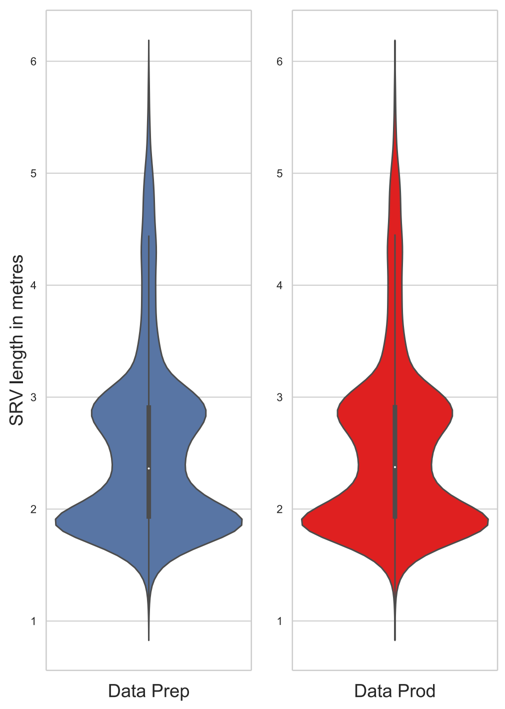
Algorithm 1: Deterministic Type Classification

```
input  : D - set of SRVs
output: class labels
1 for i ∈ D:
2     if i name contains the word "jonction":
3         i is a junction;
4     else:
5         i is NOT a junction;
```

classification problem in order to separate these 2 classes by considering the SRV lengths and the number of TIVs (track segments) composing such SRV.
From here on out we will show results obtained by classifying the type of tracks in a deterministic fashion.
In Fig. 2.3 there are two boxplots and violin plots in log scale (for better visualization) of SRV lengths for both Data Prep and Data Prod.
Firstly, we can state that there are no evident differences between the two datasets: in both cases the median is almost centrally located within the box between the first and the third interquantile (but recall that we are in log-scale!). However, the presence of many outliers in the higher whisker, represented with circles in the boxplots, makes the distribution a bit asymmetric. Indeed, as we have already mentioned, there are several principal tracks that are much longer than the others and that are



(a) Boxplots



(b) Violin plots

Figure 2.3: Log-scale Boxplots and Violin plots of SRV lengths for Data Prep and Data Prod.

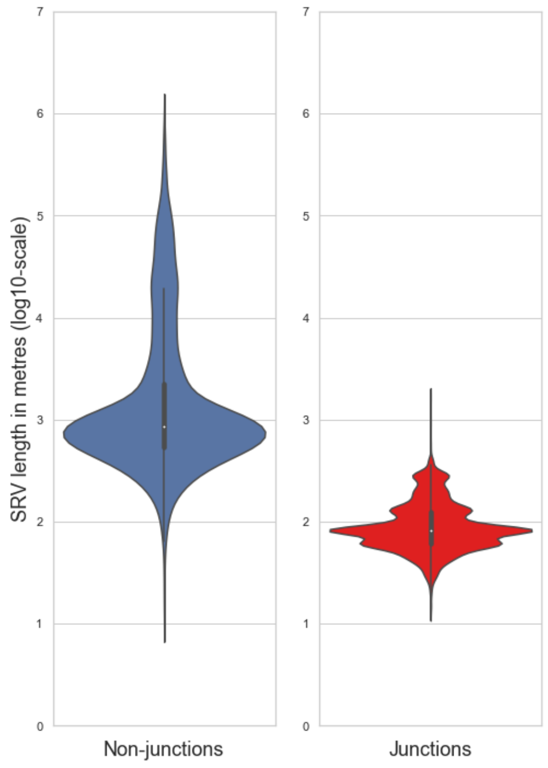
then detected as outliers.
The violin plots, compared to the boxplots, provide as additional information the distribution of SRV lengths. Even here there are no particular differences between Data Prep and Data Prod and we can conclude that we have two peaks of frequencies for "short" (mostly junctions) and "medium" length SRVs at around 90 and 900 metres respectively.
In the following table we have just collected some useful statistics, that are very similar between the two datasets due to the considerations already done:

	Data Prep	Data Prod
Junctions	5252	5205
Non-Junctions	5010	4972
Mean	5312.91	5342.70
Std	31231.43	31212.57
Min	12.00	12.00
25%	82.00	82.00
50%	230.50	237.00
75%	846.00	851.00
Max	861540.00	861540.00

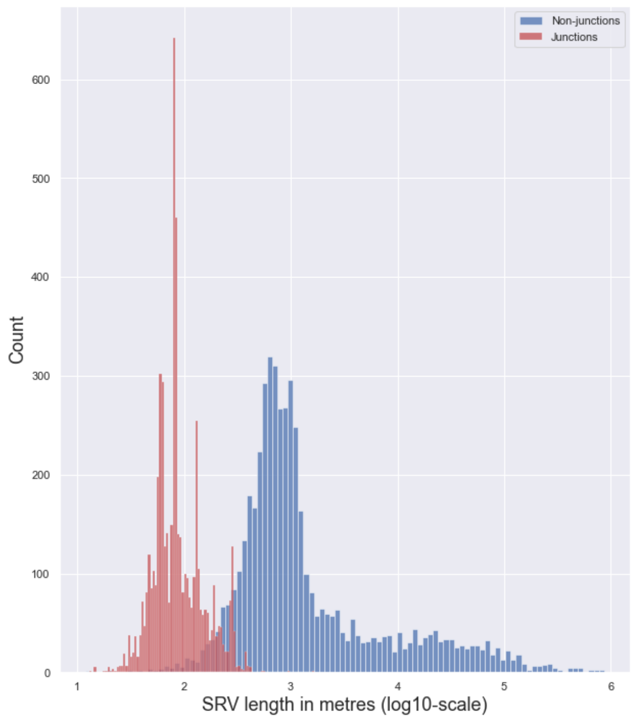
Table 2.1: SRV lengths main statistics between Data Prep and Data Prod.

Notice the huge difference between the median and the mean due to an enormous variability in the data and otuliers. In addition, we can observe the strong presence of junctions, which cover about half of the total of SRVs.

In the Figure 2.4 we have both the violin plots and distributions of the SRV lengths between junctions and non-junctions (we have taken into account only Data Prep since, as we have seen, the distributions are almost the same). Here we find again, in



(a) Violin plots



(b) Histograms

Figure 2.4: Log-scale Violin plots and Histograms of SRV lengths between junctions and non-junctions.

more detail, some patterns that we noticed in the previous graphs. We observe that there are some notable differences between the two sets: junctions have a narrower distribution (i.e., less variance) and lower average length values than the other tracks (with the peak around 90 meters); non-junctions, on the other hand, present a more variable distribution with a peak density around 900 meters.
Here some statistics that explain in more details these considerations:

	Junctions	Non-junctions
Mean	103.20	10774.28
Std	70.11	44045.70
Min	13.00	12.00
25%	61.00	540.00
50%	82.00	867.00
75%	125.00	2260.75
Max	1671.00	861540.00

Table 2.2: SRV lengths main statistics between junctions and non-junctions.

Finally, in order to answer the last of our questions, namely how many SRVs are distributed between the different railway lines, we collect the number of SRVs for each railway line in a list and we plot both a violin plot and a boxplot (for the same reasons explained before we consider only the Data Prep).

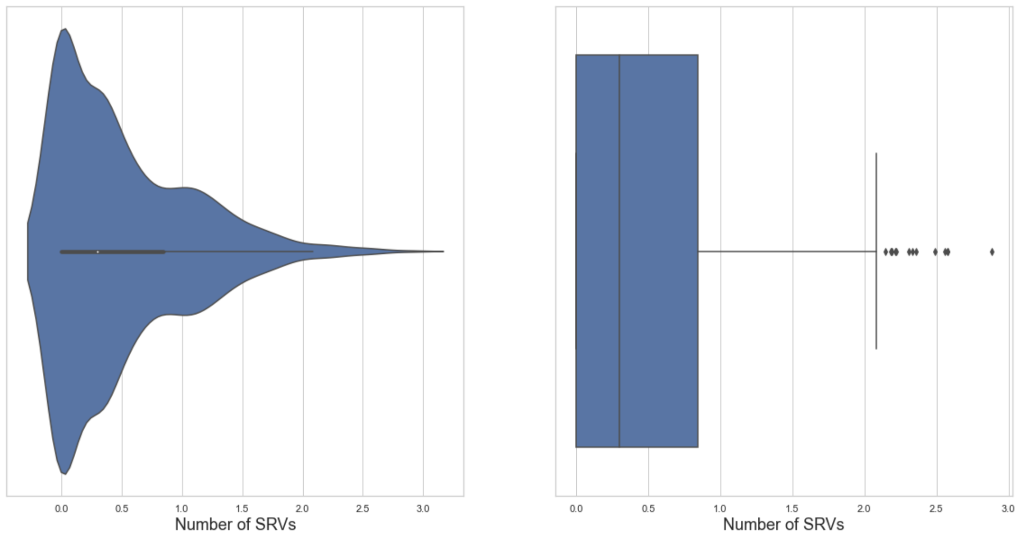


Figure 2.5: A violin plot and a boxplot in log-scale on the number of SRVs in railway lines for Data Prep.

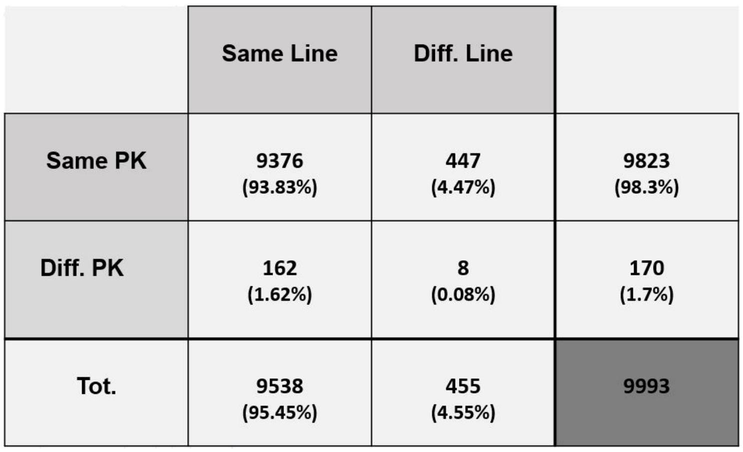
From these plots we can observe that most of railway lines have just few SRVs. Indeed, addional analysis carried out that 80% of railway lines have less than 10 SRVs and, in particular, 36.62% only one. In fact, most railway lines are short and connect provincial cities, which are much more numerous than big cities. The principal railway lines, like for instance the line Paris-Marseille, are represented as outliers in the boxplot, since they are few and with many SRVs.

2.3 Preliminary Comparison Phase

Our goal is to associate each SRV of the Data Prep to his corresponding SRV in the Data Prod and find out eliminations or new creations.
A first approach might be using the "libellé", namely the name of the SRV, but this arises some ambiguities: in fact, after a first sight to the datasets, it can be noticed that the name of tracks are often repeated. In general, all railway lines have few tracks whose names are often the same: "voie 1" (where "voie" means *track*), "voie 2", ecc.. Moreover, there are some ambiguous pathways that are frequently used, like for example "voie unique". For these reasons the variable "libellé" alone is not particularly meaningful.
Another possible idea is to use the PK, the kilometric point, to find the position of our SRVs. However, we have to discard this idea as well, since the kilometre system is not unique. Indeed, in general, the PK = 0 corresponds to Paris, and all rail lines starting from this point consequently starts from 0 and it results that PKs are repeated between the tracks.
Then, the best basic approach we can adopt for the moment is to use the ID that identifies uniquely a SRV. Indeed, thanks to the ID we can not have ambiguities. However, even if 2 SRVs share the same ID, this does not imply that the other attributes are the same. In fact, a SRV can be longer or shorter, it can have a

different name or it can belong to a different railway line. Moreover, we must take into account the fact that a track might be deleted or created in the new system.

The table in Figure 2.6 summarizes the number of changes mentioned above by taking into consideration SRV with the same ID. By analysing this table, we can



	Same Line	Diff. Line	
Same PK	9376 (93.83%)	447 (4.47%)	9823 (98.3%)
Diff. PK	162 (1.62%)	8 (0.08%)	170 (1.7%)
Tot.	9538 (95.45%)	455 (4.55%)	9993

Figure 2.6: A summary table comparing SRV between Data Prep and Data Prod by selecting the same ID

firstly notice that since our datasets are composed by 10263 and 10186 SRVs respectively for Data Prep and Data Prod and since there are 9993 pairs that have been found by combining the two datasets, it turns out that there are 270 unique SRVs in Data Prep and 193 unique SRVs in Data Prod, so most of the SRVs preserve their ID.
This table shows the number of SRVs with same ID and belonging to the same or a different railway line and having same or different PKs, that means that the initial or final PK is modified.
It can be noted that most of the SRVs are identical (93.83%) in line and PKs, but there are some exceptions. In particular, it is interesting that 455 SRVs have changed line. A deeper analysis has carried to find out that there have been 13 railway lines that are changed, or better, that have a different SRL ID, the unique identifier for the rail line, out of a total of 953 lines.

From the next chapter onwards we will analyse SRVs grouped by railway lines, as this facilitates comparison both from a practical and visual point of view.

2.4 Comparison Tool Plot

It would be nice to have a tool that allows the user to immediately visualize how the SRVs are displaced along the railway line, for both Data Prep and Data Prod, and observe which are the differences between the 2 datasets in terms of ID and length. For this purpose, we created a Python plot where railway lines and tracks are represented as segments of length [Initial_Pk, Final_Pk] on top of each other. In Fig. 2.7 there is an example of the railway line that goes from Dole-Ville to Belfort.

In the centre the railway line is represented in black.

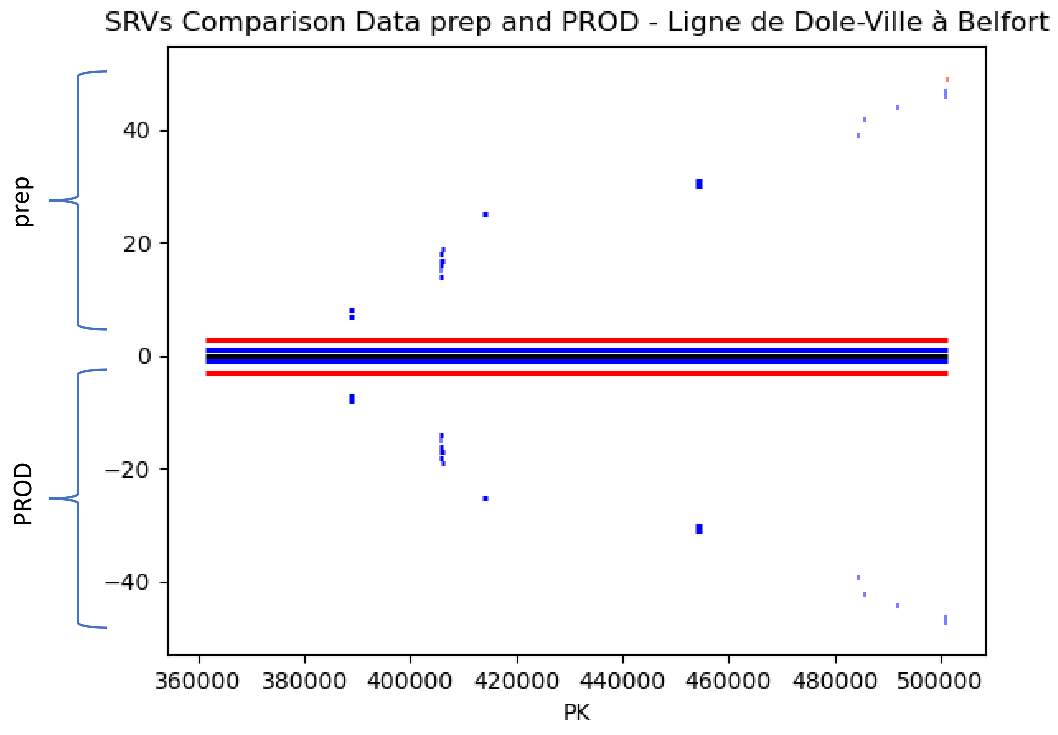


Figure 2.7: SRVs comparison between Data Prep and Data Prod of the railway line from Dole-Ville to Belfort.

The SRVs of Data Prep are represented in the half above, while those of Data Prod are represented in the half below.
SRVs are colored in blue when they share the same ID between the two datasets, in red when the ID is different. In this example, there is a long SRV that changes ID from Data Prep to Data Prod. Moreover, less evidently, there is another short SRV at the end of the line that is represented in red in Data Prep. However, this track is not presented in Data Prod, meaning that has been removed in the new data system for this particular railway line.
A common pattern that can be observed not only in this example but more generally in all railway lines with a sufficient number of SRV is that there are some main tracks that follow the entire railway line, and other shorter SRVs that are displaced all along the line. Most of the time these seem to be clustered together, probably close to stations or "collection points". We will focus more on this part on the chapter 2.6 devoted to the clustering analysis.
Also, as already said before, SRVs do not form the kilometre continuity of the railway line: in fact, there are overlap between SRVs, mainly when they are in a "cluster".

Thanks to this plot we have a fast tool to visualize the disposition of SRVs along a railway line, to understand how many and how long they are and a way to compare them between the two datasets. However, this is not enough, because we cannot easily recognize, especially for long railway lines, whether a SRV has a different ID or it is just removed from Data Prep or just created in the Data Prod. Moreover, we cannot get further information about the differences of SRVs lengths and their

type.
For this purpose, we created a table together with this plot (as output of the same Python code) that provides all the mentioned information.
In the next two sections we present the main keys we exploited in the creation of the table: the type classification and the clustering.

2.5 Track Type Classification

In Section 2.2 we have already discussed the classification of the SRV type in 2 classes: junctions and non-junctions.
In that occasion, the classification was made by following a deterministic algorithm that relies on the presence of the word "junction" in the SRV name (see Alg. 1).
In this section we go a little further by exploiting supervised learning techniques [Bishop 2006]. Indeed, sometimes we rely on tools that exploit the geographical structure of tracks and we don't have access to the SRV name, but only to the initial and final PK and to the number of segments (TIVs) that compose it. Since our deterministic algorithm fails in this case, how could we classify a track?
We will use this information to try to create a good classifier that can recognise whether a track is a junction or not, taking into account its length and how many TIVs it consists of. We have already discussed about the distribution of lengths between junctions and non-junctions (see Fig.2.4) and we have seen that there is a significant difference between the two sets. We analyze now the distribution of TIVs in the two classes. The histogram in Fig.2.8 clearly shows that junctions consist of only one or a few sections, whereas non-junctions have a wider distribution.

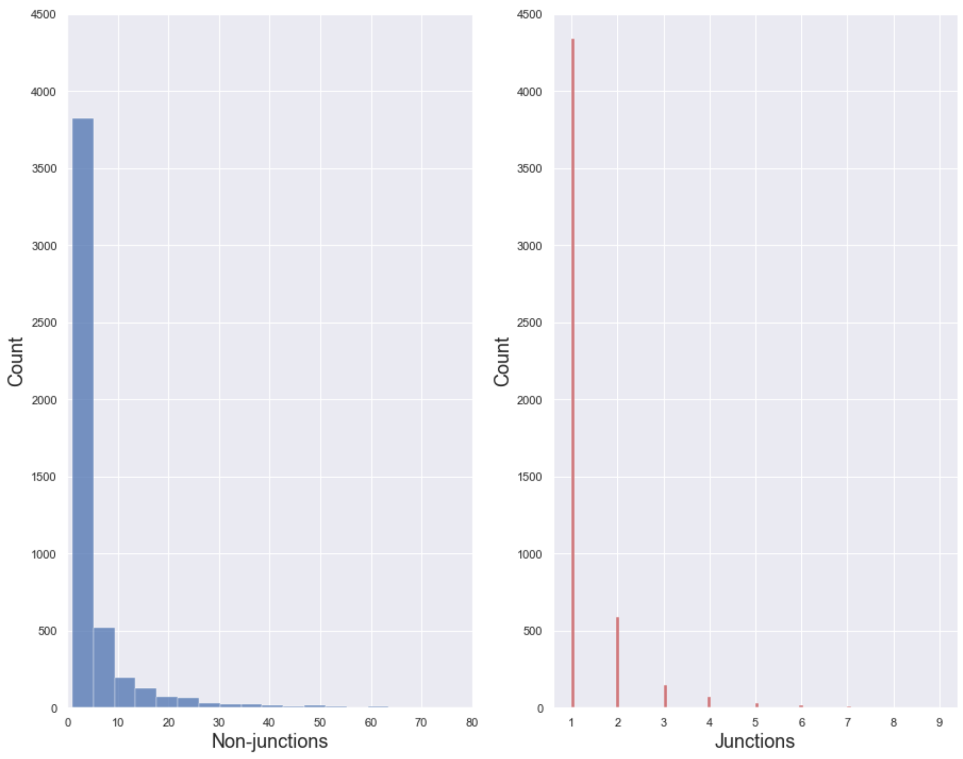


Figure 2.8: Histograms on the number of TIVs composing SRVs between junctions and non-junctions for Data Prod.

Here some more precise statistics:

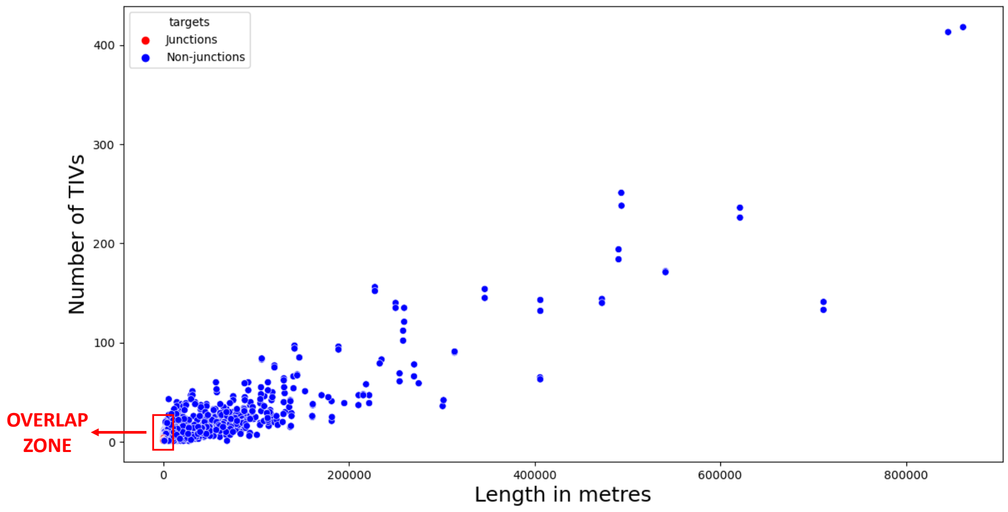
	Junctions	Non-junctions
Mean	1.26	6.20
Std	0.74	16.31
Min	1.00	1.00
25%	1.00	1.00
50%	1.00	3.00
75%	1.00	5.00
Max	9.00	418.00

Table 2.3: Main statistics on the number of TIVs composing a SRV.

