Smart network operator platform enabling shared, integrated and more sustainable urban freight logistic
[D2.1] - State of the art in optimization and machine leaning algorithms applied to last mile logistics

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 861540
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 861540

<table>
<thead>
<tr>
<th>Deliverable No.</th>
<th>Work Package No.</th>
<th>WP2</th>
<th>Task/s No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>WP2</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

### Deliverable No. 2.1: State of the art in optimization and machine learning algorithms applied to last mile logistics

<table>
<thead>
<tr>
<th>Work Package Title</th>
<th>Status</th>
<th>Dissemination level</th>
<th>Due date deliverable</th>
<th>Deliverable version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation &amp; Execution for dynamic planning using simulation and learning</td>
<td>Draft</td>
<td>PU</td>
<td>30.06.2021</td>
<td>1.0 – final fully integrated version with all comments after internal QA ready for submission</td>
</tr>
</tbody>
</table>

**Linked Task/s**
- **T2.1 Review of the State of Art in optimisation and machine learning methods for dynamic planning**

**Document Contributors**

**DELIVERABLE RESPONSIBLE**
- **Contributors**: Frank Werner, Margarete Steudter, Antonio Masegosa, Jenny Fajardo, Nadia Giuffrida
- **Organization**: Software AG, UDEUSTO, UCD

**F. WERNER (SOFTWARE AG)**
- **Reviewers**: Jorge Zapico, Enrique Onieva, Asier Moreno, Pilar Elejoste
- **Organization**: CORREOS, UDEUSTO

**Document History**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2020-11-18</td>
<td>Early TOC Draft</td>
</tr>
<tr>
<td>0.2</td>
<td>2020-11-26</td>
<td>Update and discussion with DEUSTO and SAG</td>
</tr>
<tr>
<td>0.3</td>
<td>2021-03-08</td>
<td>Finalizing TOC with all task participants</td>
</tr>
<tr>
<td>0.4</td>
<td>2021-04-30</td>
<td>1st merged version with inputs from SAG, UDEUSTO, UCD</td>
</tr>
<tr>
<td>0.5</td>
<td>2021-05-17</td>
<td>2nd integrated version with inputs from SAG, UDEUSTO, UCD</td>
</tr>
<tr>
<td>0.6</td>
<td>2021-06-01</td>
<td>3rd integrated version ready for internal QA</td>
</tr>
<tr>
<td>0.7</td>
<td>2021-06-14</td>
<td>4th integrated version with most fixes from QA</td>
</tr>
<tr>
<td>1.0</td>
<td>2021-06-21</td>
<td>final fully integrated version with all comments after internal QA ready for submission</td>
</tr>
</tbody>
</table>
Table of contents

1 Executive Summary............................................................................................................................................. 8
2 Introduction....................................................................................................................................................... 9
   2.1 Purpose..................................................................................................................................................... 9
   2.2 Relation to other Deliverables.................................................................................................................. 9
3 Machine Learning Models.................................................................................................................................. 11
   3.1 Introduction.............................................................................................................................................. 11
      3.1.1 Data Analytics................................................................................................................................... 11
   3.2 Real-time Streaming Analytics.................................................................................................................. 12
      3.2.1 IoT and Real-world Streaming Analytics ........................................................................................ 13
   3.3 Different Types of Machine Learning....................................................................................................... 14
   3.4 Supervised Machine Learning................................................................................................................... 14
      3.4.1 Regression.......................................................................................................................................... 15
      3.4.2 Classification..................................................................................................................................... 16
   3.5 Unsupervised Machine Learning................................................................................................................ 22
      3.5.1 Clustering.......................................................................................................................................... 22
   3.6 Other Machine Learning Models/Types....................................................................................................... 26
   3.7 Neural Networks......................................................................................................................................... 27
      3.7.1 Feed-forward Neural Networks (FFNN)............................................................................................. 28
      3.7.2 Long Short-term Memory (LSTM) ..................................................................................................... 30
      3.7.3 Convolutional Neural Networks (CNN)............................................................................................ 31
      3.7.4 Recurrent Neural Networks (RNN)................................................................................................... 34
      3.7.5 Deep Learning................................................................................................................................... 36
   3.8 Best Practices from Industrial Use Cases.................................................................................................... 36
      3.8.1 Big Data Pipeline for AI..................................................................................................................... 36
      3.8.2 Lambda Architecture........................................................................................................................ 38
      3.8.3 AI Model Management...................................................................................................................... 40
   3.9 Review of most recent applications domains............................................................................................... 40
      3.9.1 Smart Logistics.................................................................................................................................... 41
      3.9.2 Urban Logistics ................................................................................................................................... 44
      3.9.3 Anomaly Detection............................................................................................................................ 46

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 861540
## 3.10 Benchmarks and Datasets