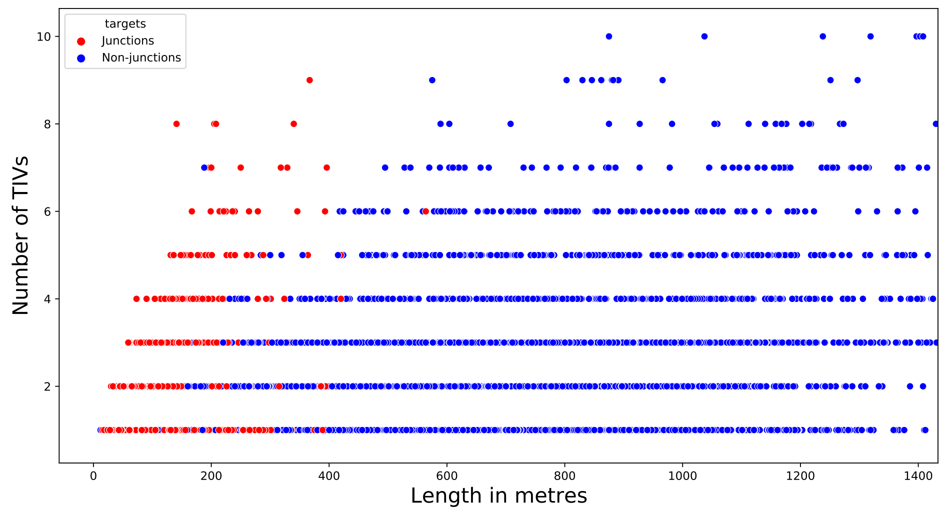
We can observe that "non-junction" class has a much larger standard deviation and, in average, longer tracks. We finally plot all our data (Fig.2.9).
Notice that the presence of several "outliers", which are the main tracks that make up the French rail network, leads to some problems in visualizing the junction class in the overall figure. For this reason we also inserted a zoom plot, where we can visually distinguish the two classes. We point out that, although the two groups seem to be numerically different, they have almost the same number of elements (Table 2.1). Moreover, we remark that the variable scales are have been left unchanged, both in the plots and algorithms. In fact, both following normalisation and not, we obtained very similar results.
The methods we have considered for our classification problem are:

- Naive algorithm.
- KNN (K-Nearest Neighbors).
- Logistic Regression.

The Naive algorithm simply consists in associating the new SRV of the test set (with coordinates his length and the number of TIVs composing it) to the class with the nearest median. The reason for using the median is that it is much more robust to the variance of the data, which is very large in the non-junction group as we have already observed.
The other two methods, i.e KNN and Logistic Regression, are widely used classification models that best performed for this problem among all those tested.
Before proceeding to analyse the performance of the three methods, we would like to devote the next two subsections to a more detailed analysis of how the last two machine learning techniques work.



(a)



(b)

Figure 2.9: (a) Overall plot of lengths and number of TIVs for all SRVs in Data Prod. divided by type class (b) Zoom over the "overlap zone" where I have all the junctions and part of the non-junctions.

2.5.1 KNN: K-Nearest Neighbors

K-Nearest Neighbour is a supervised machine learning algorithm, which means that we already know how many and which classes we are dealing with and we want to classify the new data or case based on a similarity measure after having stored all the available cases. It is mostly used to classify a data point based on how its neighbours are classified, but it can also be adopted for regression (that is not our case).
The basic idea behind KNN is simple: similar things exist in close proximity. In other words, similar things are near to each other.
There are many distance metrics that can be exploited by the algorithm (Euclidian, Manhattan, Cosine, Jaccard etc..). In the following we will use the most known one, the Euclidean distance , defined as:

$$d(x, y) = \sum_{i=1}^{2} |x_i - y_i|^2$$

where x and y are 2 dimensional vectors for our classification problem.
In order to deeper understand how KNN works, we will adopt a probabilistic approach. Suppose that we have a dataset of N points and C_k classes, with N_k points for each class such that $\sum_k N_k = N$. In order to classify a new point x we considered a sphere centered on x containing precisely K points irrespective of their class. Suppose this sphere has a volume V and contains K_k points from class C_k. Then, we can estimate the density associated with each class as:

$$p(x|C_k) = \frac{K_k}{N_k V}$$

In a similar way, the unconditional density will be:

$$p(x) = \frac{K}{NV}$$

while the class priors are given by

$$p(C_k) = \frac{K_k}{N}$$

Finally, using Bayes' theorem, we can obtain the posterior probability of class:

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{K_k}{K}$$

Since we want to minimize the misclassification error, we assign the test point x to the class having the largest posterior probability, namely with the highest value of $\frac{K_k}{K}$. In other words, we assign x to the class having the largest number of representatives among this set.
For an illustrative example, consider Fig.2.10.

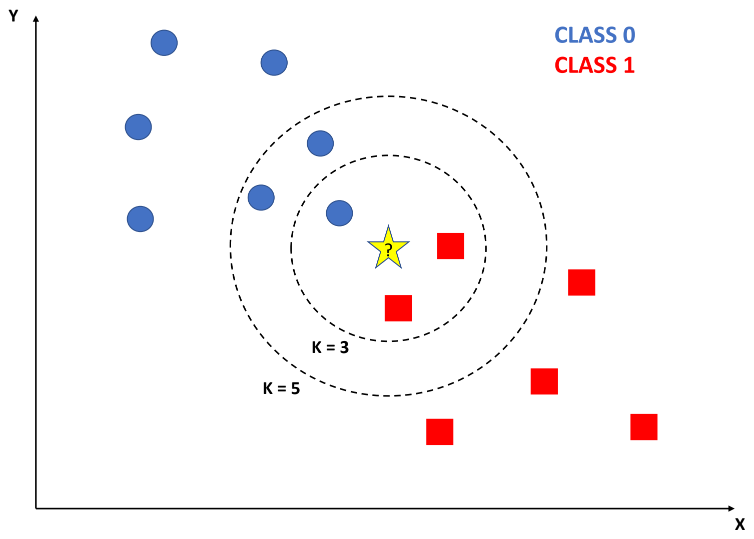
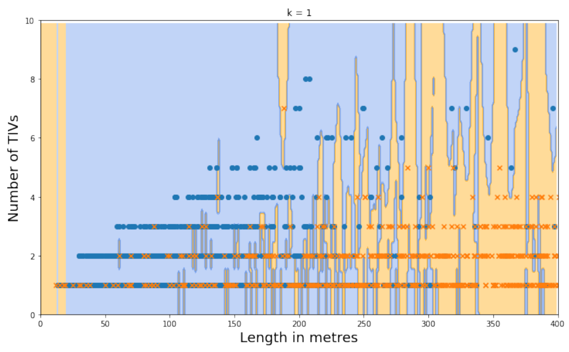


Figure 2.10: An illustrative example in order to understand the KNN algorithm for classification problem. If we set $K = 3$ we assign the new test point to the red squared class since there are 2 points belonging to this class against 1 of the blue circle class. Instead, if we set $K = 5$ we assign the test point to the blue circle class since now we have more neighbours belonging to this class than to the red squared one.

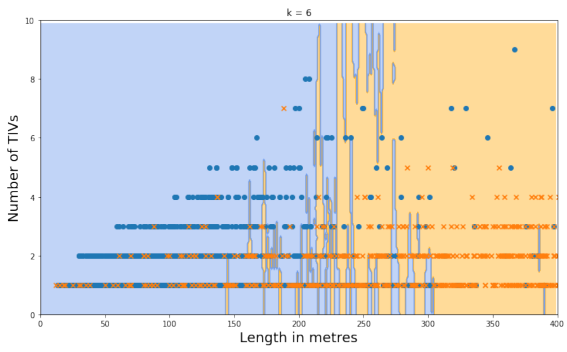
Notice that the number of neighbours K to be considered, that must be defined a priori in our model, influences the assignment of the test point to the best fit class. In fact, in the figure, if we consider 3 neighbours we classify the yellow star, our new test point, to the group of red squares, whereas if we consider 5 neighbours, we classify the new point to the blue points class.

In Figure 2.11, we show the results of applying the K-nearest-neighbour algorithm to our specific classification problem junction vs non-junctions for various values of K. As expected, we see that K controls the degree of smoothing, so that small K produces many small regions of each class, whereas large K leads to fewer larger regions.

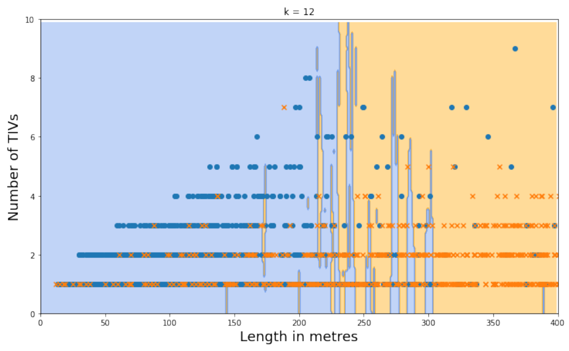
In order to choose the best K we proceed as in Algorithm 2, where we used the K-Fold Cross-Validation method (or better, we called it M-Fold in the algorithm to avoid confusion with the number of neighbours K).
In Figure 2.12 we have a plot of all accuracy (test, training and Cross-Validation accuracy) by setting different values of K and we picked up the best score for the CV-accuracy, i.e. 0.9592, corresponding to $K = 9$.
Notice that Cross-Validation accuracy is smaller than the others since we are training on a smaller dataset. Moreover, observe that if K is small, the model tends to overfits the train set (high training accuracy, but small test accuracy) and incurs in a large variance. When K assumes bigger values instead, the train accuracy decreases while the test accuracy increases and they become similar.
We remark that KNN algorithm does not involve a real training phase, but simply arranges the data in a sort of indexing process in order to find the closest neighbors efficiently during the inference phase. Otherwise, it would have to compare each new case during inference with the whole dataset making it quite inefficient.



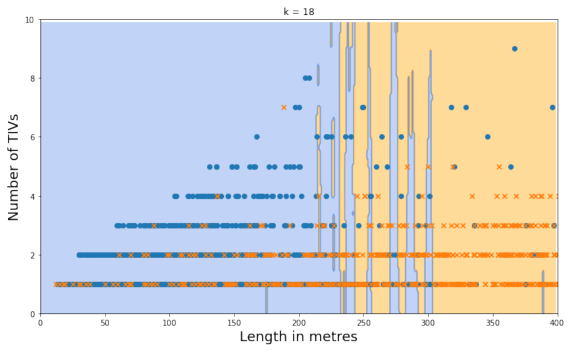
(a) $K = 1$



(b) $K = 6$



(c) $K = 12$



(d) $K = 18$

Figure 2.11: Plot of KNN for different values of K for the junction and non-junction classification problem.

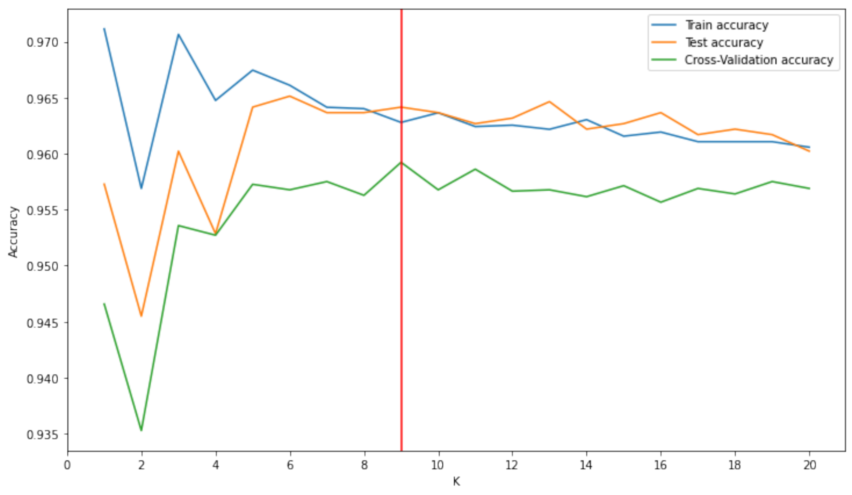


Figure 2.12: Plot of all KNN algorithm accuracy by setting different values for the number of neighbours K. The red vertical line corresponds to the best score for Cross-Validation ($K = 9$).

Algorithm 2: Choice of best K in K-NN with Cross-Validation

```
input : $D$ - training set of observations $x \in D$ of dimension $N$
        $t$ - training set of targets
        $K\_max$ - maximum value of $K$ we use as parameter for training
KNN.
output: $best\_k$ - best $K$
1  foreach $k \in range(K\_max)$:
2      Compute the KNN classifier setting $K = k$;
3      Split the training data into $M$ folds: $D_1$, ..., $D_M$;
4      foreach $i \in range(M)$:
5          Train the model on $D$ - $D_i$;
6          Compute accuracy on $D_i$: $L_{D_i} = \frac{M}{N} \sum_{(x_n,t_n) \in D_i} (t_n - y_{D-D_i}(x_n))^2$;
7      Compute the mean of accuracy: $L_{M-fold} = \frac{1}{M} \sum_{i=1}^{M} L_{D_i}$ ;
8      Add $L_{M-fold}$ to a list of scores;
9  Plot and find the best $K$ linked to the best score;
10 return $best\_k$;
```

2.5.2 Logistic Regression

Logistic Regression is a statistical model very popular for supervised machine learning tasks that, despite his name, is mainly used for classification problems. In particular, we will rely on the Binary Logistic Regression since we want to predict a binary categorical variable (junction/non-junction).
Logistic Regression is based on the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ exploited for modelling the posterior probability of class C_1 as:

$$p(C_1|\phi) = \frac{1}{1 + e^{-w^T\phi}} = \sigma(w^T\phi),$$

where $\phi : x \mapsto \phi(x)$ is any transformation (also non-linear) from the input space of the dataset to the feature space and w is the vector of model weights ($p(C_2|\phi) = 1 - p(C_1|\phi)$).
Then, given the dataset $D = \{x_i, t_i\}$, with $i = 1, ..., N$, where t is the vector of labels ($t_i \in \{0, 1\}$), we want to maximize the likelihood, i.e. the probability to observe the targets given the inputs, using the Bernoulli distribution of parameter $p(C_1|\phi) = y_i$:

$$p(t_i|x_i, w) = y_i^{t_i}(1 - y_i)^{1-t_i}$$

By taking the negative log-likelihood and considering our data independent we obtain the **cross-entropy loss**:

$$L(w) = -\ln(p(t|X, w) = -\sum_{n=1}^{N}(t_n \ln(y_n) + (1 - t_n)\ln(1 - y_n)) = \sum_{n=1}^{N} L_n$$

Finally, by computing the derivative of L_n with respect to w we have:

$$\nabla L = \sum_{n=1}^{N}(y_n - t_n)\phi_n$$

Since our problem is convex, we can adopt a gradient-based optimized technique to solve it.
Notice that if we define:

$$\text{logit}(y_i) = \log\left(\frac{y_i}{1-y_i}\right)$$

we get:

$$\text{logit}(y_i) = w^T x$$

We have the same statistical characterization of the parameters w that we have in linear regression if we consider as output the logit transformation of the target. In this way, we are also able to perform hypothesis tests on the significance of parameters w.

2.5.3 Comparison of Classification Models

After having introduced the three main models used to solve the classification problem on the type of track (junction or non-junction) we want to compare their performances by computing confidence intervals for accuracy.
A robust way to calculate them for machine learning algorithms is to use the **bootstrap**. This is a general technique for estimating statistics that can be used to calculate empirical confidence intervals, regardless of the distribution of skill scores (e.g. non-Gaussian).
We proceed as follows:

1. Split the dataset in training set (80%) and test set (20%) with resampling.
2. Train K-NN (with $K = 9$) and Logistic Regression over the training set.
3. Compute the median of each class for the naive model.
4. Compute models accuracy with the test set.
5. Go to 1 and repeat these steps several times (e.g. 500 times).

Finally we compute the confidence intervals for each of the three models accuracy. This is done by first ordering the scores, then selecting values at the chosen percentile for the confidence interval (that we call α). For example, we are interested in building a confidence interval of 95%, which is probably the most popular case. Then, we set α at 0.95 and we select the value at the 2.5% percentile as the lower bound and the 97.5% percentile as the upper bound on accuracy.
In this way, we are calculating a non-parametric confidence interval that does not make any assumption about the functional form of the distribution of the statistic, and for this reason is also called empirical confidence interval.
These are the results:

Model	**Mean**	**Std. Dev.**	**CI 95%**
Naive	0.905	0.00575	$[89.4\%, 91.6\%]$
9-NN	0.959	0.003	$[95.1\%, 96.7\%]$
Logistic Regr.	0.945	0.00462	$[93.6\%, 95.4\%]$

Table 2.4: Empirical confidence intervals of three model accuracy for type classification problem.

From these results we can conclude that all three models perform well in classifying the type of track by knowing its length and how many sections (TIVs) it consists of. However, thanks to the two simple machine learning models used, namely K-Nearest-Neighbours and Logistic Regression, we can increase the accuracy respectively by 5.4% and 4% on average compared to the naive model.

2.6 SRV Clustering

As we have already discussed, in the table in Fig.2.6 we considered the same SRV ID to compare the tracks in the Data Prep and Data Prod and we divided them taking into account differences or similarities in the railway line they belong to and their PK.
However, there are some SRVs that have an unique ID in both the datasets and that are not present in the table. For these SRV we can not rely on neither the name nor the PK as they are not unique. However, we know that the PK is unique within a railway line and using it combined with the name we can find a track to associate SRVs even if with different ID.
The idea is based on **clustering**. By taking a closer look to Fig.2.4, we can notice that SRVs seem to be clustered based on initial and final PK.
Hence, we can find groups of SRVs in both Data Prep and Data Prod and compare them: even if the SRV IDs are different within the same cluster, we can find a match using the name. In addition, the clustering helps the process of comparison, also from a logical and visual point of view, managing thus to give an answer to the question: "Which group of SRV becomes which in the new data system (Data Prod)?"
Before showing a concrete example (see Chapter 2.7), we want to focus on the clustering techniques, unsupervised machine learning methods [Bishop 2006], that we adopted for our problem: K-Means and DBSCAN.
Before proceeding, we want just to remark that clustering methods have been used for all railway lines with a sufficient number of SRVs (e.g. 50 SRVs), otherwise it doesn't make really sense to apply it for our purpose, since we can directly compare all SRVs of the two datasets.

2.6.1 K-means

K-Means clustering is an unsupervised learning algorithm, which groups the unlabeled dataset into different clusters by minimizing the internal distance between points of the same cluster. Here, K is the number of pre-defined clusters that need to be created in the process. Given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a d-dimensional real vector, this algorithm aims to partition the n observations into $k(\leq n)$ sets $C = C_1, C_2, \ldots, C_k$ so as to minimize the Within-Cluster Sum of Squares (WCSS), defined as:

$$\text{WCSS}(C) = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, \mu_i) \tag{2.1}$$

$$C^\star = \arg\min_{C} \text{WCSS}(C) \tag{2.2}$$

where μ_i is the cluster centroid of points in C_i, computed as:

$$\mu_i = \underset{\mu \in \mathbb{R}^d}{\arg\min} \sum_{x \in C_i} d^2(x, \mu) \tag{2.3}$$

where d is a distance.
The centroid is strictly related on how we measure dissimilarities: if d is the Euclidian distance, defined as $d(x, \mu_i) = \|x - \mu_i\|^2$, it coincides with the sample mean of the cluster, otherwise it might have a different interpretation.

The idea behind K-Means clustering is an iterative greedy approach: firstly, the clusters centroids are initialized following a criteria (e.g. randomly); then, iteratively each observation is assigned to the nearest centroid, which are updated by following again eq. 2.3, and this process goes on until convergence (see Alg. 3).

Algorithm 3: K-Means Algorithm

```
input : D - set of observations $x \in D$
        k - number of resulting clusters
        $\epsilon > 0$ - convergence tolerance
output: C - set of clusters
1  $t = 0$;
2  Initialize k centroids: $\mu_1^0, \mu_2^0, \dots, \mu_k^0 \in \mathbf{R}^d$;
3  while $\sum_{i=1}^{k} \|\mu_i^t - \mu_i^{t-1}\|^2 > \epsilon$:
4      $t = t + 1$;
5      $C_j = \emptyset \quad \forall j = 1, \dots, k$;
       /* Cluster assignment step                         */
6      foreach $x \in D$:
7          $j^\star = \arg\min \|x - \mu_i^{t-1}\|^2$;
8          $C_{j^\star} = C_{j^\star} \cup x$;
       /* Centroid update step                            */
9      for $i$ from 1 to $k$:
10         $\mu_i^t = \frac{1}{|C_i|} \sum_{x \in C_i} x$;
11 $C = \{C_i\}_{i=1}^k$;
12 return $C$;
```

The complexity of this algorithm is $O(nkdi)$ where n is the number of d-dimentional observations, k is the number of clusters and i is the number of iterations needed until convergence.

2.6.2 How to choose K

In order to apply the K-Means clustering we have to provide as input the number K of clusters we want to get. Since we are in an unsupervised setting, how to choose this parameter? This is the most important part of our algorithm.
The principal methods are:

- Elbow method
- Silhouette method

We start from the first one, the **Elbow method**. This is probably the most well-known method for determining the optimal number of clusters.
As we have seen, the basic idea behind K-Means clustering is to define clusters such that the total intra-cluster variation (or total Within-Cluster Sum of Square (WCSS)) is minimized (see Eq. 2.2). The total WCSS measures the compactness of the clustering and we want it to be as small as possible.
The Elbow method looks at the total WCSS as a function of the number of clusters: one should choose a number of clusters so that adding another cluster does not improve much better the total WCSS.
The steps to follow are, then, the following:

1. Compute K-means clustering for different values of K, chosen between 2 and the minimum between 90 and the number of SRVs in the railway line we are considering (the value of 90 was chosen experimentally to avoid too large and useless computations).
2. For each K, calculate the total WCSS.
3. Plot the curve of WCSS according to the number of clusters K.
4. Choose the location of an elbow in the plot as an indicator of the appropriate number of clusters.

Unfortunately, we do not always have clearly clustered data, meaning that the elbow may not be explicit and sharp. In such cases, we should rely on other methods.

The second method we present here is the **Silhouette method**. Silhouette analysis [Rousseeuw 1987] can be used to study the separation distance between the resulting clusters. The silhouette value is between $+1$ and -1 and measures how similar a point is to its own cluster compared to other clusters. A high value is desirable and indicates that the point is placed in the correct cluster, whereas a negative value indicates that the point is wrongly classified. If many points have a negative silhouette score, it may indicate that we have created too many or too few clusters.
Assume the data have been clustered into K clusters. For a data point $i \in C_i$ (data point i in the cluster C_i), let

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \tag{2.4}$$

be the mean distance between i and all other data points in the same cluster, where $d(i,j)$ is the distance between data points i and j in the cluster C_i.
$a(i)$ is the measure of similarity of the point i to its own cluster (the smaller the value, the better the assignment). We then define the mean dissimilarity of point i to some cluster C_k as the mean of the distance from i to all points in C_k (where $C_k \neq C_i$). For each data point $i \in C_i$, we now define

$$b(i) = \min_{k \neq i} \frac{1}{|C_K|} \sum_{j \in C_k} d(i,j) \tag{2.5}$$

to be the smallest mean distance of i to all points in any other cluster, of which i is not a member. The cluster with this smallest mean dissimilarity is said to be the *neighboring cluster* of i because it is the next best fit cluster for point i. We now define a *silhouette coefficient* of one data point i

$$s(i) = \frac{b(i) - a(i)}{\max\{(a(i), b(i)\}}, \text{ if } |C_i| > 1 \tag{2.6}$$

and

$$s(i) = 0, \text{ if } |C_i| = 1 \tag{2.7}$$

which can also be written as:

$$s(i) = \begin{cases} 1 - a(i)/b(i) & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1 & \text{if } a(i) > b(i) \end{cases} \tag{2.8}$$

from which it is clear that $-1 \leq s(i) \leq 1$. Also, note that score is 0 for clusters with size equal one. This constraint is added to prevent the number of clusters from increasing significantly.