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10.1 Benchmarks in AI</td>
<td>48</td>
</tr>
<tr>
<td>3.10.2 SVHN – Dataset</td>
<td>48</td>
</tr>
<tr>
<td>3.10.3 CIFAR-100 - Dataset</td>
<td>50</td>
</tr>
<tr>
<td>3.10.4 KITTI Optical Flow</td>
<td>51</td>
</tr>
</tbody>
</table>

## 4 Optimization Models and Techniques for Dynamic Planning in Logistics

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Introduction</td>
<td>53</td>
</tr>
<tr>
<td>4.2 Mathematical Models for Vehicle Routing Optimization</td>
<td>54</td>
</tr>
<tr>
<td>4.2.1 Rich VRP</td>
<td>55</td>
</tr>
<tr>
<td>4.2.2 Stochastic VRP</td>
<td>59</td>
</tr>
<tr>
<td>4.2.3 Dynamic VRP</td>
<td>59</td>
</tr>
<tr>
<td>4.2.4 Multi-objective VRP</td>
<td>61</td>
</tr>
<tr>
<td>4.3 Optimization techniques</td>
<td>63</td>
</tr>
<tr>
<td>4.3.1 Exact algorithms</td>
<td>63</td>
</tr>
<tr>
<td>4.3.2 Approximate algorithms/ Metaheuristics</td>
<td>64</td>
</tr>
<tr>
<td>4.3.3 Hybrid algorithms</td>
<td>65</td>
</tr>
<tr>
<td>4.4 Current software tools for dynamic planning in logistic</td>
<td>66</td>
</tr>
<tr>
<td>4.4.1 JSprit</td>
<td>67</td>
</tr>
<tr>
<td>4.4.2 OR-Tools</td>
<td>68</td>
</tr>
<tr>
<td>4.4.3 VROOM</td>
<td>69</td>
</tr>
<tr>
<td>4.4.4 VRP Service from ArcGIS (ESRI)</td>
<td>69</td>
</tr>
<tr>
<td>4.4.5 Circuit</td>
<td>70</td>
</tr>
<tr>
<td>4.4.6 LOCUS</td>
<td>70</td>
</tr>
<tr>
<td>4.4.7 OptaPlanner</td>
<td>71</td>
</tr>
<tr>
<td>4.4.8 Here</td>
<td>71</td>
</tr>
<tr>
<td>4.4.9 GraphHopper</td>
<td>71</td>
</tr>
<tr>
<td>4.4.10 Carto</td>
<td>72</td>
</tr>
<tr>
<td>4.4.11 DELMIA Quintiq</td>
<td>72</td>
</tr>
<tr>
<td>4.5 Assessment of different optimization models and techniques</td>
<td>72</td>
</tr>
<tr>
<td>4.5.1 Assessment of mathematical models for the VRP</td>
<td>73</td>
</tr>
<tr>
<td>4.5.2 Assessment of optimization techniques for the VRR</td>
<td>74</td>
</tr>
<tr>
<td>4.5.3 Assessment of software tools</td>
<td>75</td>
</tr>
<tr>
<td>4.6 Benchmarks and Datasets</td>
<td>77</td>
</tr>
</tbody>
</table>
List of tables

Table 1 SVHN - Datasets and Benchmarks
Table 2 CIFAR-100 -- Datasets and Benchmarks
Table 3 KITI Optical Flow -- Datasets and Benchmarks
Table 4 Comparative analysis of software tools for dynamic planning in logistics according to license and Rich VRP features
Table 5 Comparative analysis of software tools for dynamic planning in logistics according to Stochastic, Dynamic and Multi-Objective VRP features

List of figures

Figure 1 Hierarchy of different AI Modell-Sets
Figure 2 Different Correlator Queries for Real-time Streaming Analytics
Figure 3 Types of Machine Learning
Figure 4 Sample Plot to exemplify the idea of linear regression
Figure 5: Decision Boundary on a training set [165]
Figure 6 Hyperplane through two linearly separable classes [1]
Figure 7 Hyperplane through two non-linearly separable [1]
Figure 8 Visualization of the soft-margin SVM [1]
Figure 9 Plot Showing K-means evolution (plot created with [5])
Figure 10 Examples showing the downside of use-less clustering produced by K-means
Figure 11 Example plots of different hierarchical structures
Figure 12 Commonly used activation functions [3]
Figure 13 Feed-forward NN with different hidden layers [3]
Figure 14 Detailed schematic of the Simple Recurrent Network (SRN) unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network [3]
Figure 15 A Generic Simplified CNN Architecture for MNIST [2]
Figure 16 Figures detected by the MINST Approach [2]
Figure 17 Kernel moving an input image and producing an output image [2]
Figure 18 A Simple RNN with three layers [161]
Figure 19 A Simple RNN with recurrent connections [161]
Figure 20: Big Data Pipeline for Data Processing in SENATOR
Figure 21: Overview of the Data Analytics and CEP module
Figure 22. Information Flow within the AI Processing Chain
Figure 23. Information Flow for selected SENATOR data sources
Figure 24: The total amount of compute, in petaflop/s-days used to train selected results [163]
Figure 25: increasing number of options in the AI development stack [162]
Figure 26. Evolution of the number of publications containing “Vehicle Routing Problem” keywords on Scopus on April 27th, 2021
Figure 27 Main characteristics of RVRP (Extracted from [128])
Figure 28 Classification of constraints in RVRP
Figure 29 Classification of Stochastic VRP’s problems
Figure 30 Example of dynamic vehicle routing [142] .................................................. 60
Figure 31 Classification of Dynamic VRP’s problems .................................................. 61
Figure 32 Example Non-dominated solutions called Pareto set .................................. 62
Figure 33 Model’s classification and optimisation techniques [107] ............................... 63
Figure 34 JSprit general metaheuristic algorithm flowchart ....................................... 68