The silhouette method can be used as evaluation method for any clustering method, also for the DBSCAN algorithm that will be introduced in the next section 2.6.3, and we will rely on it in the final example of section 2.7 in order to choose the best clustering model.
Moreover, it can be applied specifically for K-Means to set the best number of clusters K. In particular, we will rely on the average silhouette score, obtained by a Python pre built function that simply averages all Silhouette values of all points. We follow this procedure:

1. Compute K-means clustering for different values of K, chosen, as in the elbow method, between 2 and the minimum between 90 and the number of SRVs in the railway line we are considering.

2. For each K, calculate the average silhouette score.

3. Plot the curve of WCSS according to the number of clusters K.

4. Choose the first local maximum in the plot as an indicator of the appropriate number of clusters.

The choice to take the first local maximum was done experimentally after having examined the trend of the average silhouette score for all railway lines with at least 50 SRVs (that's a possible threshold we chose for applying clustering methods).
From the plot in Figure 2.13 one can observe a common trend in which the function initially increase to a local maximum, then decreases and regrow again. We want to keep the first local maximum because too many clusters would result in a situation where almost each SRV forms its own cluster and this would be useless and unnecessarily expensive for our analysis.

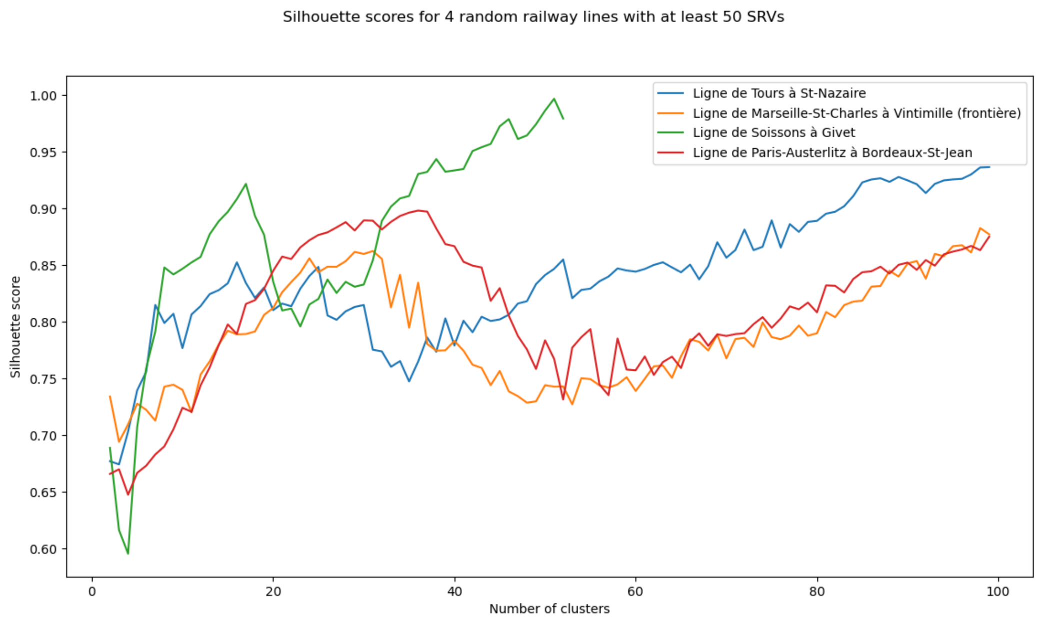


Figure 2.13: Average silhouette plot of 4 random railway lines with at least 50 SRVs

2.6.3 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) [Ester et al. 1996] is a density-based clustering non-parametric algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away).

Consider a set of points in some space to be clustered. Let ϵ be a parameter specifying the radius of a neighborhood with respect to some point, we define the ϵ-neighborhood of a point as:

$$N_\epsilon(x) = \{y \mid d(x, y) \leq \epsilon\}$$

For the purpose of DBSCAN clustering, the points are classified as *core points*, *reachable points* and *outliers*, as follows:

- A point x is a core point if the ϵ-neighborhood of x contains at least $minpts$, including x;
- A point x is a directly reachable point from y if x is within the ϵ-neighborhood of y and y is a core point;
- A point x is reachable from y if there is a chain of points $x_1, x_2, \ldots, x_p$ where $x_1 = x$, $x_p = y$ and such that x_{i+1} is directly density reachable from x_i;
- A point x is density-connected to y with respect to ϵ and $minpts$ if there is a point z such that both x and y are reachable from z;
- All points not reachable from any other point are outliers or noise points.

Now if x is a core point, then it forms a cluster together with all points (core or non-core) that are reachable from it. Each cluster contains at least one core point; non-core points can be part of a cluster, but they form its "edge", since they cannot be used to reach more points (see Algorithm 4).

Algorithm 4: DBSCAN Algorithm

```
input : $D$ - set of observations $x \in D$
        $\epsilon$ - neighborhood radious
        $minpts$ - minimum number of points to get a core point
output: $C$ - set of clusters
        $Core$ - set of core points
        $Border$ - set of not core points in neighborhood of core points
        $Noise$ - set of noise points
DBSCAN($D$,$\epsilon$,$minpts$):
    $Core = \emptyset$;
    foreach $x \in D$:
        Compute $N_\epsilon(x)$;
        $id(x) = -1$;                        /* Cluster id for x */
        if $|N_\epsilon(x)| \geq minpts$:
            $Core = Core \cup \{x\}$
    $k = 0$;
    foreach $x \in Core$ such that $id(x) = -1$:
        $id(x) = k$;                         /* Assign x to cluster id k */
        DENSITYCONNECTED($x$, $k$);
        $k = k + 1$;
    $C = \{C_i\}_{i=1}^{k}$, where $C_i = \{x \in D \mid id(x) = i\}$;
    $Noise = \{x \in D \mid id(x) = -1\}$;
    $Border = D \setminus \{Core \cup Noise\}$;
    return $C$, $Core$, $Border$, $Noise$;

DENSITYCONNECTED($x$, $k$):
    foreach $y \in N_\epsilon(x)$:
        $id(y) = k$;                         /* Assign y to cluster id k */
        if $y \in Core$:
            DENSITYCONNECTED($y$, $k$);
    return;
```

DBSCAN visits each point of the database, possibly multiple times, however, the time complexity is mostly governed by the number of point neighborhood computations. DBSCAN executes exactly one such computation for each point, and if an indexing structure is used that executes a neighborhood computation in $O(\log n)$, an overall average time complexity of $O(n \log n)$ is obtained; however the worst case run time complexity still remains $O(n^2)$. We can benefit from the distance matrix of size $O(n^2)$ to avoid distance recomputations, but this needs $O(n^2)$ memory, whereas a non-matrix based implementation of DBSCAN only needs $O(n)$ memory.

2.6.4 How to choose the *minpts* and ϵ parameters

The trickiest part of DBSCAN is the choice of his parameters *minpts* and ϵ, where the former is the fewest number of points required to form a cluster and the latter is the maximum distance two points can be from one another while still belonging to the same cluster.
There is not an automatic way to determine the *minpts* value for DBSCAN. Ultimately, this value should be set using domain knowledge and familiarity with the data set. However, there are some "rules of thumb" to set an appropriate value:

- The larger the dataset, the larger the value of *minpts* should be.
- The noisier the dataset, the larger the value of *minpts* should be.
- In general, *minpts* should be greater than or equal to the dimensionality of the dataset: for 2-dimensional data, use DBSCAN's default value of $minpts = 4$. For higher dimensions, choose $minpts = 2 * dim$, where *dim* is the dimension of the dataset [Sander et al. 1998].

As far as the ϵ choice is concerned, an automatic way to determine the optimal value is the **elbow method** [Rahmah and Sitanggang 2016].

Algorithm 5: How to find the optimal epsilon in DBSCAN algorithm (Elbow method)

```
input  : $D$ - set of observations $x \in D$
         $minpts$ - minimum number of points to get a core point
output: $\epsilon$ - neighborhood radius
1 Create an empty list of distances;
2 foreach $x \in D$:
3     Neighbs = NN(minpts);    /* Use Nearest Neighbours to find the
        closest minpts points to x */
4     Compute distance between x and the $minpts$-neighbour and append it
        to the list of distances;
5 Sort distances ascending and plot to find each value;
6 $\epsilon$ corresponds to critical change (elbow) in the curve;
7 return $\epsilon$;
```

This technique calculates the average distance between each point and its k nearest neighbors, where $k = minpts$. The average k-distances are then plotted in ascending order on a k-distance graph. The optimal value for ϵ is that one at the point of maximum curvature (i.e. where the graph has the greatest slope or elbow) (see Algorithm 5).

2.7 Practical Example

After having explained in details in the previous chapters the methods that we considered to perform the comparison analysis of SRVs between Data Prep and Data Prod, we want to apply them to a concrete example. In particular, we have chosen the railway line between Creil and Jeumont, two cities in the north of France (2.14).

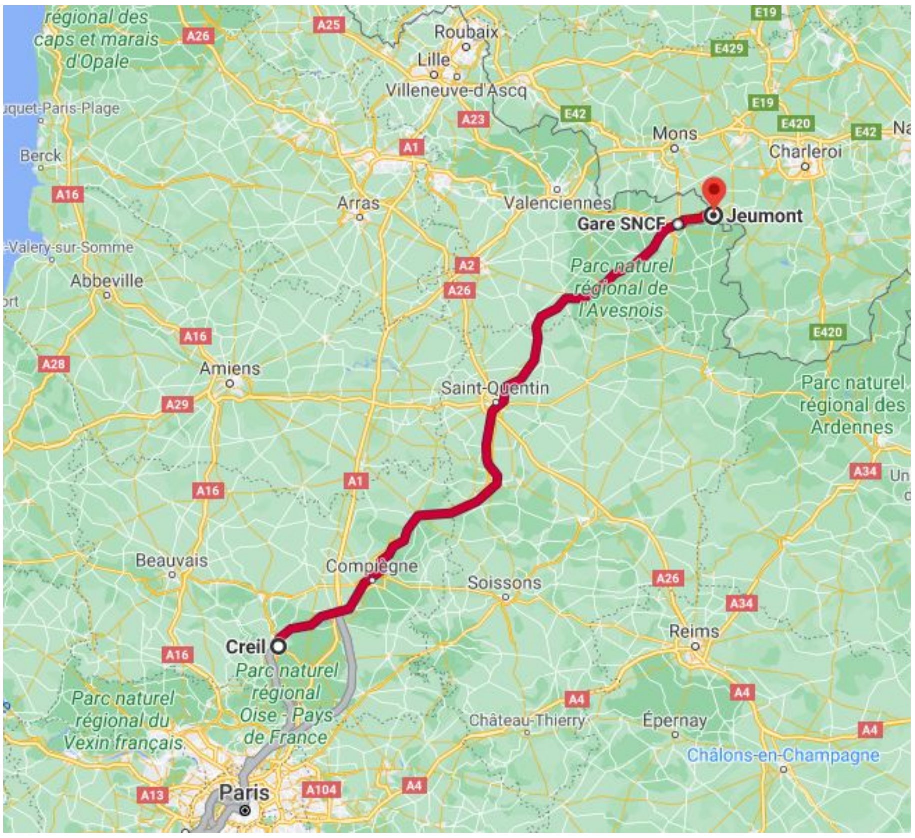


Figure 2.14: Railway line between Creil and Jeumont

In the comparison plot in Fig. 2.17 we can firstly notice the distribution of SRVs along the railway line. As usual pattern, we have 2 long SRVs that are the principal tracks, called "Voie V1 de Creil à Jeumont" and "Voie V2 de Creil à Jeumont", and many (in total 162 and 160 for Data Prep and Data Prod respectively) shorter SRVs that are distributed all along the railway lines.

As far as the SRV type **classification** part is concerned, we can simply use the deterministic algorithm 1 since we have all the information concerning the SRV names for this railway line.
However, we want to address the problem in the "unsupervised" settings, ignoring the real classes, putting us in a possible situation where we only have information on the lengths of the tracks and the number of sections (TIVs) that make them up. In order to be able to use the classification techniques analysed in chapter 2.5, we will train the classifiers on one of the main lines, for example the Marseille - Paris line, and test them on our example (see Fig. 2.16). In this way, we will also be able to verify the robustness of these algorithms, as they are trained on a smaller dataset.

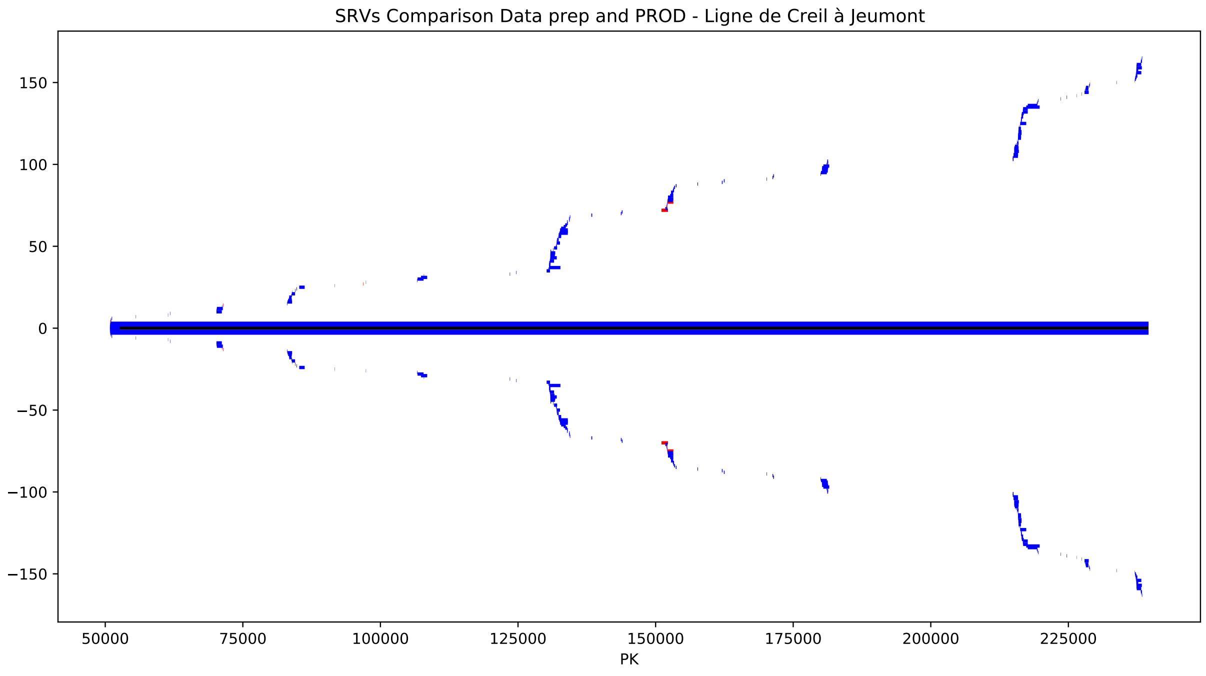


Figure 2.15: SRV Comparison between Data prep (above) and Data Prod (below) of the railway line Creil-Jeumont

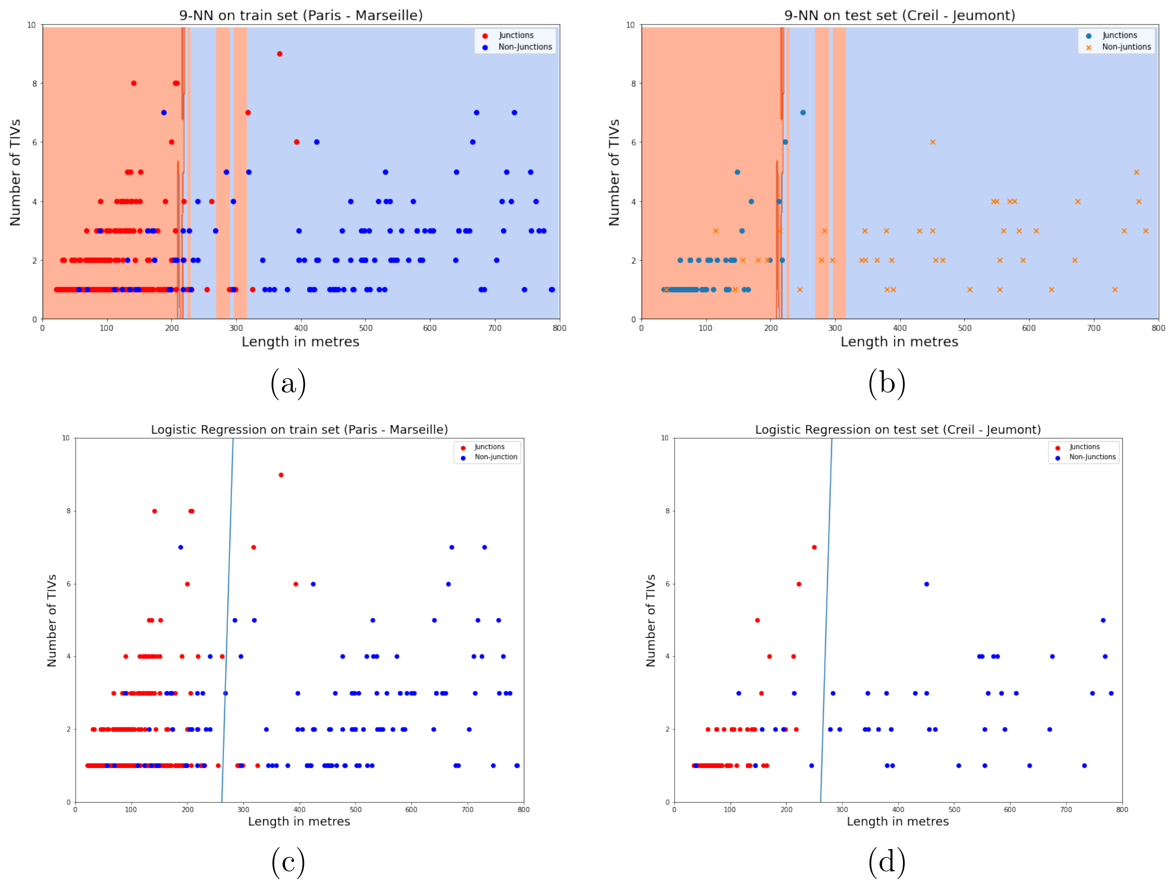
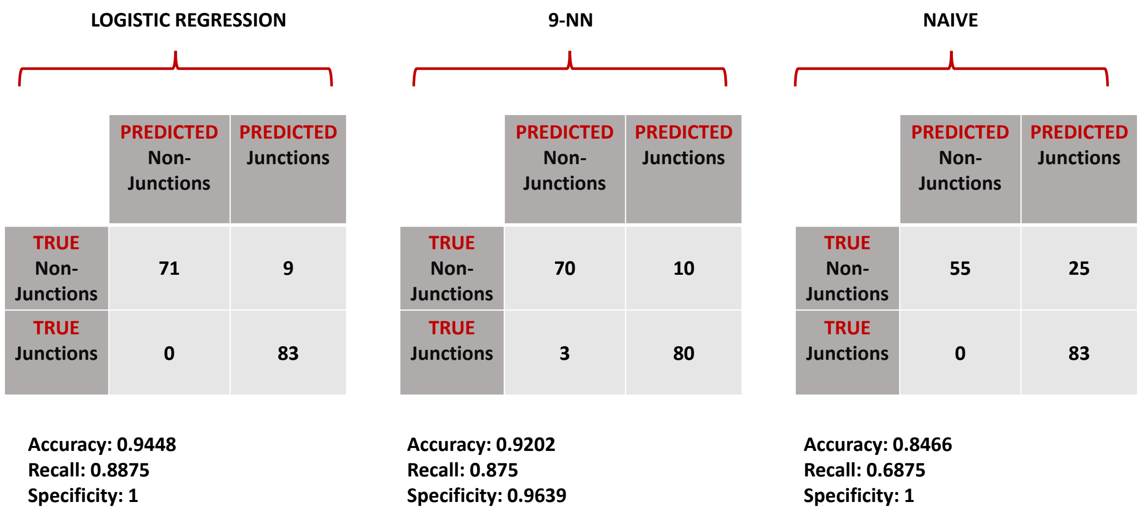


Figure 2.16: (a) 9-NN on train set (Paris-Marseille) (b) 9-NN on test set (Creil-Jeumont) (c) Logistic Regression on train set (Paris-Marseille) (d) Logistic Regressionon test set (Creil-Jeumont).

In order to better evaluate the performances of our model, we show the relevant confusion matrices together with the more relevant scores:



LOGISTIC REGRESSION

	PREDICTED Non-Junctions	PREDICTED Junctions
TRUE Non-Junctions	71	9
TRUE Junctions	0	83

Accuracy: 0.9448
Recall: 0.8875
Specificity: 1

9-NN

	PREDICTED Non-Junctions	PREDICTED Junctions
TRUE Non-Junctions	70	10
TRUE Junctions	3	80

Accuracy: 0.9202
Recall: 0.875
Specificity: 0.9639

NAIVE

	PREDICTED Non-Junctions	PREDICTED Junctions
TRUE Non-Junctions	55	25
TRUE Junctions	0	83

Accuracy: 0.8466
Recall: 0.6875
Specificity: 1

Figure 2.17: Confusion matrix for the three methods with the more relevant scores.

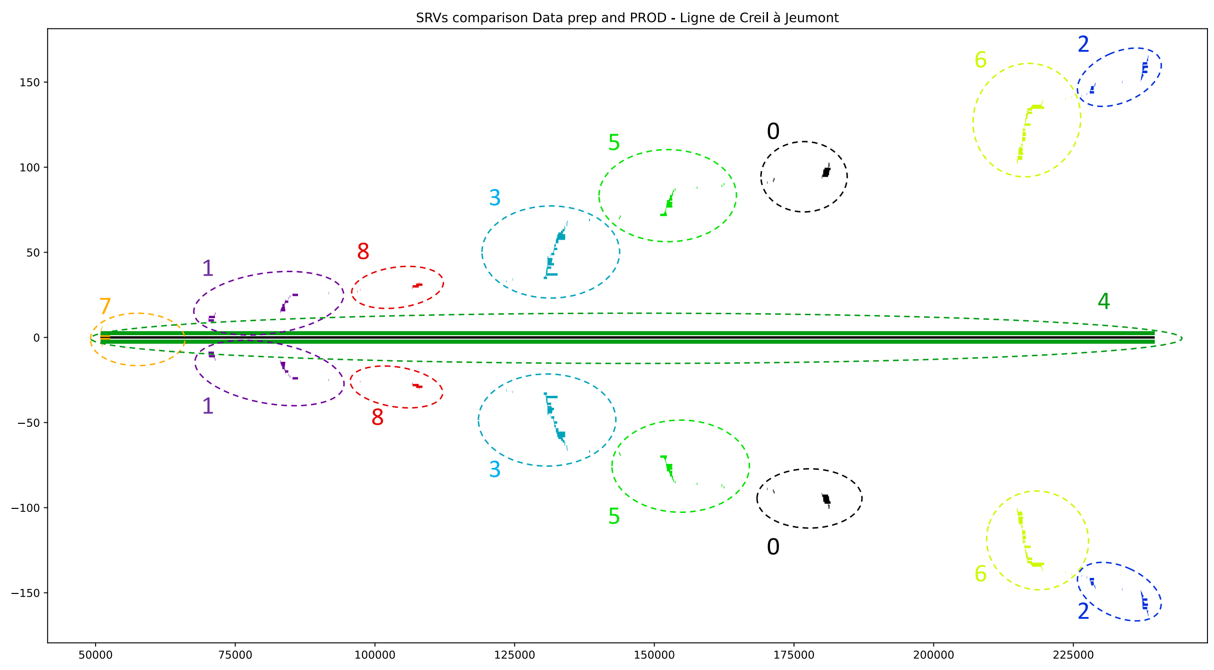
The accuracy of 9-NN and Logistic Regression are similar and definitely higher than Naive one. The recall (i.e. the number of corrected classified non-junctions over the total sum of them) is lower than accuracy for the three models and, in particular, for the naive model, which has some difficulty in classify the non-junctions. However, all three models manage to classify all junctions well, since the value of specificity is very high (i.e. the number of corrected classified junctions over the total sum of them). Note that apart from the Logistic Regression, the other methods have a slightly lower accuracy than the previously calculated with an empirical confidence interval, as the training phase is only done on 766 SRVs compared to the about 8000 SRVs as before. However, the results are still good, which confirms the robustness of the method even when training on only one railway line.