List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>AAC</td>
<td>Artificial Ant Colony</td>
</tr>
<tr>
<td>ACO</td>
<td>Ant colony optimization</td>
</tr>
<tr>
<td>AD</td>
<td>Anomaly Detection</td>
</tr>
<tr>
<td>APLS</td>
<td>Automated Parcel Locker System</td>
</tr>
<tr>
<td>BN</td>
<td>Bayesian networks</td>
</tr>
<tr>
<td>CEP</td>
<td>Complex Event Processing</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>DL</td>
<td>Deep Learning</td>
</tr>
<tr>
<td>DT</td>
<td>Digital Twin</td>
</tr>
<tr>
<td>DVRP</td>
<td>dynamic VRP</td>
</tr>
<tr>
<td>GELS</td>
<td>Gravitational Emulation Local Search</td>
</tr>
<tr>
<td>GHG</td>
<td>Greenhouse Gas</td>
</tr>
<tr>
<td>GM</td>
<td>Graphical Models</td>
</tr>
<tr>
<td>GPUs</td>
<td>Graphical Processing Unit</td>
</tr>
<tr>
<td>GRNN</td>
<td>generalized regression neural network</td>
</tr>
<tr>
<td>HVRP</td>
<td>Heterogeneous VRP</td>
</tr>
<tr>
<td>ILPs</td>
<td>integer linear programs</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transportation Systems</td>
</tr>
<tr>
<td>kNN</td>
<td>k-nearest neighbours algorithm</td>
</tr>
<tr>
<td>LNS</td>
<td>Large Neighbourhood Search</td>
</tr>
<tr>
<td>LP</td>
<td>linear programming</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long-Short Term Memory</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MNIST</td>
<td>Modified National Institute of Standards and Technology</td>
</tr>
<tr>
<td>MOVPR</td>
<td>multi-objective VRP</td>
</tr>
<tr>
<td>NB-IoT</td>
<td>Narrowband Internet of Things</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>OR</td>
<td>Operational Research</td>
</tr>
<tr>
<td>PMML</td>
<td>Predictive Model Mark-up Language</td>
</tr>
<tr>
<td>PSO</td>
<td>particle swarm optimization</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>RVRP</td>
<td>Rich Vehicle Routing Problems</td>
</tr>
<tr>
<td>SA</td>
<td>Simulated annealing</td>
</tr>
<tr>
<td>SRN</td>
<td>Simple Recurrent Network</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>SVRP</td>
<td>stochastic VRP</td>
</tr>
<tr>
<td>TPU</td>
<td>Tensor Processing Unit</td>
</tr>
<tr>
<td>TS</td>
<td>Traveling Salesman</td>
</tr>
<tr>
<td>TSP</td>
<td>Travelling Salesman Problem</td>
</tr>
<tr>
<td>UMP</td>
<td>Urban Mobility Package</td>
</tr>
<tr>
<td>VNS</td>
<td>Variable Neighbourhood Search</td>
</tr>
<tr>
<td>VRP</td>
<td>Vehicle Routing Problem</td>
</tr>
<tr>
<td>VRPSC</td>
<td>stochastic customers</td>
</tr>
<tr>
<td>VRPSDC</td>
<td>stochastic demands and customers</td>
</tr>
<tr>
<td>VRPSTS</td>
<td>stochastic travel and service times</td>
</tr>
</tbody>
</table>
1 Executive Summary

The present document is dedicated to a state of the art (SotA) analysis in dynamic planning and assesses suitable technologies, methodologies, and highlights best standards and approaches. It represents D2.1 “Review of the state of the art in optimization and machine learning methods for dynamic planning” from WP2 “Operation & Execution for dynamic planning using simulation and learning” with focus on design and development of the dynamic planning module of SENATOR. It is grouped in two main areas such as optimization and machine learning to provide individually for both an in-depth analysis and examples from industry. This deliverable has an is self-explaining character as it provides algorithmic details and examples which make it comprehensible and a useful SotA.