At this point, for the reasons already explained in chapters 2.3 and 2.6, we would like perform a **clustering analysis** as the data distribution seems to be suitable for this type of analysis.
We compare the 3 spacial clustering methods we have analyzed in the previous chapters:

- K-Means with K set to the elbow value of the WCSS plot (the Within-Cluster Sum of Square or, sometimes, simply SSE, the Sum of Square Errors);
- K-Means with K set to the first highest average silhouette value;
- DBSCAN with $minpts$ set to 4 and ϵ set to the elbow value of the $minpts$-neighbour distances of all points (see algorithm 5).

In figures 2.18, 2.19 and 2.20 we have displayed a pair of plots for each of the 3 clustering methods: on the top a SRVs comparison plot between Data Prep and Data Prod highlighting with different colors and circles the clusters obtained with the relative method; on the bottom a silhouette plot for the different clusters, where

K-Means with SSE method



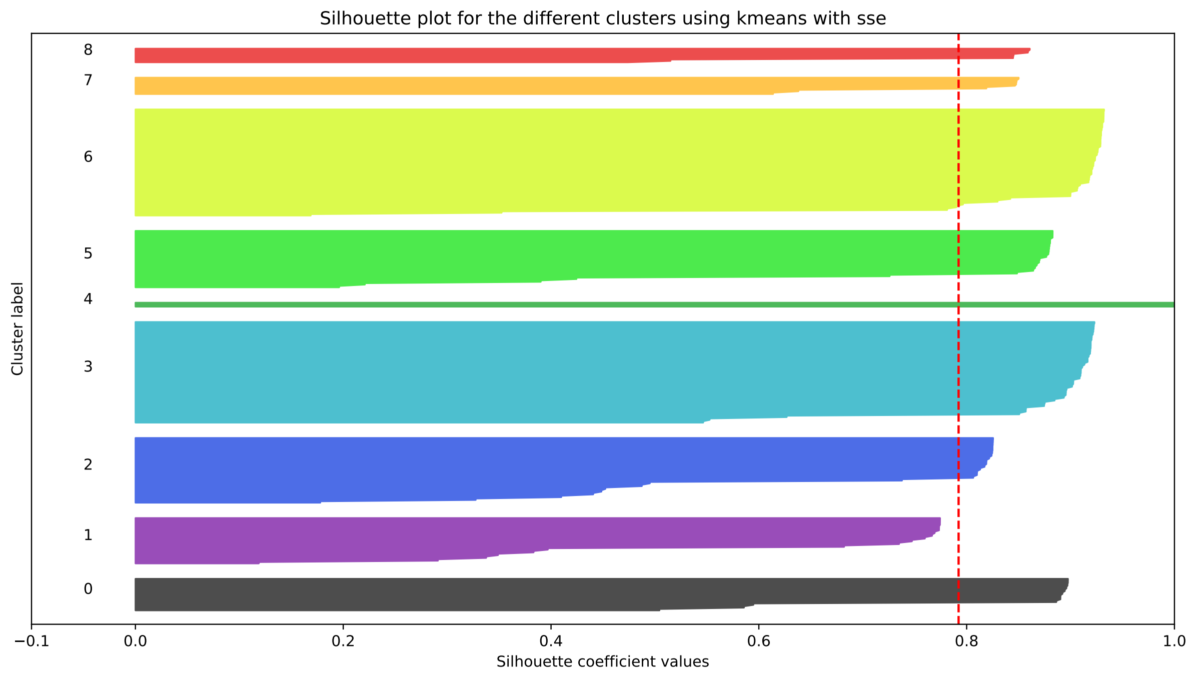
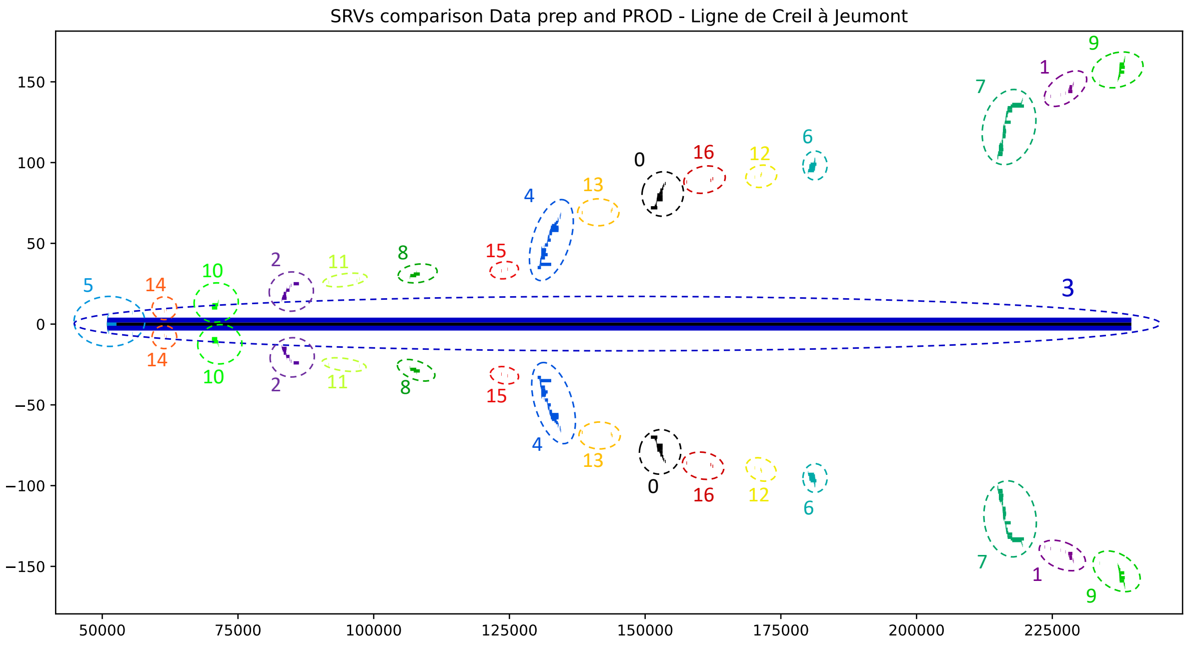


Figure 2.18: Silhouette Plot for K-Means with SSE method.

K-Means with silhouette method



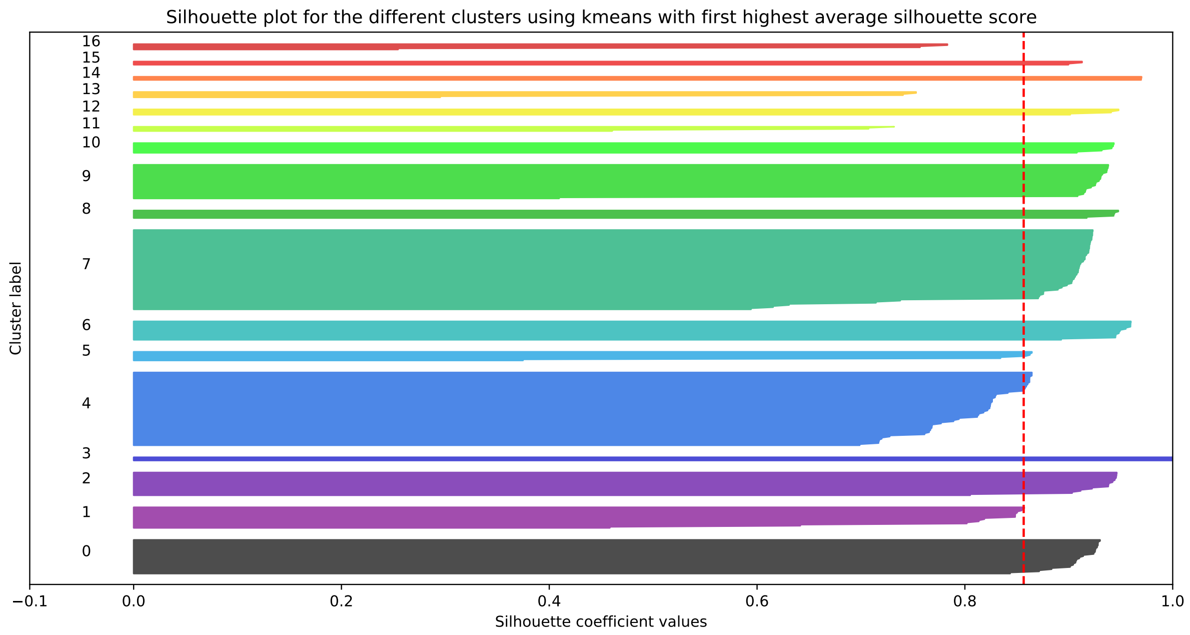
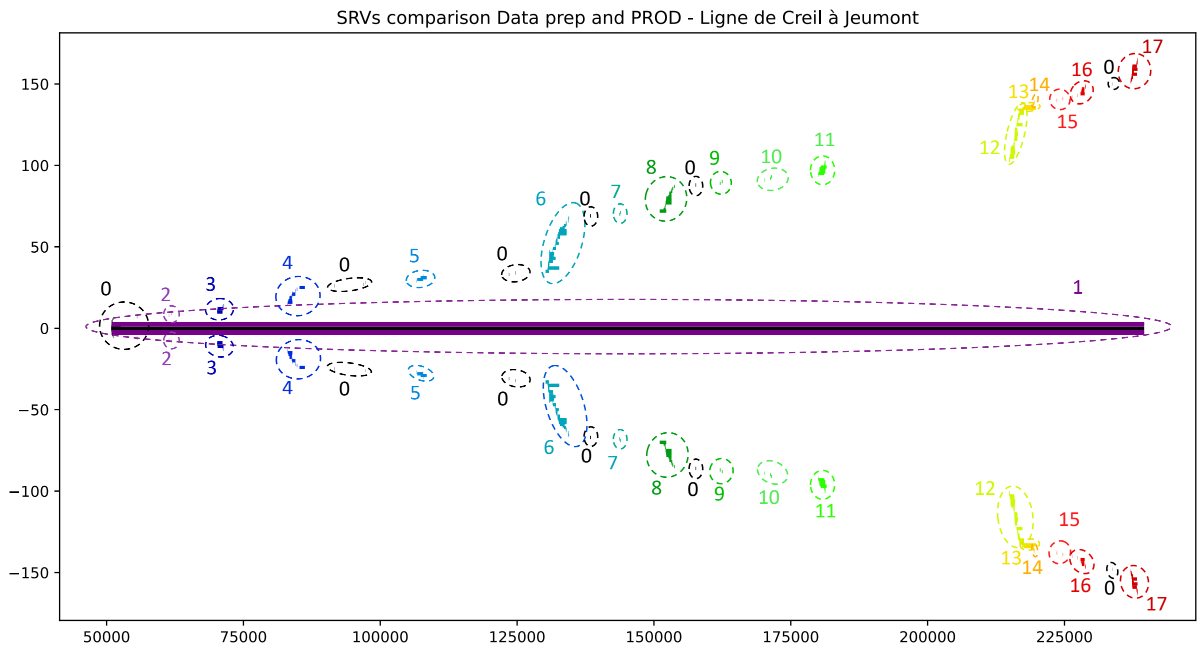


Figure 2.19: Silhouette Plot for K-Means with Silhouette method.

DBSCAN



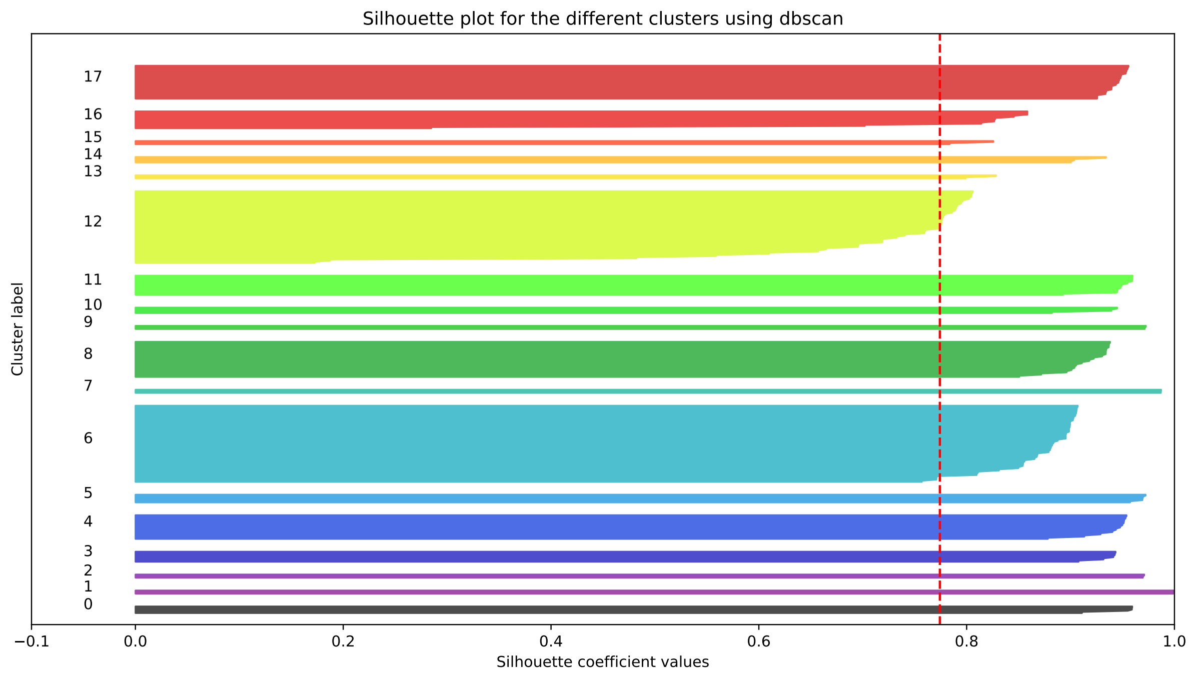
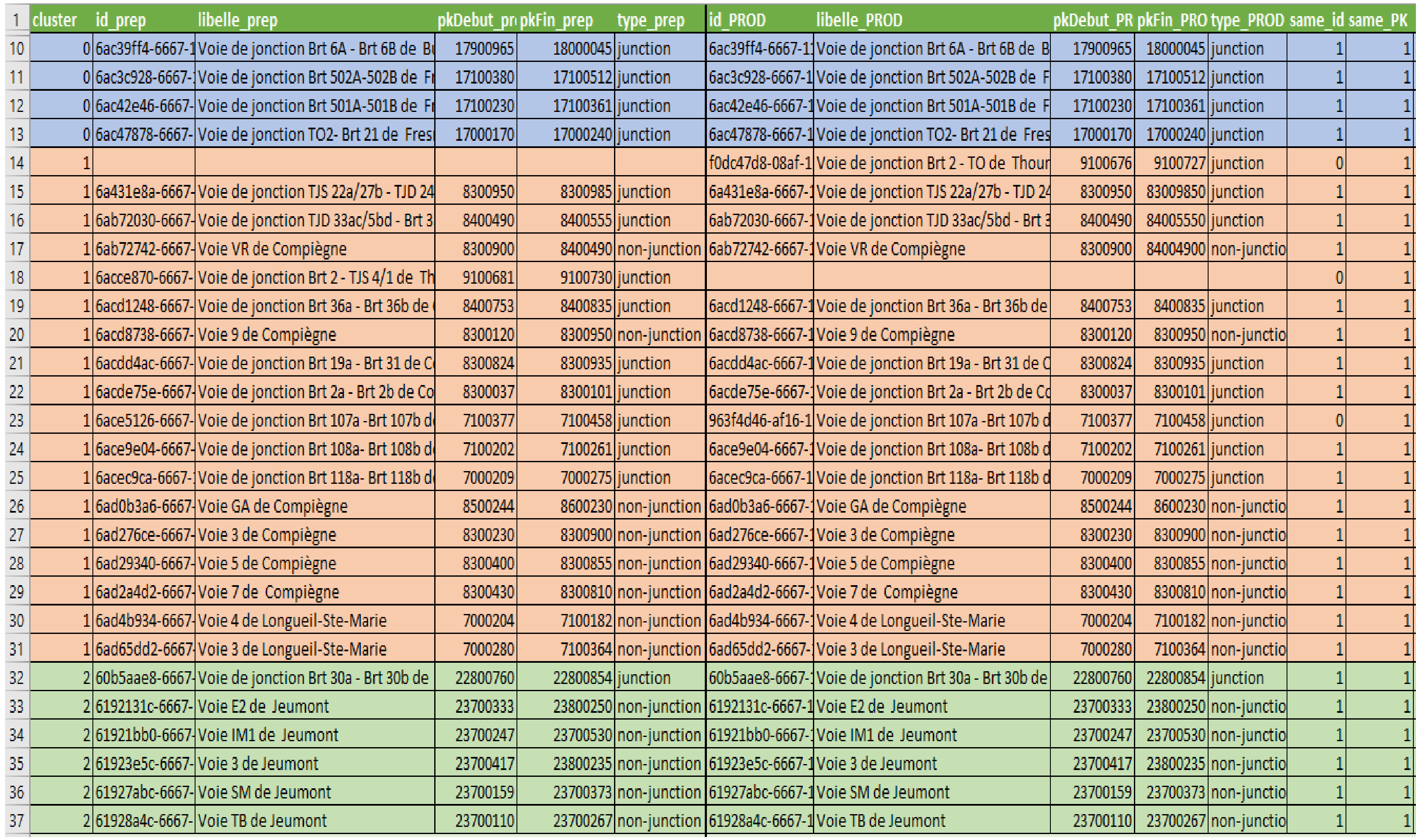


Figure 2.20: Silhouette Plot for DBSCAN.

for each of them we sorted in ascending order the silhouette value of all points belonging to it. In addition, we added a vertical red dashed line indicating the average silhouette score. Notice that the colors are shared between the two plots in order to make it easier to read them.

Firstly, notice that the K-Means with silhouette method and DBSCAN find 17 and 18 clusters respectively against the only 9 of the K-Means with SSE method. On one hand, having too many clusters can be a problem since we may miss some matches in the case we have 2 SRVs that change ID and they are assigned to 2 different groups. On the other hand, a smaller number of clusters results in a larger number of comparisons which could be very expensive in the case of railway lines with many SRVs. However, it is also true that this cost would be largely covered by the computational savings of calculating fewer clusters (e.g., K-Means complexity is linear in K). Also, notice that all methods recognise the main SRVs that follow the entire railway line as a separate cluster.
Another remark is that K-Means with silhouette method (Fig. 2.19) has the highest average silhouette score (also due to the way it is built) but the presence of several clusters (11,13,16) that are below the average silhouette score and the wide fluctuations in the size of the silhouette plots make it a bad pick for the given data. We have large fluctuations in clusters size for the DBSCAN as well (Fig. 2.20), but all clusters manage to exceed the average silhouette score. As far as the K-Means with SSE is concerned, we have just a problem with the cluster 1 that is not really well-formed since all silhouette scores are low, but the number of clusters and their sizes make it probably the most suitable for our example.
As last remark, observe that in DBSCAN we have the cluster 0 that is splitted in more subgroups. This is the "noisy cluster" built by this method: it is formed by all the small and isolated SRVs. For our objective it is not a particularly useful cluster as we can hardly associate SRVs that are distant from each other and even may be dangerous.

Finally, after the clustering analysis, we compare SRVs of the two datasets proceeding cluster by cluster. Firstly, we consider only SRVs that share the same ID and we compare the PK (Kilometric Point) to understand if there has been a shortening or elongation of the track. Then, we take SRVs with different ID and we check if there are any names in common. In that case, we will associate such SRVs even if with different ID, otherwise we will have a "new" SRV if this was not present in Data Prep, or an "elimination" if this is no longer present in Data Prod.
To better understand this final procedure, consider Fig. 2.21, where we have a part of the final comparison table. We have highlighted clusters in different colours to facilitate reading. When a box is empty it means that the SRV has been deleted or created. For example, on the 14th row we have a SRV that is new in Data Prod, while on the 18th row the SRV was in the Data Prep but is no longer present on Data Prod.
Note that we added two columns at the end to indicate whether the SRVs share the same ID and/or PKs between the two datasets (0 if they not share, 1 if yes). Obviously, for the latter two examples we have talked about, the ID is different as the SRV is present in only one of the two datasets, but there is also a case (row 23) where the SRV has a different ID but the same name between the datasets.



1	cluster	id_prep	libelle_prep	pkDebut_pr	pkFin_prep	type_prep	id_PROD	libelle_PROD	pkDebut_PR	pkFin_PRO	type_PROD	same_id	same_PK
10	0	6ac39ff4-6667-1	Voie de jonction Brt 6A - Brt 6B de Bı	17900965	18000045	junction	6ac39ff4-6667-1	Voie de jonction Brt 6A - Brt 6B de B	17900965	18000045	junction	1	1
11	0	6ac3c928-6667-	Voie de jonction Brt 502A-502B de Fı	17100380	17100512	junction	6ac3c928-6667-1	Voie de jonction Brt 502A-502B de F	17100380	17100512	junction	1	1
12	0	6ac42e46-6667-	Voie de jonction Brt 501A-501B de Fı	17100230	17100361	junction	6ac42e46-6667-1	Voie de jonction Brt 501A-501B de F	17100230	17100361	junction	1	1
13	0	6ac47878-6667-	Voie de jonction TO2- Brt 21 de Fres	17000170	17000240	junction	6ac47878-6667-1	Voie de jonction TO2- Brt 21 de Fres	17000170	17000240	junction	1	1
14	1						f0dc47d8-08af-1	Voie de jonction Brt 2 - TO de Thour	9100676	9100727	junction	0	1
15	1	6a431e8a-6667-	Voie de jonction TJS 22a/27b - TJD 24	8300950	8300985	junction	6a431e8a-6667-1	Voie de jonction TJS 22a/27b - TJD 24	8300950	83009850	junction	1	1
16	1	6ab72030-6667-	Voie de jonction TJD 33ac/5bd - Brt 3	8400490	8400555	junction	6ab72030-6667-1	Voie de jonction TJD 33ac/5bd - Brt 3	8400490	84005550	junction	1	1
17	1	6ab72742-6667-	Voie VR de Compiègne	8300900	8400490	non-junction	6ab72742-6667-1	Voie VR de Compiègne	8300900	84004900	non-junctio	1	1
18	1	6acce870-6667-	Voie de jonction Brt 2 - TJS 4/1 de Th	9100681	9100730	junction						0	1
19	1	6acd1248-6667-	Voie de jonction Brt 36a - Brt 36b de	8400753	8400835	junction	6acd1248-6667-1	Voie de jonction Brt 36a - Brt 36b de	8400753	8400835	junction	1	1
20	1	6acd8738-6667-	Voie 9 de Compiègne	8300120	8300950	non-junction	6acd8738-6667-1	Voie 9 de Compiègne	8300120	8300950	non-junctio	1	1
21	1	6acdd4ac-6667-	Voie de jonction Brt 19a - Brt 31 de C	8300824	8300935	junction	6acdd4ac-6667-1	Voie de jonction Brt 19a - Brt 31 de C	8300824	8300935	junction	1	1
22	1	6acde75e-6667-	Voie de jonction Brt 2a - Brt 2b de Co	8300037	8300101	junction	6acde75e-6667-	Voie de jonction Brt 2a - Brt 2b de Co	8300037	8300101	junction	1	1
23	1	6ace5126-6667-	Voie de jonction Brt 107a -Brt 107b d	7100377	7100458	junction	963f4d46-af16-1	Voie de jonction Brt 107a -Brt 107b d	7100377	7100458	junction	0	1
24	1	6ace9e04-6667-	Voie de jonction Brt 108a- Brt 108b d	7100202	7100261	junction	6ace9e04-6667-1	Voie de jonction Brt 108a- Brt 108b d	7100202	7100261	junction	1	1
25	1	6acec9ca-6667-	Voie de jonction Brt 118a- Brt 118b d	7000209	7000275	junction	6acec9ca-6667-1	Voie de jonction Brt 118a- Brt 118b d	7000209	7000275	junction	1	1
26	1	6ad0b3a6-6667-	Voie GA de Compiègne	8500244	8600230	non-junction	6ad0b3a6-6667-	Voie GA de Compiègne	8500244	8600230	non-junctio	1	1
27	1	6ad276ce-6667-	Voie 3 de Compiègne	8300230	8300900	non-junction	6ad276ce-6667-	Voie 3 de Compiègne	8300230	8300900	non-junctio	1	1
28	1	6ad29340-6667-	Voie 5 de Compiègne	8300400	8300855	non-junction	6ad29340-6667-	Voie 5 de Compiègne	8300400	8300855	non-junctio	1	1
29	1	6ad2a4d2-6667-	Voie 7 de Compiègne	8300430	8300810	non-junction	6ad2a4d2-6667-	Voie 7 de Compiègne	8300430	8300810	non-junctio	1	1
30	1	6ad4b934-6667-	Voie 4 de Longueil-Ste-Marie	7000204	7100182	non-junction	6ad4b934-6667-	Voie 4 de Longueil-Ste-Marie	7000204	7100182	non-junctio	1	1
31	1	6ad65dd2-6667	Voie 3 de Longueil-Ste-Marie	7000280	7100364	non-junction	6ad65dd2-6667-	Voie 3 de Longueil-Ste-Marie	7000280	7100364	non-junctio	1	1
32	2	60b5aae8-6667-	Voie de jonction Brt 30a - Brt 30b de	22800760	22800854	junction	60b5aae8-6667-	Voie de jonction Brt 30a - Brt 30b de	22800760	22800854	junction	1	1
33	2	6192131c-6667-	Voie E2 de Jeumont	23700333	23800250	non-junction	6192131c-6667-1	Voie E2 de Jeumont	23700333	23800250	non-junctio	1	1
34	2	61921bb0-6667-	Voie IM1 de Jeumont	23700247	23700530	non-junction	61921bb0-6667-	Voie IM1 de Jeumont	23700247	23700530	non-junctio	1	1
35	2	61923e5c-6667-	Voie 3 de Jeumont	23700417	23800235	non-junction	61923e5c-6667-1	Voie 3 de Jeumont	23700417	23800235	non-junctio	1	1
36	2	61927abc-6667-	Voie SM de Jeumont	23700159	23700373	non-junction	61927abc-6667-1	Voie SM de Jeumont	23700159	23700373	non-junctio	1	1
37	2	61928a4c-6667-	Voie TB de Jeumont	23700110	23700267	non-junction	61928a4c-6667-1	Voie TB de Jeumont	23700110	23700267	non-junctio	1	1

Figure 2.21: Part of the final table comparing SRVs of Data Prep and Data Prod of the railway line from Creil to Jeumont.

2.8 TIVs Comparison

In the previous chapters we have discussed several times about TIVs ("Tronçon d'Itinéraire Voie", *Itinerary Section Track*) using them as a variable, together with the length of the track, to discriminate between junctions and non-junctions, since on average, as we have seen, junctions have fewer segments than the others.
In this last section of this chapter we want to perform a short comparison analysis of TIVs between Data Prep and Data Prod, similarly to what we did with the SRVs. However, unlike SRVs, the work here is facilitated by the fact that we have topological continuity of the TIVs. Moreover, a TIV is simply represented by:

- **Id**: long and unique alphanumeric value identifying the TIV.
- **PkDebut**: starting PK.
- **PkFin**: final PK.
- **SrvId**: ID of the SRV to which it belongs.

Notice that they don't have a name, but they have just an unique ID that identifies them.
The idea is the following:

1. Collect the list of TIVs from a SRV that is in common between Data Prep and Data Prod.
2. Compare the list of IDs to understand which TIVs are the same and which are removed or created in Data Prod.
3. Comparing PKs to see which segments have been lengthened and which shortened.

Notice that, due to the kilometre continuity of PKs, if a TIV has been shortened/lengthened to the right, for example, the following one has received an elongation/shortening to the left.
In order to have a tool that helps us in visualizing the differences in PK and ID, together with an output comparison table, we created an interactive plot that shows all the TIVs as segments making up a SRV of Data Prep and Data Prod.
For a practical example, consider the SRV "Voie V2 de Creil à Jeumont", that is one of the 2 main tracks composing the railway line we analyzed in Chapter 2.7. In the Figure 2.22 we have two lines: in the upper side we have the sequence of TIVs, separated by black vertical lines, composing the Data Prep SRV; in the lower side, instead, we have the same for the SRV belonging to Data Prod. We colored in blue the TIVs that share the same ID, in red when these are different.
When we have a track formed by several segments, as in this case, it's not easy to visually understand the differences of PK. For this reason, a red dot has been inserted wherever we have a difference of PK. For instance, consider the section starting at PK 91681 in Data Prep and at PK 91676 in Data Prod: firstly, it is colored in red because his ID has been changed; secondly, there is a red point close to the starting border indicating the adjustment of PK undergone. In order to have detailed information on the PK of any border, we have added an event to the graph which opens a small window containing the value of PK by moving the mouse over it.

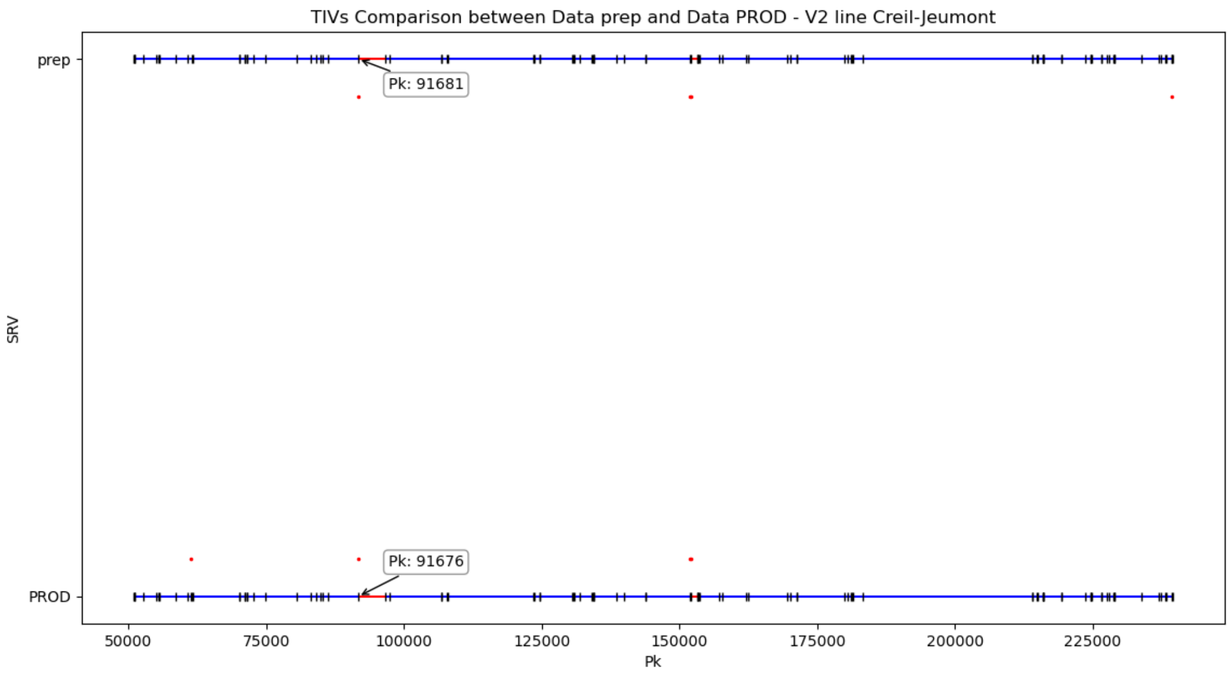
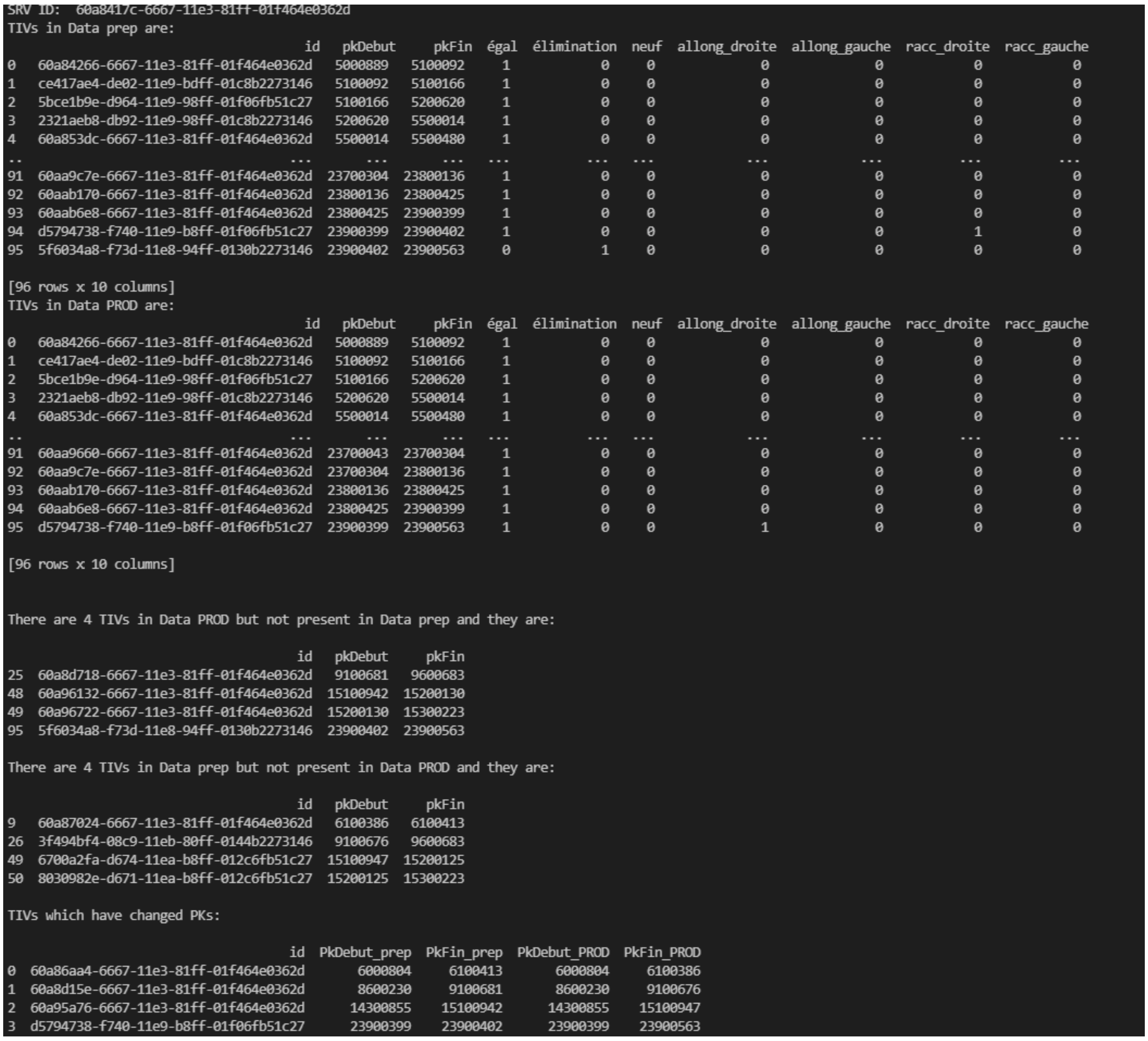


Figure 2.22: Comparison plot between Data Prep and Data Prod for TIVs composing the SRV "Voie V2 de Creil à Jeumont".

Finally, the output of our Python function gives all specific information about the comparison we are looking for (Fig. 2.23).
The first two blocks displays the list of TIVs for both datasets (we have just plotted the heads and the tails for size issues) with a resume of their situations. It's a table of 0-1 values with these attributes:

- **égal** (*equal*): if the TIV ID is the same between the datasets.
- **élimination** (*elimination*): if the TIV ID has been removed from Data Prep.
- **neuf** (*new*): if the TIV ID has been created in Data Prod.
- **allong_droite** (*extension to the right*): if the TIV has been stretched to the right with respect to its counterpart in the other dataset.
- **allong_gauche** (*extension to the left*): if the TIV has been stretched to the left with respect to its counterpart in the other dataset.
- **racc_droite** (*shortening to the right*): if the TIV has been shortened to the right with respect to its counterpart in the other dataset.
- **racc_gauche** (*shortening to the left*): if the TIV has been shortened to the left with respect to its counterpart in the other dataset.

Notice, for example, that the TIV at position 95 has been removed from Data Prep and, for this reason, the TIV at position 94 has had a shortening with respect to the same TIV in Data Prod, that inevitably has been extended at right.
Then, we have two blocks with TIVs that are unique in Data Prod and Data Prep respectively. Finally, the last block shows the list of TIVs which have changed PKs.



```
SRV ID:  60a8417c-6667-11e3-81ff-01f464e0362d
TIVs in Data prep are:
                                      id   pkDebut     pkFin  égal  élimination  neuf  allong_droite  allong_gauche  racc_droite  racc_gauche
0   60a84266-6667-11e3-81ff-01f464e0362d   5000889   5100092     1            0     0              0              0            0            0
1   ce417ae4-de02-11e9-bdff-01c8b2273146   5100092   5100166     1            0     0              0              0            0            0
2   5bce1b9e-d964-11e9-98ff-01f06fb51c27   5100166   5200620     1            0     0              0              0            0            0
3   2321aeb8-db92-11e9-98ff-01c8b2273146   5200620   5500014     1            0     0              0              0            0            0
4   60a853dc-6667-11e3-81ff-01f464e0362d   5500014   5500480     1            0     0              0              0            0            0
..                                   ...       ...       ...   ...          ...   ...            ...            ...          ...          ...
91  60aa9c7e-6667-11e3-81ff-01f464e0362d  23700304  23800136     1            0     0              0              0            0            0
92  60aab170-6667-11e3-81ff-01f464e0362d  23800136  23800425     1            0     0              0              0            0            0
93  60aab6e8-6667-11e3-81ff-01f464e0362d  23800425  23900399     1            0     0              0              0            0            0
94  d5794738-f740-11e9-b8ff-01f06fb51c27  23900399  23900402     1            0     0              0              0            1            0
95  5f6034a8-f73d-11e8-94ff-0130b2273146  23900402  23900563     0            1     0              0              0            0            0

[96 rows x 10 columns]
TIVs in Data PROD are:
                                      id   pkDebut     pkFin  égal  élimination  neuf  allong_droite  allong_gauche  racc_droite  racc_gauche
0   60a84266-6667-11e3-81ff-01f464e0362d   5000889   5100092     1            0     0              0              0            0            0
1   ce417ae4-de02-11e9-bdff-01c8b2273146   5100092   5100166     1            0     0              0              0            0            0
2   5bce1b9e-d964-11e9-98ff-01f06fb51c27   5100166   5200620     1            0     0              0              0            0            0
3   2321aeb8-db92-11e9-98ff-01c8b2273146   5200620   5500014     1            0     0              0              0            0            0
4   60a853dc-6667-11e3-81ff-01f464e0362d   5500014   5500480     1            0     0              0              0            0            0
..                                   ...       ...       ...   ...          ...   ...            ...            ...          ...          ...
91  60aa9660-6667-11e3-81ff-01f464e0362d  23700043  23700304     1            0     0              0              0            0            0
92  60aa9c7e-6667-11e3-81ff-01f464e0362d  23700304  23800136     1            0     0              0              0            0            0
93  60aab170-6667-11e3-81ff-01f464e0362d  23800136  23800425     1            0     0              0              0            0            0
94  60aab6e8-6667-11e3-81ff-01f464e0362d  23800425  23900399     1            0     0              0              0            0            0
95  d5794738-f740-11e9-b8ff-01f06fb51c27  23900399  23900563     1            0     0              1              0            0            0

[96 rows x 10 columns]


There are 4 TIVs in Data PROD but not present in Data prep and they are:

                                      id   pkDebut     pkFin
25  60a8d718-6667-11e3-81ff-01f464e0362d   9100681   9600683
48  60a96132-6667-11e3-81ff-01f464e0362d  15100942  15200130
49  60a96722-6667-11e3-81ff-01f464e0362d  15200130  15300223
95  5f6034a8-f73d-11e8-94ff-0130b2273146  23900402  23900563

There are 4 TIVs in Data prep but not present in Data PROD and they are:

                                      id   pkDebut     pkFin
9   60a87024-6667-11e3-81ff-01f464e0362d   6100386   6100413
26  3f494bf4-08c9-11eb-80ff-0144b2273146   9100676   9600683
49  6700a2fa-d674-11ea-b8ff-012c6fb51c27  15100947  15200125
50  8030982e-d671-11ea-b8ff-012c6fb51c27  15200125  15300223

TIVs which have changed PKs:

                                     id  PkDebut_prep  PkFin_prep  PkDebut_PROD  PkFin_PROD
0  60a86aa4-6667-11e3-81ff-01f464e0362d       6000804     6100413       6000804     6100386
1  60a8d15e-6667-11e3-81ff-01f464e0362d       8600230     9100681       8600230     9100676
2  60a95a76-6667-11e3-81ff-01f464e0362d      14300855    15100942      14300855    15100947
3  d5794738-f740-11e9-b8ff-01f06fb51c27      23900399    23900402      23900399    23900563
```

Figure 2.23: Output of the TIVs comparison algorithm for the SRV "Voie V2 de Creil à Jeumont".

Chapter 3

Trackers and SRVs

In this chapter we will focus on a new problem that involves a new particular type of infrastructure element (*organ*): the "Localisateurs" (*trackers* or *locators*). These are sensors that are physically attached to the tracks and detect useful information related to the passage of trains.
Let's analyse the diagram in Figure 3.1, which explains how the trackers are integrated into the SNCF Réseau data collection system.

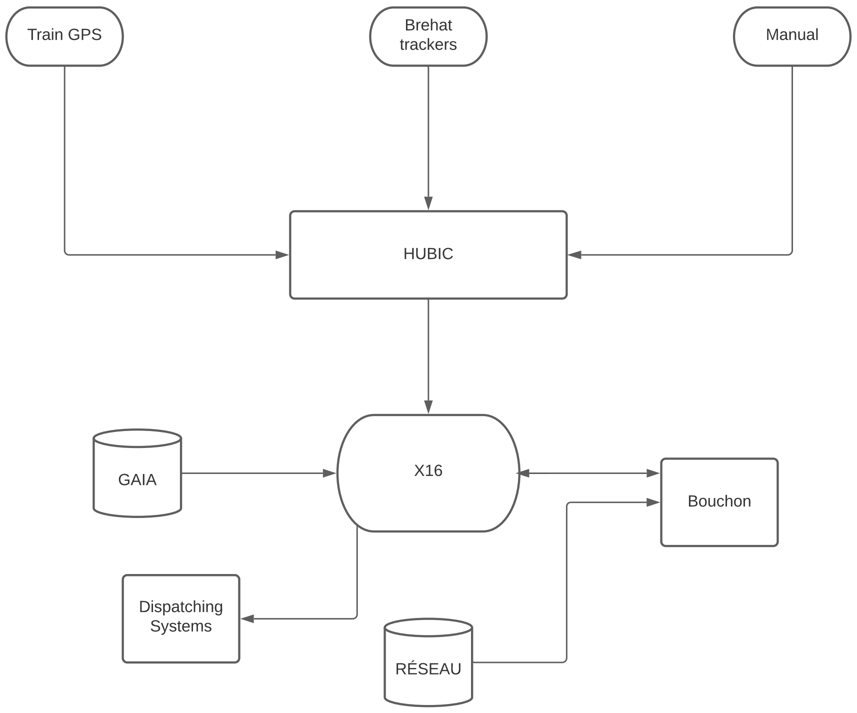


Figure 3.1: Diagram for the creation of file *bouchon*.

In the past, 3 sources of location data were used: 2 "historical" sources and one, more modern, based on the **GPS** signal of trains. The first of the two historical sources uses "**Brehat**" trackers attached directly to the railroad ways to provide information on the time of the train's passage, while the second source is **manual**, namely some agents detect the train's passage directly on the spot. This stream of data is then projected onto PRs (Points Remarquables, *Marker Points*) on the network where information on theoretical train passing times is available. In this way, it is possible to calculate the hourly variance and consequently the train delays. However, this stream of data is very noisy and inaccurate. For this reason, the **X16** project was created, which, through a model called **HUBIC**, aims to build a reference location base consistent with the bases identified by the railway companies that can cover all traffic in real time. This represents a major advantage, particularly in the event of disruptions.
X16 continuously uses this raw data stream together with the one from **GAÏA** to create a dynamic association between the tracks (SRV) and the locators. These data are then used for dispatching systems for communication. However, this dynamism in the data often leads to errors and missing information, which is the reason why we want now to create a stable and static system that allows each tracker, identified by a module and window number, to be associated uniquely with the SRV to which it is attached.
How to do this? There exists another data source, called **Réseau**, which provides association between trackers and railroad ways, detected with 3 particular systems that we will show in the next section (Section 3.1).
Then, the idea is to associate the data coming from *Réseau* to SRVs coming from *GAÏA* and create a table that we called **bouchon**.
Finally, in order to make more reliable the *bouchon* model and complete it, we will exploit the dynamic stream of data from *X16* (Section 3.2). In fact these data, although they may present anomalies (Section 3.2.1) or missing values (Section 3.2.2), provide a way to validate our reference file and make it more robust. In the same time, we will use the *bouchon* model to modify this dynamic stream of data so as to decrease the percentage of errors they have.

3.1 Tracker SRV Detection

As already mentioned, a tracker provides information on the passage of trains. In particular, we have:

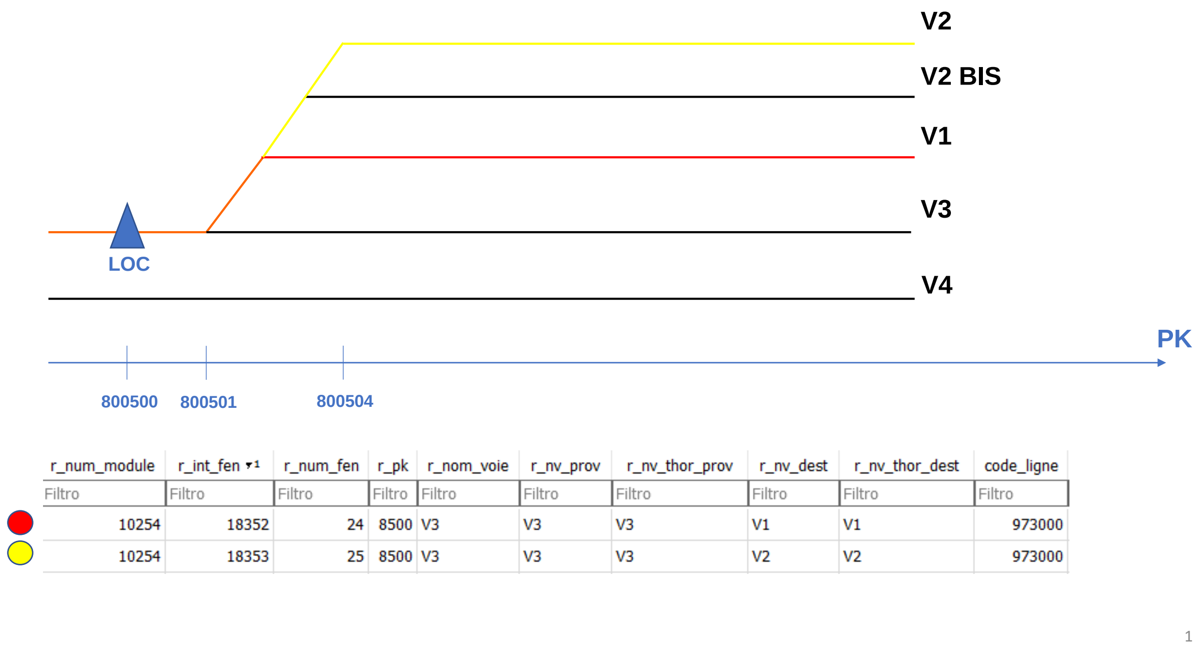
- **r_num_module**: tracker module number.
- **r_fen**: tracker window number.
- **r_int_fen**: an unique locator number for each pair module-window.
- **r_pk**: tracker PK (Kilometric Point).
- **r_nom_voie, r_nv_prov, r_nv_thor_prov**: 3 different systems for locating the origin track of the train. They are the tracks on which the locator is attached to.

- **r_nv_dest, r_nv_thor_dest**: 2 different systems for locating the destination track of the train.

- **id_srl**: unique ID of the railway line on which the tracker is located on.

- **libelle_ligne**: name of the railway line on which the tracker is located on.

Each tracker is identified by a module number, the **r_num_module**, that is not unique. To understand why, consider the Fig. 3.2.
There are several pathways parallel to each other. The tracker, identified by the



	r_num_module	r_int_fen ▾1	r_num_fen	r_pk	r_nom_voie	r_nv_prov	r_nv_thor_prov	r_nv_dest	r_nv_thor_dest	code_ligne
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
🔴	10254	18352	24	8500	V3	V3	V3	V1	V1	973000
🟡	10254	18353	25	8500	V3	V3	V3	V2	V2	973000

Figure 3.2: Example of a tracker with two different window numbers: in the first one, in red, the train starts from V3 and goes to V1. In the second one, in yellow, the train starts from V1 and goes to V2.

module number 10254 is located on V3 according to the three systems for the origin track. Recall that the origin track is also the railroad way on which the locator is installed. However, there are two possible destination tracks corresponding to two different window numbers: 24 and 25. The first module-window pair, identified with the **r_int_fen** number 18352 and colored in red in the figure, has V1 as destination track; whereas the second pair, identified with the **r_int_fen** number 18353 and colored in yellow, has V2 as destination track.

What is missing in our data is information about SRVs. In fact, we have 5 different systems for detecting the origin and destination tracks of the train for each specific module-window pair, but these are just generic abbreviations for the railroad way coming from an old source of data called "Réseau". Our task is to associate each of these "acronyms" to the corresponding SRV. In fact, the SRV is the unique

object that describes the railway track and provides us a lot of new useful information that we do not currently have with just the *Réseau* information.
In the next section we will focus on the origin track of the trains and we will implement an algorithm that creates the associations we are looking for, whereas in the following one we will almost adopt the same algorithm (with some modifications) for the destination track of the trains. In the following, we will consider only the Data Prod since, as we explained in the introduction, it is that one that is constantly updated and used for analysis.

3.1.1 Detection of Origin Track

In this part, we want to associate for each locator, and specifically for each pair of module-window number, the origin pathway of the train, expressed by the 3 variables **r_nom_voie**, **r_nv_prov** and **r_nv_thor_prov**, with the corresponding SRV.
The algorithm adopted can be summerized in the Alg.6.

Algorithm 6: Detection of the origin track

```
input  : Data_PROD - set of all SRV in Data Prod
         Line_PROD - set of all railway lines in Data Prod
         Localisateurs - set of all trackers
output:
         bouchon table - table of all trackers with the corresponding SRV
  for the origin track ("voie de provenance")
         not found tables - tables of all trackers without matching any SRV
  for the origin track
1 foreach line $\in$ Line_PROD:
2     Take a line;
3     Select all SRV (except for junctions) and all module-window tracker
        pairs on that line;
4     Select all SRVs such that: Initial PK SRV - X $\leq$ Tracker PK $\leq$ Final
        PK SRV + X, where X is a threshold (in metres);
5     Semantic comparison between the 3 abbreviations for the origin track
        and the SRV name;
6 Creation of the bouchon file and the two tables for matches not found;
7 return bouchon table and not found tables;
```

Firstly notice that since the origin railroad way is also the track on which the locator is attached to, we need to exploit the position of both the locator and SRV, expressed by their Kilometric Point (PK). Considering that PK is unique only over railway lines, we start by looping on the set of all lines and by extracting the list of all SRVs and trackers belonging to them. At this point, for each module-window tracker pair, we select those SRVs such that:

$$\text{Initial PK SRV} - \text{X} \leq \text{Tracker PK} \leq \text{Final PK SRV} + \text{X}$$

where X is a threshold in metres that we should take into account because sometimes the measurements are not really precise. In fact, it might happen that a tracker is

located just a little beyond the end of the track or a little before his starting PK. This value cannot be too small, otherwise we risk having false negatives, i.e. we miss some good matches, but neither can it be too large, otherwise we can get too many false positives, i.e. chosen fake matches. During our analysis we considered a threshold of 150 metres.
After this step we have in total more than 200000 possible matches out of 40667 unique trackers. In fact, as we can simply observe from Fig. 3.3, the position of a single tracker may belong to several SRVs.

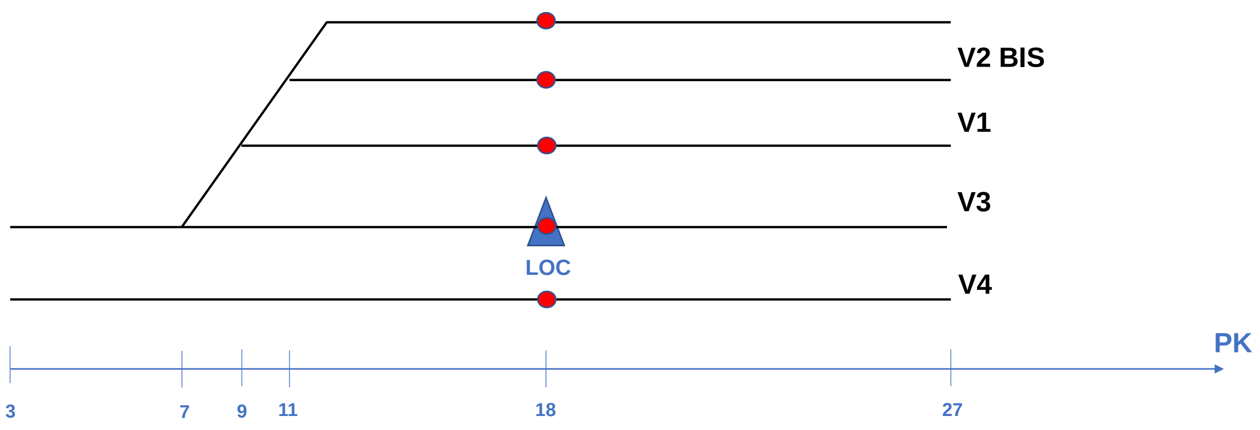


Figure 3.3: Example of a tracker, indicated with a blue triangle called "LOC" (*localisateur* in french, *tracker* in english) whose position is included between the initial and final PK of several SRVs.

In order to understand which is the real match among those available, we have to proceed with the **semantic comparison**.

After a first analysis of the structure of abbreviations and names of SRVs, we can observe that most of the tracks are called as "voie V* de **" (i.e., *track V* of ***), where * stands for a number and ** generally indicates the cities of origin and destination. The corresponding abbreviation, since it is shorter, is usually written as V* or just only *, where, as before, * stands for a number.
However, although this is the most frequent case, there are so many special structures and unique names that have made the creation of a generic algorithm very challenging. In fact, we can find tracks written in the most diverse manner: V1BIS, V2G, CIRC, SAS, V11-V12, etc.. Due to the hyper vertical structure of the problem, the application of a complex deep learning method would probably have been ineffective. However, some recurring patterns are present, and thanks to the use of regex (regular expressions), namely sequence of characters that specifies a search pattern, we have managed to build an algorithm (Alg.7) that succeeds in constructing most of the associations we are looking for.

1. Splitting in numbers and words.
The first step consists on dividing both the three abbreviations and the SRV name

```
Algorithm 7: Semantic Comparison
input  : $r\_nom\_voie, r\_nv\_prov, r\_nv\_thor\_prov$ - the 3 systems for
         locating the origin track
         $libelle\_srv$ - SRV name
output:
         bouchon table - set of all trackers with the corresponding SRV for
  the origin track ("voie de provenance")
         not found tables - tables of all trackers without matching any SRV.
1 Splitting in numbers and words;
2 Comparison between SRV and each of the three abbreviations;
3 Choice of the unique match;
4 return bouchon table and not found tables;
```

in words by splitting by spaces, brackets or other special punctuation marks. For the SRV name, we take everything after the words "voie" (that means *track*) and before "de","des","d'" (that means *of*).
Finally, we divided the list of strings in 2 big groups: **numbers** and **words**. Indeed, as we will see in the second step, the first comparison we are going to make is between numbers. If two strings do not share the same numbers, they are unlikely to refer to the same track and thus be considered a match. However, it is also true that even if the numbers in the two strings are the same, it may happen that there are any additional different words or letters and therefore the association is not valid. For instance, consider as SRV name "Voie V1G de Paris-Marseille" and as the found abbreviation "V1". Despite the fact that the number 1 appears in both notations, we cannot consider this a true match, since the SRV name contains the letter G, which could stand for "Garage" and thus refer to another track. Therefore, it is important to distinguish the numbers from the rest as first step in order to be able to make a more accurate comparison later.
Just a specific remark: when there is a string starting by V and longer than 1 character, we remove the V. The reason is that the letter V, the initial of "voie", may or may not be present, and to avoid complications and ambiguities we prefer to eliminate it.
Here some examples:

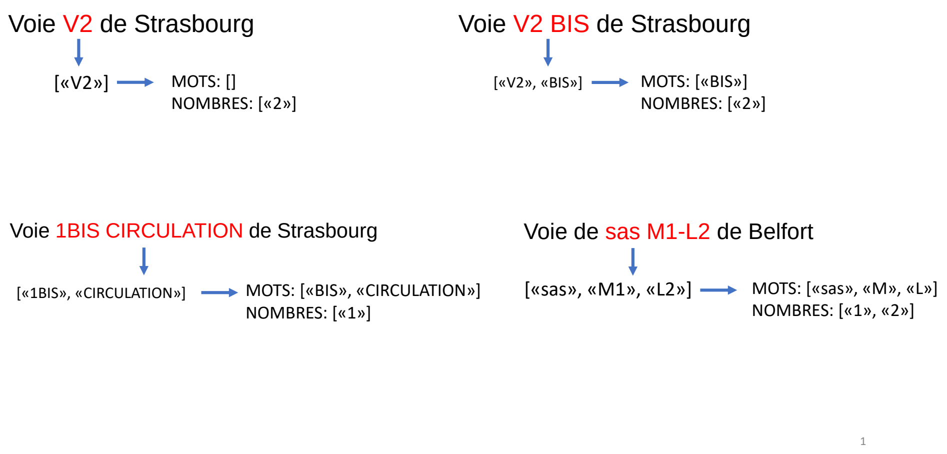


Figure 3.4: Examples of SRV name splitting in words and numbers

2. Comparison between SRV and each of the three abbreviations.
At this point, we are going into the heart of the semantic comparison. The algorithm simply consists of sequencing "if-else" conditions in descending order of probability of success so as to minimize the computational cost. In fact, we will start, as anticipated, from the comparison of the group of numbers and then we will gradually consider more and more specific and rare cases.
The comparisons can be listed as:

1. Comparison between the group of numbers. If true we continue the search, otherwise we stop and we consider it a false match.

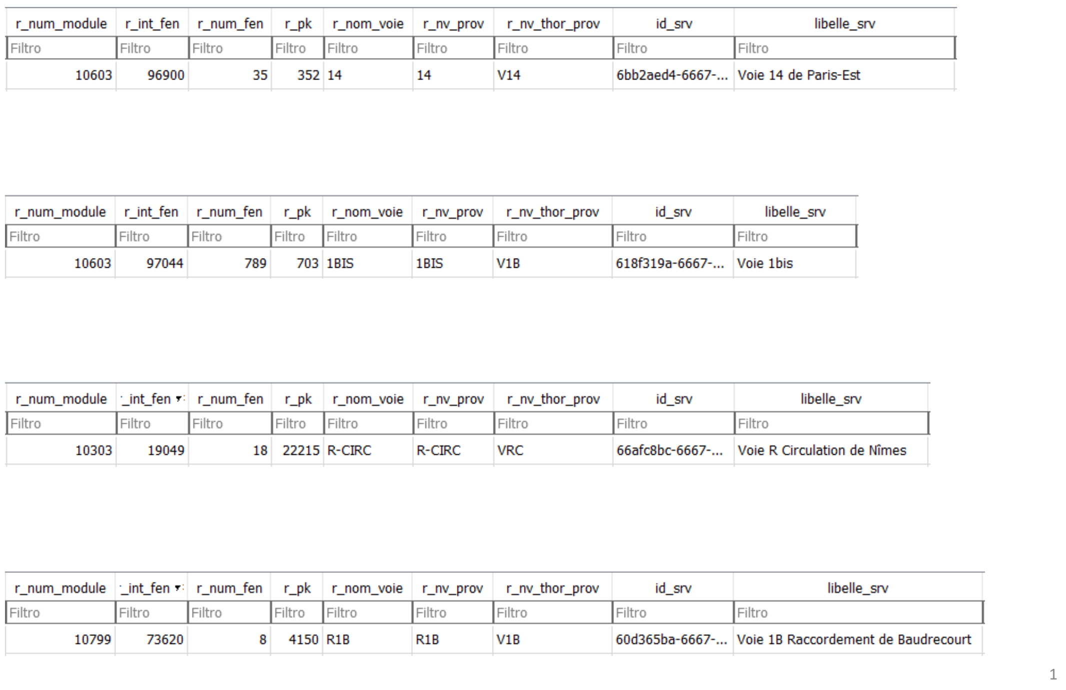
2. Check whether the group of words is exactly the same.

3. Check whether the words of the abbreviations are sub-sequences of some of the words of the SRVs.
   Ex: "CIRC" is a sub-sequence of "CIRCULATION".

4. Check whether the words of the abbreviations are a sequence formed by the initial letters of the SRV words.
   Ex: "V1BC" is a sequence formed by initial letters of "V1 BIS CIRCULATION".

5. The same as before, but considering permutations.
   Ex: "V1BC" is a sequence formed by initial permuted letters of "V1 CIRCULATION BIS".

Here some examples:



r_num_module	r_int_fen	r_num_fen	r_pk	r_nom_voie	r_nv_prov	r_nv_thor_prov	id_srv	libelle_srv
Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
10603	96900	35	352	14	14	V14	6bb2aed4-6667-...	Voie 14 de Paris-Est

r_num_module	r_int_fen	r_num_fen	r_pk	r_nom_voie	r_nv_prov	r_nv_thor_prov	id_srv	libelle_srv
Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
10603	97044	789	703	1BIS	1BIS	V1B	618f319a-6667-...	Voie 1bis

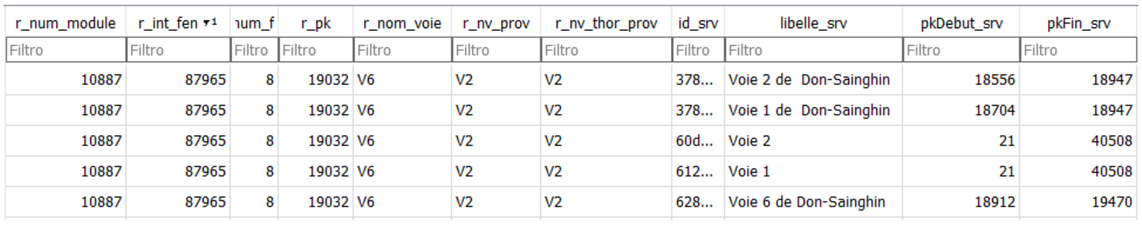
r_num_module	_int_fen	r_num_fen	r_pk	r_nom_voie	r_nv_prov	r_nv_thor_prov	id_srv	libelle_srv
Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
10303	19049	18	22215	R-CIRC	R-CIRC	VRC	66afc8bc-6667-...	Voie R Circulation de Nîmes

r_num_module	_int_fen	r_num_fen	r_pk	r_nom_voie	r_nv_prov	r_nv_thor_prov	id_srv	libelle_srv
Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
10799	73620	8	4150	R1B	R1B	V1B	60d365ba-6667-...	Voie 1B Raccordement de Baudrecourt

1

Figure 3.5: Examples of semantic analysis.

3. Choice of the unique match.
After the second step, we have collected in total 31812 out of 40667 module-window tracker pairs. However, some of these are multiples, i.e. a pair may have found 2 or more matches. Indeed, it usually happens that some track names are repeated along the railway line or that different systems for locating the origin track find different railroad ways. Which one should we choose?
Consider, as example, these 5 matches found by tracker with r_int_fen = 87965.



r_num_module	r_int_fen ▾1	ıum_f	r_pk	r_nom_voie	r_nv_prov	r_nv_thor_prov	id_srv	libelle_srv	pkDebut_srv	pkFin_srv
Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
10887	87965	8	19032	V6	V2	V2	378...	Voie 2 de Don-Sainghin	18556	18947
10887	87965	8	19032	V6	V2	V2	378...	Voie 1 de Don-Sainghin	18704	18947
10887	87965	8	19032	V6	V2	V2	60d...	Voie 2	21	40508
10887	87965	8	19032	V6	V2	V2	612...	Voie 1	21	40508
10887	87965	8	19032	V6	V2	V2	628...	Voie 6 de Don-Sainghin	18912	19470

Figure 3.6: Example of unique match choice.

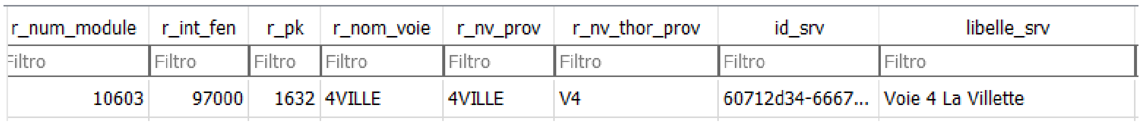
As there are 3 origin track identification systems, I select those rows for which I have the highest number of matches. In this particular example, since V2 appears two times out of three, I select the first and the third row.
After this first selection, if I have still ambiguity as in this case, we select the shortest track, so the first row in the example. The reason for this is that since there are railroad ways with repeated names, some of these are the main tracks (like "Voie 1" or "Voie 2" in our example) that run the entire length of the railway line and are, consequently, detected by all trackers on that line. Therefore, in case of ambiguity, the shortest railroad way is more likely to be the one actually identified.

At the end of this phase we found 28409 matches out of a total of 40667 (**70%**). We remark that we considered the row to be true whenever at least one of the three source pathway location systems finds a true match, as we gave the same importance to all the three systems.
The reasons why it is not possible to obtain 100% for the moment are as follows:

- 3355 trackers are without line code and this makes them useless for understanding which SRVs are attached to.
- Some trackers are located on secondary railroad ways (e.g. service) and not identified as SRV.
- There are so many special cases and creating an algorithm that takes them all into account would inevitably lead to many more false positives.

However, it is possible to increase this percentage in a second step (see more details Chapter 3.2).
In fact, as output of the algorithm, in addition to the creation of the *bouchon* model, we have two more tables of matches not found automatically: a larger table, created considering only the criterion of the same group of numbers, and a smaller one, created taking into account also the cases where all the words of the abbreviations were found in the SRV name, but this also contains something extra. It is clear then that the small table is a subset of the big one, where the probability of having a real

match is higher.
For instance, consider this not found match:



r_num_module	r_int_fen	r_pk	r_nom_voie	r_nv_prov	r_nv_thor_prov	id_srv	libelle_srv
Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
10603	97000	1632	4VILLE	4VILLE	V4	60712d34-6667...	Voie 4 La Villette

Figure 3.7: Example of not found match

The set of numbers, composed by only the number 4, is the same, as well as the word "VILLE" in the first two abbreviations, that stands for "Villette". However, in the SRV name appears also the word "La" (i.e. *the*) that is an extra word which prohibited the algorithm from considering it as a true match, but it is clear that this is a false negative.
Then, by creating a graphical interface that allows the user to select or remove the row from the table and through more precise tools it is possible to manually increase the number of total matches.

3.1.2 Detection of Destination Track

It is possible to carry out a similar analysis that we did for the origin track also for the destination track, using the variables **r_nv_dest** and **r_nv_thor_dest**. The algorithm we are going to use is approximately the same, but in this case the destination track is not anymore the pathway on which the locator is attached, but is a pathway that may be further or further back, depending on the direction of the train, than its position.
To better understand this concept, consider Fig. 3.8.

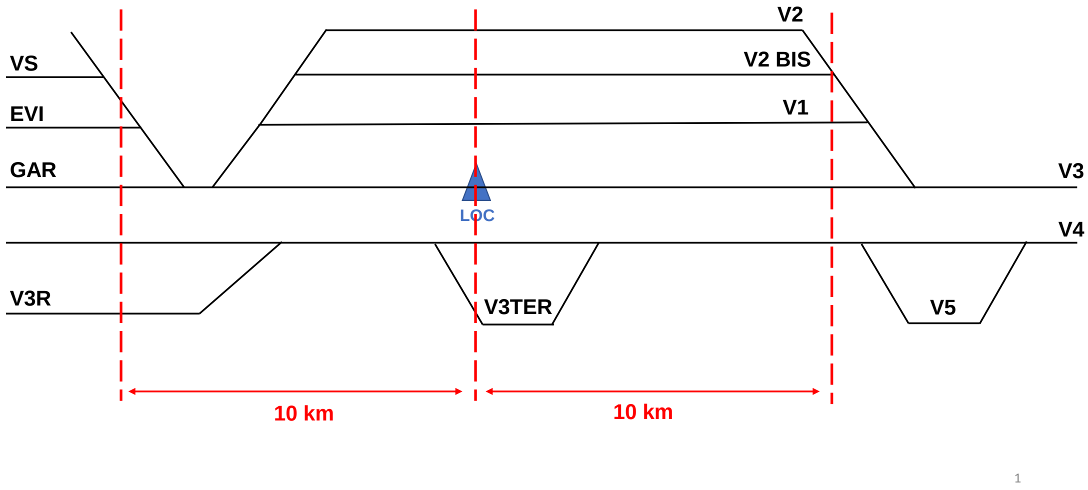


Figure 3.8: Example of a tracker attached to V3 in a system of railroad ways. We consider a spatial interval of 10 km before and after the tracker to trace the destination pathway.

We have a tracker that is located on V3 and there are several railroad ways around it. As a train can run from right to left or vice versa, we need to consider a spatial interval of a certain threshold around the position of the locator to find the destination track, that for our analysis we decided to set to 10 km. As before, this threshold has been set considering the fact that a too big value would lead more false positives, whereas with a too small value we can miss some good matches (false negatives). The general algorithm can be summarised in Alg. 8.

```
Algorithm 8: Detection of the destination track
input  : $Data\_PROD$ - set of all SRV in Data Prod
         $Line\_PROD$ - set of all railway lines in Data Prod
         $Localisateurs$ - set of all trackers
output:
         $assoc\_dest\_loc\_srv$ - set of all trackers with the corresponding
         SRV for the destination track ("voie de destination")
         $not\_found\_assoc\_dest\_loc\_srv$ - set of all trackers without
         matching any SRV for the destination track
1 foreach $line \in Line\_PROD$:
2     Take a line;
3     Select all SRV (except for junctions) and all module-window tracker
        pairs on that line;
4     Select all SRVs such that: Initial PK SRV $\leq$ Tracker PK + X AND
        Final PK SRV $\geq$ Tracker PK - X, where X is a threshold (in metres);
5     Semantic comparison between the 3 abbreviations for the destination
        track and the SRV name;
6 Creation of the tables $assoc\_dest\_loc\_srv$ and
    $not\_found\_assoc\_dest\_loc\_srv$ for found and not found matches
    respectively;
7 return $assoc\_dest\_loc\_srv$ and $not\_found\_assoc\_dest\_loc\_srv$;
```

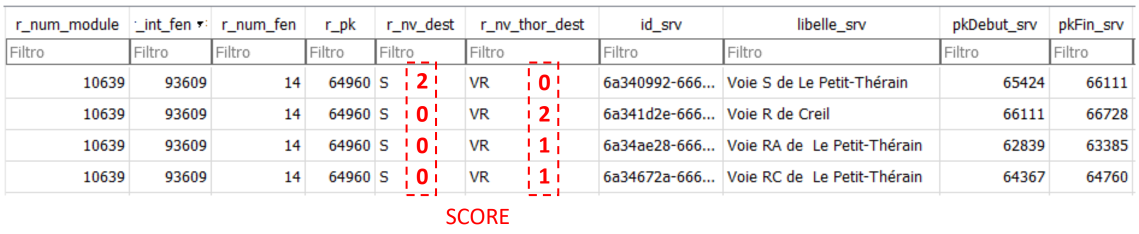
The only difference with the origin track algorithm (Alg.6) regards the choice of the list of SRV we select, where the following conditions must be satisfied:

$$\text{Initial PK SRV} \leq \text{Tracker PK} + \text{X} \qquad \text{AND} \qquad \text{Final PK SRV} \geq \text{Tracker PK} - \text{X}$$

where X is the threshold of 10 km we set up.
Namely, we consider all those SRVs such that overlap this spacial interval even if only for a few metres.
The semantic comparison is exactly the same as that made for the track of origin. The only difference concerning the algorithm is in case of multiple matches. In fact, since the spatial interval is much wider than in the case of the origin track, and since we only have two localisation systems instead of the three we had previously, there are many more multiple matches.
Therefore, it is necessary to develop a more precise algorithm that allows us to choose the destination track with highest likelihood.
The idea is the following:

- We assign a different score depending on the method used to obtain the match:
  - 4 points if the group of words is empty and we have only shared numbers;
  - 3 points if the words in the abbreviation are a sub-sequence of the starting letters of the words in the SRV name;
  - 2 points if the group of words are exactly the same;
  - 1 point if the words in the abbreviation are sub-sequences of some words in the SRV name;
  - 0 otherwise.
- After taking the matches with a higher score, I consider the ones with the highest number of matches between the 2 systems;
- I consider as final match that one with the closest corresponding SRV (the distance is 0 whenever the tracker belongs to the SRV).

For instance, consider this example:



r_num_module	_int_fen	r_num_fen	r_pk	r_nv_dest		r_nv_thor_dest		id_srv	libelle_srv	pkDebut_srv	pkFin_srv
Filtro	Filtro	Filtro	Filtro	Filtro		Filtro		Filtro	Filtro	Filtro	Filtro
10639	93609	14	64960	S	2	VR	0	6a340992-666...	Voie S de Le Petit-Thérain	65424	66111
10639	93609	14	64960	S	0	VR	2	6a341d2e-666...	Voie R de Creil	66111	66728
10639	93609	14	64960	S	0	VR	1	6a34ae28-666...	Voie RA de Le Petit-Thérain	62839	63385
10639	93609	14	64960	S	0	VR	1	6a34672a-666...	Voie RC de Le Petit-Thérain	64367	64760

Figure 3.9: Example of unique match for destination track in case of multiple matches.

We have inserted the scores next to each abbreviations for easier reading. Since the maximum between the last two scores is 1 and between the first two is 2, we discard last two rows. Then, in order to choose the best one, we look at PKs: we notice that the first SRV is closer to the tracker than the second one and, for this reason, this will be our best fit.

At the end of this phase we found 26771 matches out of a total of 40667 (**65.8%**). The percentage is lower than that one found with the origin track, but it is an expected result as the destination track may be very far away from the locator and then not found by the algorithm. Moreover, many trains terminate on service tracks that are not present on the SRV list and therefore cannot be detected.

3.2 Reliability of *Bouchon* File and Interaction with *X16*

In this section and in the following one we will work on the file *bouchon* constructed in chapter 3.1.1 concerning the association of SRVs with trackers physically attached to them together with concrete problems where these data are important, coming from the *X16* system (Fig. 3.1).
Our goals are as follows:

1. Complete and make more reliable the *bouchon* file through the stream of data coming from *X16*.

2. At the same time, update and/or complete the data coming from *X16*.

In fact, as we already discussed in the introduction of this chapter, the stream of data coming from *X16* is often inaccurate and need to be fixed by the *bouchon* model that is more robust. However, in these data there are often locators that have not been used in the creation of the *bouchon* file because a good match with a SRV has not been found, and they could therefore be added to our reference table (albeit with the right caution as we have no guarantee that they are definitely correct). In summary, we will make the *bouchon* model interact with *X16* in such a way as to improve both.
In the next sections we will consider just a small part of data coming from *X16*. The new trackers we will add to the *bouchon* file will be then only a part of the total number of trackers we could have. In fact, the next two sections are only an example of a methodology we can adopt to complete and make more reliable our reference table.

3.2.1 Anomaly Detection Problem

Every day, hundreds of trains cross the tracks of the French railway network and locators are useful tools for monitoring their transit.
In this section we will consider a file from *X16* that takes into account all train journeys in the week between 11-07-2021 and 18-07-2021.
We have many different train numbers. Each train run on one or more paths, sometimes similar and other times different, in the week under consideration. For each itinerary the trackers detect the time of passages, the railway lines and tracks they are on and the PKs. Therefore, the attributes we will consider are:

- **Numero_Sillon** (*Train Number*): unique number for a train.

- **Date_Heure_Depart_Theorique** (*Theoretical Departure Date*): departure time of the train (format: yyyy-mm-dd hh:mm:ss).

- **Date_Mesure** (*Measurement Date*): detection time of train passage (format: yyyy-mm-dd hh:mm:ss).

- **SRL**: ID of the SRL.

- **SRV**: ID of the SRV.

- **Libelle_Voie** (*Track Name*): name of the SRV.

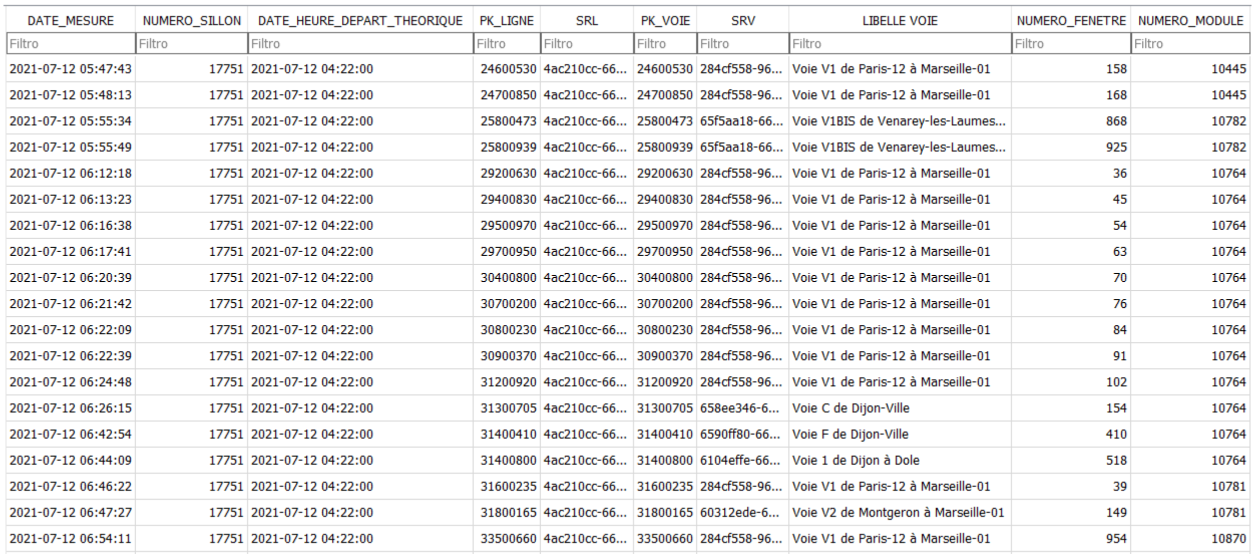
- **Pk_Ligne** (*Line PK*): tracker PK (Kilometric Point) on the detected SRL.

- **Pk_Voie** (*Track PK*): tracker PK on the detected SRV.

- **Numero_Module** (*Module Number*): module number of the tracker.

- **Numero_Fenetre** (*Window Number*): window number of the tracker.

These are the principal information that we can extract from trackers displaced all along the lines and thanks to which we can construct a time series for each train's path.
In order to better visualize these data, in Fig. 3.10 we have considered a part of the itinerary computed by the train number 17751 with departure time set to 2021-07-12 04:22:00.



DATE_MESURE	NUMERO_SILLON	DATE_HEURE_DEPART_THEORIQUE	PK_LIGNE	SRL	PK_VOIE	SRV	LIBELLE VOIE	NUMERO_FENETRE	NUMERO_MODULE
Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
2021-07-12 05:47:43	17751	2021-07-12 04:22:00	24600530	4ac210cc-66...	24600530	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	158	10445
2021-07-12 05:48:13	17751	2021-07-12 04:22:00	24700850	4ac210cc-66...	24700850	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	168	10445
2021-07-12 05:55:34	17751	2021-07-12 04:22:00	25800473	4ac210cc-66...	25800473	65f5aa18-66...	Voie V1BIS de Venarey-les-Laumes...	868	10782
2021-07-12 05:55:49	17751	2021-07-12 04:22:00	25800939	4ac210cc-66...	25800939	65f5aa18-66...	Voie V1BIS de Venarey-les-Laumes...	925	10782
2021-07-12 06:12:18	17751	2021-07-12 04:22:00	29200630	4ac210cc-66...	29200630	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	36	10764
2021-07-12 06:13:23	17751	2021-07-12 04:22:00	29400830	4ac210cc-66...	29400830	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	45	10764
2021-07-12 06:16:38	17751	2021-07-12 04:22:00	29500970	4ac210cc-66...	29500970	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	54	10764
2021-07-12 06:17:41	17751	2021-07-12 04:22:00	29700950	4ac210cc-66...	29700950	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	63	10764
2021-07-12 06:20:39	17751	2021-07-12 04:22:00	30400800	4ac210cc-66...	30400800	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	70	10764
2021-07-12 06:21:42	17751	2021-07-12 04:22:00	30700200	4ac210cc-66...	30700200	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	76	10764
2021-07-12 06:22:09	17751	2021-07-12 04:22:00	30800230	4ac210cc-66...	30800230	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	84	10764
2021-07-12 06:22:39	17751	2021-07-12 04:22:00	30900370	4ac210cc-66...	30900370	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	91	10764
2021-07-12 06:24:48	17751	2021-07-12 04:22:00	31200920	4ac210cc-66...	31200920	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	102	10764
2021-07-12 06:26:15	17751	2021-07-12 04:22:00	31300705	4ac210cc-66...	31300705	658ee346-6...	Voie C de Dijon-Ville	154	10764
2021-07-12 06:42:54	17751	2021-07-12 04:22:00	31400410	4ac210cc-66...	31400410	6590ff80-66...	Voie F de Dijon-Ville	410	10764
2021-07-12 06:44:09	17751	2021-07-12 04:22:00	31400800	4ac210cc-66...	31400800	6104effe-66...	Voie 1 de Dijon à Dole	518	10764
2021-07-12 06:46:22	17751	2021-07-12 04:22:00	31600235	4ac210cc-66...	31600235	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	39	10781
2021-07-12 06:47:27	17751	2021-07-12 04:22:00	31800165	4ac210cc-66...	31800165	60312ede-6...	Voie V2 de Montgeron à Marseille-01	149	10781
2021-07-12 06:54:11	17751	2021-07-12 04:22:00	33500660	4ac210cc-66...	33500660	284cf558-96...	Voie V1 de Paris-12 à Marseille-01	954	10870

Figure 3.10: Part of the itinerary of train number 17751 with departure time set to 2021-07-12 04:22:00.

After a deep pre-processing phase of the dataset, we want to:

1. Modify the detection of SRVs by trackers through the *bouchon* file.

2. Detect anomalies in the updated version of the *X16* file.

3. Update the *bouchon* file by adding all trackers that are present uniquely in the *X16* file.

We analyse now step by step this procedure.

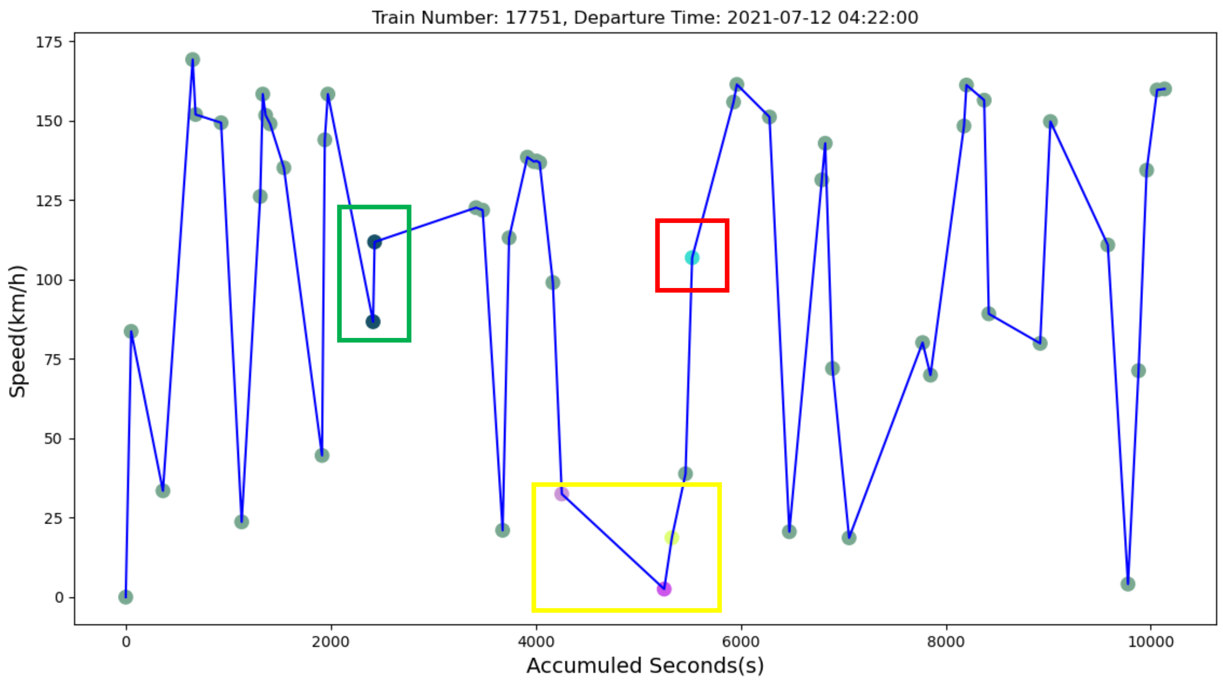
1. Modify the detection of SRVs by trackers through the *bouchon* file.
The first problem is updating the *X16* file with the *bouchon* file. To start with, we perform a short descriptive analysis of the data. In total we have 777 trackers in the file *X16* (detected as pairs *module_number* - *window_number*) out of 51274 rows, due to the fact that the pathways on which the trackers are located are used more times by different trains on different dates. Of these 777 trackers, 554 are shared with the *bouchon* model and 223 are unique in the *X16* file. We will add these 223 trackers to *bouchon* model in a second moment, after making some considerations.
We focus on the 554 shared trackers: 432 detect the same SRVs as in *bouchon* model, whereas 122 find a different SRV ID. To sum up, on average 1 locator out of 6 detect a different SRV between the two tables.

2. Detect anomalies in the updated version of the *X16* file.
After having updated the *X16* file, we look for anomalies. But firstly we must provide a definition of anomaly.

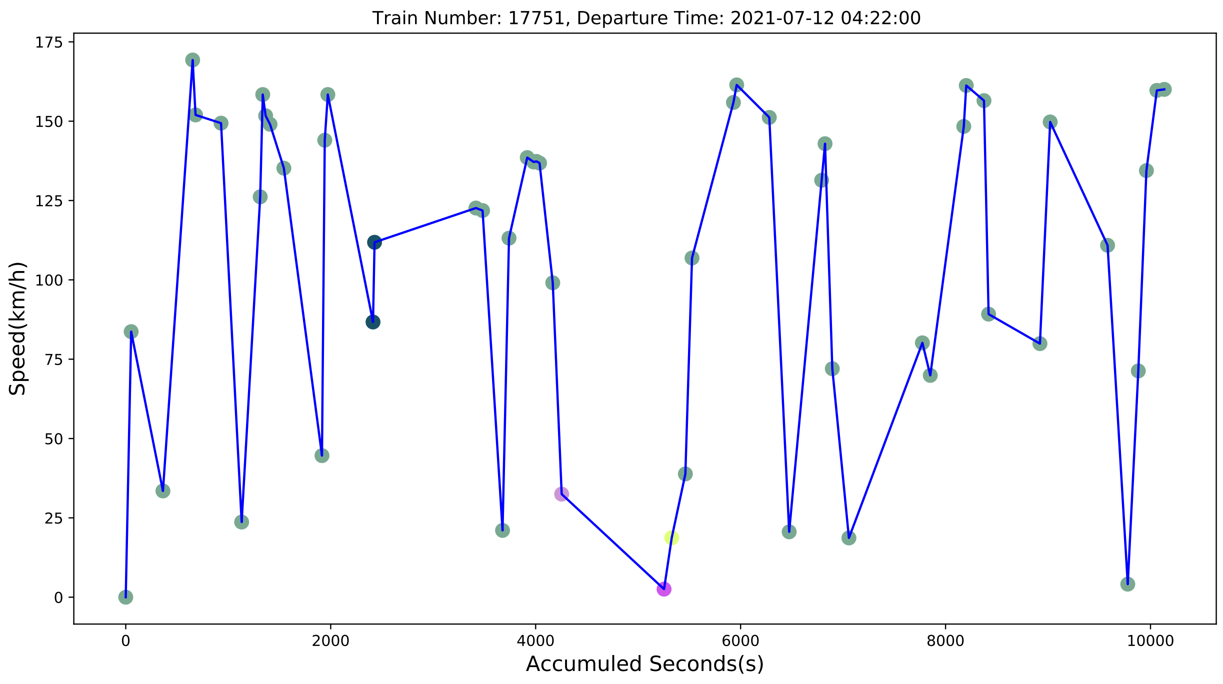
We define an **anomaly** as an **ABA** or **ABB** or **ABC** sequence belonging to the same itinerary (i.e. same train number and same departure date) and repeated more times, where A, B and C stand for three different SRV IDs, such that the average train speed is above a certain threshold in a small time interval around the detection of the central track.

In other words, we know that a train can change its track at a certain maximum speed, which corresponds to 30 km/h or 60 km/h depending on the type of railway switch. If the train switches from track A to track B and then back to track A in few seconds with a speed higher than the maximum allowed, this sequence is probably an anomaly and should be corrected in the AAA sequence.
As far as the ABB and ABC sequences are concerned, these are a bit more difficult to correct because more combinations are possible. However, the same argument applies as for ABA sequences, as a track switch cannot be performed above a certain speed threshold.
In particular, we decided to take all sequences with time intervals of **less than 4 minutes** and speeds **above 60km/h** OR with time intervals of **less than 8 minutes** and speeds **above 100km/h**. The reason why we put two conditions is not to loose possible anomalies when the time window is large. In these cases, logically the threshold for speed must be increased.
In order to compute the train speed, the best we can do is to use the difference of PKs and the difference of detection times. Obviously, the result obtained will be an average of the train speed between two successive measurement times and will not be particularly accurate, especially when the time gap is large (another reason why we look for small time windows).

In total, **43** ABA unique sequences are found after the update with *bouchon* model, of which **2** are possible anomalies. Instead, **183** ABB and ABC sequences are found, of which **37** are possible anomalies.
We take now under analysis the itinerary we have considered before as example (Fig. 3.10).
In Fig. 3.11 we have two graphs of the train speeds and times detected by the trackers along its itinerary: the upper plot is the original one, the lower one follows a possible modification.
In the y-axis we have the speed in km/h, whereas along the x-axis we have the accumulated seconds from the start of the path. We point out that we have considered the accumulated seconds for two reasons: to better visualize the time gaps between two sequential detections (longer is the segment, wider is the time interval) and not to have overlaps of labels in the plot in case we had taken the measurement dates.
Each point corresponds to a detection of a particular locator and has as coordinates the detected speed of the train and the time of passage. Points have a different color according to their ID.
Notice, as one can imagine, that the speed of the train fluctuates continuously be-



(a)



(b)

Figure 3.11: (a) Speeds detected by trackers in the original file. (b) Speeds detected by trackers after anomaly resolution.

tween higher values, when the train is between two stations for example, and lower values, in the proximity of stations or interchange points.
The horizontal distance between two points indicates how much time has elapsed between two measurements (this data is not very clear from this plot, but by zooming in on the original output it is possible to better quantify this time interval).
In the graph all points are in light grey except for points inside the 3 rectangles. These are the SRVs with a different ID. The red rectangle has only one dot, that forms an ABA sequence. The green one, instead, enclose 2 dots with same dark grey color, composing a ABB sequence. Finally, the yellow rectangle contains three points of different colors, that do not compose any particular sequence among those we are looking for.
The points inside the first two rectangles are the most problematic ones, because the train here was detected at higher speeds (on average by considering the previous detection) than the points inside the yellow square, which probably are in the proximity of a station.
We focus now on the ABA sequence (light blue point in the red rectangle): this small table describes the data of the SRV B detection and the second SRV A detection (we insert only the first 8 characters of the SRV ID for size issues).

ABA Sequence			
SRV IDs	284cf558	60312ede	284cf558
Loc. Numbers	10781-39	10781-149	10870-954
	Distance(m)	**Time(s)**	**Speed(km/h)**
Detect. B	1930	65	106.89
Detect.(2^{nd}) A	17495	404	155.90

Table 3.1: Distance, Time and Speed detection for an ABA sequence.

As we can see, we are within the range of values we set for detecting an anomaly (just looking at the speed, these are all over 100km/h with a time interval below the 8 minutes) and for this reason in the below plot of Fig. 3.11 we colored in light grey the anomalous point, namely as the other "ordinary" points.
We proceed similarly with the ABB sequence inside the green box. These are the measured data:

ABB Sequence			
SRV IDs	284cf558	65f5aa18	65f5aa18
Loc. Numbers	10445-168	10782-868	10872-925
	Distance(m)	**Time(s)**	**Speed(km/h)**
Detect. B	10623	441	86.71
Detect. A	466	15	111.84

Table 3.2: Distance, Time and Speed detection for an ABB sequence.

Although the values of the second detection are within the range of the parameters of an anomaly, the speed detected by the first SRV B is not high enough (less than 100km/h with a time interval of 441 seconds (7 minutes and 21 seconds)) and therefore we left the SRVs unchanged.

As mentioned above, this detection method is very approximate and is based on the considerations listed above. However, it provides a first preliminary step to find possible anomalies, which will then be analysed one by one with other tools. In addition, a new speed detection system will be introduced in few months, which will provide data on train speeds in real time, representing a more precise method of finding these anomalies.

3. Update the *bouchon* file by adding all trackers that are present uniquely inthe *X16* file.

As we mentioned before, there are 223 locators that are present in the *X16* file but not in the *bouchon* file.

In order to update the *bouchon* model with these new trackers, we followed the procedure divided in 3 steps that is explained in the diagram in Fig. 3.12.

At the end of Chapter 3.1.1 we introduced 2 tables of not found matches: a larger

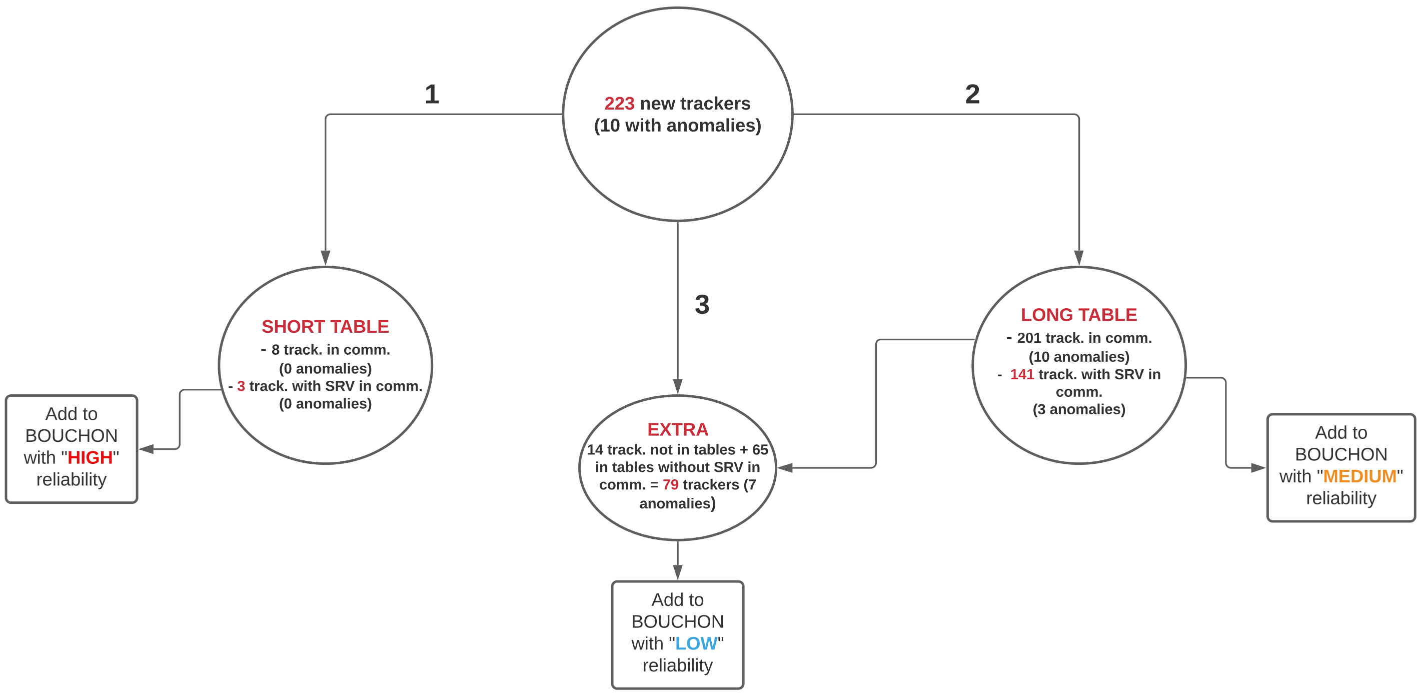

223 new trackers (10 with anomalies)
1
2
3
SHORT TABLE
- 8 track. in comm. (0 anomalies)
- 3 track. with SRV in comm. (0 anomalies)
LONG TABLE
- 201 track. in comm. (10 anomalies)
- 141 track. with SRV in comm. (3 anomalies)
EXTRA
14 track. not in tables + 65 in tables without SRV in comm. = 79 trackers (7 anomalies)
Add to BOUCHON with "HIGH" reliability
Add to BOUCHON with "MEDIUM" reliability
Add to BOUCHON with "LOW" reliability

Figure 3.12: Diagram that explains the updating procedure for *bouchon* model for the detection problem (the order of the 3 steps is highlighted with the numbers).

table, created considering only the criterion of the same group of numbers, and a smaller one, created taking into account also the cases where all the words of the abbreviations were found in the SRV name, but this also contains something extra. We have firstly checked that all 223 trackers belong to that 30% of matches not found, but to go further we want to verify if any of these are contained in the two tables. In order to update the *bouchon* model we started the comparison with the short table, since this contains matches not found with a higher probability of being good matches (i.e. false negatives). We concluded that 8 trackers out of 223 are in common with this table, of which **3** identify the same SRV of *X16* file. We added these 3 trackers, with the associated SRVs, to the *bouchon* file with "HIGH" reliability. At the same time, we removed them from both the "not found" tables.

After that, we moved to the long table and we found out that 201 locators are shared, of which 141 identify the same SRV of *X16* file. As before, we added these **141** locators, with the associated SRVs, to the *bouchon* file with "MEDIUM" reliability, since this table has lower probability of containing false negative matches, and we remove them from the two tables.
Finally, we considered the **79** "extra" trackers left and we added them to the *bouchon* model with "LOW" reliability.
As far as the anomalies are concerned, 10 out of 39 we have found before are in the 223 new locators, whereas the others are in the *bouchon* model (see diagram for more details). We decided not to edit the *bouchon* file with the SRV found by the anomaly, but to add them separately as they have to be analysed with other methods.
These results confirm the presence of some good matches in the two tables, which can therefore be exploited to increase the reliability and completeness of the *bouchon* table.

3.2.2 Missing Data Problem

Data coming from *X16* not only might be inaccurate as we have seen in the previous section, but they may have some missing values.
In this final part of our work, we analyse some train itineraries on 3 and 4 November 2021 that present some missing data for the SRV and the name of the track. The objectives are basically the same as those we discussed in the introduction to Chapter 3.2, and specifically we want to:

1. Modify the detection of SRVs by trackers through the *bouchon* file.

2. Fill some missing values.

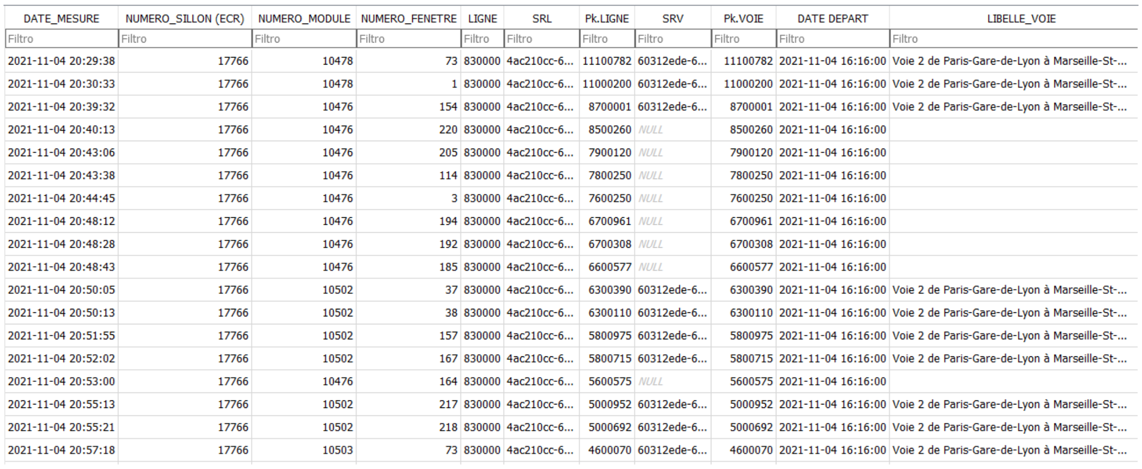
3. Update the *bouchon* file by adding all trackers that are present uniquely in the *X16* file.

Also in this case we go through the procedure step by step.

1. Modify the detection of SRVs by trackers through the *bouchon* file.
The structure of the dataset is the same as before with the same features we described in Chapter 3.2.1.
In total we have 1331 unique trackers in this file out of 7490 rows. Of these, 1123 are shared with the *bouchon* model (after the update of the anomaly detection problem!) and 208 are unique in the *X16* file. These are all present in the list of all trackers except for one, that is a new locator since this file is more recent than *bouchon* file. What is surprising, however, is that all 208 locators do not identify any tracks, or rather, the SRV that should be identified by them is missing from the table. At the end of the filling phase, we will add to the *bouchon* file only the filled data we found.

2. Fill some missing values.
The original dataset has 2624 missing SRVs (out of 7490 rows), but after updating with the *bouchon* table, this number decreases to 672, meaning that almost the 75% of missing values has been filled by our reference table. However, since the 208 locators we want to add are unique in this dataset, they still remain without SRV.

In Fig. 3.13 we inserted part of the table referred to a particular train itinerary. Note that sometimes we have many missing values in a row, sometimes only one hole. In the following, we will consider only the last case, because with our data it is difficult and dangerous to fill in so many close missing data and would compromise the reliability of the *bouchon* model. The algorithm used is illustrated in Alg. 9.



DATE_MESURE	NUMERO_SILLON (ECR)	NUMERO_MODULE	NUMERO_FENETRE	LIGNE	SRL	Pk.LIGNE	SRV	Pk.VOIE	DATE DEPART	LIBELLE_VOIE
Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
2021-11-04 20:29:38	17766	10478	73	830000	4ac210cc-6...	11100782	60312ede-6...	11100782	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...
2021-11-04 20:30:33	17766	10478	1	830000	4ac210cc-6...	11000200	60312ede-6...	11000200	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...
2021-11-04 20:39:32	17766	10476	154	830000	4ac210cc-6...	8700001	60312ede-6...	8700001	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...
2021-11-04 20:40:13	17766	10476	220	830000	4ac210cc-6...	8500260	NULL	8500260	2021-11-04 16:16:00	
2021-11-04 20:43:06	17766	10476	205	830000	4ac210cc-6...	7900120	NULL	7900120	2021-11-04 16:16:00	
2021-11-04 20:43:38	17766	10476	114	830000	4ac210cc-6...	7800250	NULL	7800250	2021-11-04 16:16:00	
2021-11-04 20:44:45	17766	10476	3	830000	4ac210cc-6...	7600250	NULL	7600250	2021-11-04 16:16:00	
2021-11-04 20:48:12	17766	10476	194	830000	4ac210cc-6...	6700961	NULL	6700961	2021-11-04 16:16:00	
2021-11-04 20:48:28	17766	10476	192	830000	4ac210cc-6...	6700308	NULL	6700308	2021-11-04 16:16:00	
2021-11-04 20:48:43	17766	10476	185	830000	4ac210cc-6...	6600577	NULL	6600577	2021-11-04 16:16:00	
2021-11-04 20:50:05	17766	10502	37	830000	4ac210cc-6...	6300390	60312ede-6...	6300390	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...
2021-11-04 20:50:13	17766	10502	38	830000	4ac210cc-6...	6300110	60312ede-6...	6300110	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...
2021-11-04 20:51:55	17766	10502	157	830000	4ac210cc-6...	5800975	60312ede-6...	5800975	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...
2021-11-04 20:52:02	17766	10502	167	830000	4ac210cc-6...	5800715	60312ede-6...	5800715	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...
2021-11-04 20:53:00	17766	10476	164	830000	4ac210cc-6...	5600575	NULL	5600575	2021-11-04 16:16:00	
2021-11-04 20:55:13	17766	10502	217	830000	4ac210cc-6...	5000952	60312ede-6...	5000952	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...
2021-11-04 20:55:21	17766	10502	218	830000	4ac210cc-6...	5000692	60312ede-6...	5000692	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...
2021-11-04 20:57:18	17766	10503	73	830000	4ac210cc-6...	4600070	60312ede-6...	4600070	2021-11-04 16:16:00	Voie 2 de Paris-Gare-de-Lyon à Marseille-St-...

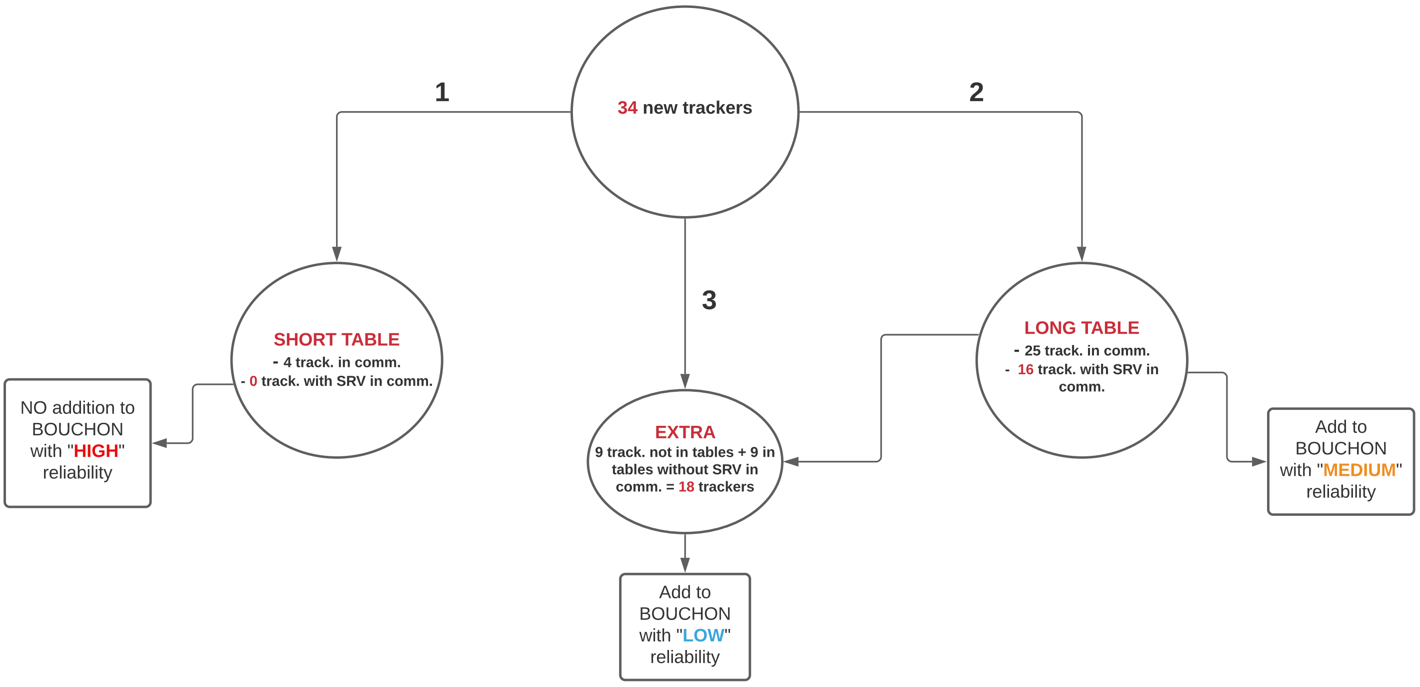
Figure 3.13: Part of the *X16* table with some missing values.

Cycling on each railroad way, we consider successive sequences of 3 locators (or 2, in case of beginning or end of sequence) and we associate the tracker with the missing SRV with the most similar one. In the case of sequences of 3, the first similarity criterion is the SRL, i.e. the railway line on which the locator lies.
If the previous or the following tracker with respect to the central missing one shares the same SRL, we copy the data from that SRV. If, however, the three lines are all different, then we skip this sequence, as the missing data belongs to another line and we have no clues about the possible track.
Finally, there is the case where the three tracks are on the same railway line. In this case, if the previous SRV is the same as the next one, it is logical to think that my missing data follows the same trend. If, however, the preceding SRV is different from the following one, one can use PK as a second similarity criterion. Logically, the missing SRV will be more similar to the closest one in terms of distance. However, we prefer to omit this association because it may create too many false positives and compromise the reliability of the *bouchon* file.
In case of initial or final sequence (the first two conditions in Alg. 9), we simply compare the SRL with the second and penultimate row respectively and, if they are the same, we copy otherwise we skip.

After this procedure, we have filled 118 new rows. Therefore, there are still 554 missing values, that we will leave blank in order not to risk having too many false positives, since with the information provided we cannot go further.

3. Update the *bouchon* file by adding all trackers that are present uniquely in the *X16* file
Finally, we update the *bouchon* table with the new trackers. These are 208 in total,

but originally none of these identified an SRV. After the filling algorithm we implemented, we managed to fill 34 of these. As we have done with the anomaly detection problem, we checked whether these 34 locators belong to the two not found tables and we added with different reliability level depending on the table where we found them. We started from the short table: here only **4** trackers are found, but none of them also share the same SRV. For this reason, we didn't add any trackers with "HIGH" reliability level.
Then, we moved to the long table. We found out that 25 trackers share the same module and window numbers and, in particular, 16 of them have the same SRV as well. We added these **16** locators to *bouchon* table with "MEDIUM" reliability level.
Finally, we added the remaining **18** locators with "LOW" reliability level.
The diagram in Fig. 3.14 summarizes what we have just explained. We recall that



34 new trackers
1
2
3
SHORT TABLE
- 4 track. in comm.
- 0 track. with SRV in comm.
LONG TABLE
- 25 track. in comm.
- 16 track. with SRV in comm.
EXTRA
9 track. not in tables + 9 in tables without SRV in comm. = 18 trackers
NO addition to BOUCHON with "HIGH" reliability
Add to BOUCHON with "MEDIUM" reliability
Add to BOUCHON with "LOW" reliability

Figure 3.14: Diagram that explains the updating procedure for *bouchon* file for the missing data problem (the order of the 3 steps is highlighted with the numbers).

these new detected trackers that we added to *bouchon* model, for both the anomaly detection and missing values problems, are just a small part of the total we could have. Indeed, as we mentioned above, we are just considering some examples of data coming from *X16* in order to show a possible method for completing and making more reliable our reference table. As we said in the introduction, this is an exploratory part of our work, that it will be exploited in future thanks to new more precise tools that will be available.

Algorithm 9: Filling sequences with a missing data

```
input  : $D$ - Train itineraries table with missing values
output: $D$ - Train itineraries table with (partial) filled values
foreach $(train\_number, departure\_time) \in D$:
    Select the relevant sub-table $S$;
    foreach $i \in range(len(S))$:
        /* Start of sequence                                  */
        if $i = 0$:
            Select SRL of row[0] and row[1];
            if $SRL[0] == SRL[1]$:
                SRV_ID[0] = SRV_[1];
                SRV_name[0] = SRV_name[1];
            else:
                continue;
        /* End of sequence                                    */
        elif $i = len(S)$:
            Select SRL of row[len(S)] and row[len(S)-1];
            if $SRL[len(S)] == SRL[len(S) - 1]$:
                SRV_ID[len(S)] = SRV_[len(S)-1];
                SRV_name[len(S)] = SRV_name[len(S)-1];
            else:
                continue;
    Select SRL of row[i-1], row[i] and row[i+1];
    if $SRL[i-1]! = SRL[i]! = SRL[i+1]$:
        continue;
    elif $SRL[i] == SRL[i-1]$:
        SRV_ID[i] = SRV_[i-1];
        SRV_name[i] = SRV_name[i-1];
    elif $SRL[i] == SRL[i+1]$:
        SRV_ID[i] = SRV_[i+1];
        SRV_name[i] = SRV_name[i+1];
    elif $SRL[i-1] == SRL[i] == SRL[i+1]$:
        if $SRV\_ID[i-1] == SRV\_ID[i+1]$:
            SRV_ID[i] = SRV_[i-1];
            SRV_name[i] = SRV_name[i-1];
        /* I can assign $i-1$ or $i+1$ since they are the same   */
        else:
            continue;
return $D$;
```

Chapter 4

Conclusions

In this work, we have addressed two very important issues from the practical point of view of data analysis of the French railway network.
The first one concerns the transition from the old and time frozen version of *GAÏA* to the current and dynamic one, which is used every day for the creation of fundamental tools for the consumer. Several machine learning algorithms have been used, leading to excellent results and discrete improvements over simpler algorithms, without having a particularly high computational cost. In addition, we created a useful plot for visualizing SRVs along a single railway line, where we compared with different colours the tracks that have been modified from Data Prep to Data Prod, and an another interesting graph for comparing TIVs that make up a specific SRV.
The second topic was the creation of a model, the *bouchon* file, which will be used nationwide as a reference model for the association of locators with railway tracks. This is a part that is still under development at *SNCF Réseau*. The contribution of this work was the creation of a first version of the model based on semantic comparison algorithms between words which allowed to couple about 70% of the total of the trackers with their respective SRVs. Moreover, after showing the limitations of the algorithm, we created two tables that can be used to increase the number of matches.
In the final part, an exploratory method has been proposed, which exploits the dynamic data flow of *X16* as a way for completing and making the *bouchon* table more reliable. At the same time, we showed how the *bouchon* model can be adopted for improving this data flow by also adjusting any anomalies and filling in missing data. The contribution of this last part of the work was therefore to provide a good starting method that will be improved in the future by the introduction of new and more efficient data collection tools.

Bibliography

Bishop, C. M. (2006). *Pattern Ricognition and Machine Learning.* Springer.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu (1996). 'A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise'. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.* KDD'96. Portland, Oregon: AAAI Press, pp. 226–231.

Rahmah, N. and I. S. Sitanggang (2016). 'Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatras'. In: *IOP Conf. Series: Earth and Environmental Science* 31.

Rousseeuw, Peter J. (1987). 'Silhouettes: A graphical aid to the interpretation and validation of cluster analysis'. In: *Journal of Computational and Applied Mathematics"* 20, pp. 53–65.

Sander, J., M. Ester, and H.P. Kriegel (1998). 'Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications'. In: *Data Mining and Knowledge Discovery* 2, pp. 169–194.