In the area of Machine Learning it provides the technological base by investigating different fields of data analytics and defines a hierarchy of different AI model-sets which encompass Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) as different complexity classes. This will be the base technology applied to all SENATOR AI problems. This analysis further discusses SotA of streaming analytics, a technique to analyse, monitor and aggregate real-time event that is up today very established in the field of IoT and event stream processing. A classification of machine learning into supervised and unsupervised learning paves the ground for applying different classification, regression, and clustering methods to logistic problems. A short wrap-up of the Lambda Architecture, an AI data-pipeline, an information flow cycle in AI and prominent applications eventually conclude the Machine Learning part.

The second part of this document related to simulation and optimization captures optimization models and techniques for dynamic planning. From a mathematical point different models for vehicle touring problems (VRP) and optimization are assessed along with their problem domain. These include the Rich VRP, Stochastic VRP, Dynamic VRP and Multi-Objective VRP along with their individual advantages and examples. The list of optimization techniques which are described further include exact algorithms, approximation algorithms and hybrid forms of them as being lately researched. A list of available tools for dynamic planning along with an assessment of different optimization models concludes this second part.
2 Introduction

2.1 Purpose

Having a solid technological foundation is one of the main success criteria of an IT-related project. Such a document shall summarize the state of the art (SotA) of what is currently and generally accepted as good practice in IT and technology – both from a scientific, use case driven and technological viewpoint. SotA does not necessarily imply the most technologically advanced solution but reflects the state which is generally acknowledged.

This document endeavours to deliver the SotA in the context of SENATOR by making an in-depth study related to optimization models, techniques for dynamic planning in logistics as well as machine learning algorithms and strives to highlight improvements for network and urban planning. It is further intended to review state-of-the-art in dynamic planning of last mile logistics and to define benchmark instances and data sets, as well as baseline of performance for optimization and machine learning algorithms. It does so by providing exact mathematical models which are complemented a class of approximation algorithms to establish the trade-off between time and prevision which is very often needed. On one hand, the mathematical models used to abstract the processes to be optimized (e.g. linear programming, probabilistic programming, etc.); and on the other hand, the optimization techniques most employed as exact methods, heuristics and metaheuristics, among others.

As for Machine Learning algorithms, the SotA includes a review of existing models ranging from Support Vector Machines up to Neural Networks (RRNN, CNN, RNN, LSTM, etc.) are presented and provide best industrial approaches in data processing, architectural design decisions and a processing cycle for AI models.

In the second part, state of the art in optimization is summarized in terms of the VRP optimisation models and those optimization methods most aligned with SENATOR’s objectives, as well as to assess their main strengths and weaknesses. In these sections, VRP variants are described that are most aligned with SENATOR’s objectives and which show the state-of-the-art regarding most common used optimisation methods to solve these VRP variants. Major available software platforms and tools are described for vehicle routing optimisation and strengths and weaknesses of the different VRP optimisation models and methods are discussed.

All algorithms strive to maximise the use of realistic datasets which are indispensable to obtain effective and precise models. Hence datasets will be created from real data coming from the use cases foreseen in the project and updated along the next tasks as new data and sub-systems are released to ensure the alignment of the developments within the SENATOR project.

2.2 Relation to other Deliverables

D2.1 is loosely coupled with the use cases of work package 1. However, it is indirectly influenced by T1.1 “Use Case Definition and IT requirements” which set the focus and use cases under investigation. T2.1 is the first deliverable in WP2 and as such builds up the base for “T2.2 Design of optimization models for Dynamic planning” by working out and highlighting State of the Art in optimization and
dynamic planning. It further paves the ground for T2.3 “Development of Algorithms for Dynamic Optimization” by thorough study of models and underlying algorithms.
3 Machine Learning Models

3.1 Introduction

The use of machine learning has grown to a revolutionary potential shifting from classical programmatic approaches to an unforeseen capability in solving complex problems. Thanks to machine learning, more information than ever before can be efficiently processed and transformed from purely uninterpreted data points to value and business insights that can derive decision-making, improve customer experiences and even a decade ago unthinkable problem solving.

The set of machine learning algorithms is rich, ranging from classification of problems, detection of pattern, to speech or image analytics. The following sections will detail the SotA in Machine Learning and its application within the SENATOR project.

3.1.1 Data Analytics

Data analytics is the science of analysing raw data to make conclusions about that information. It comprises the processes, tools and techniques of data analysis and management, including the collection, organization, and storage of data. The aim of data analytics is to apply statistical analysis and technologies on data to find trends and solve problems.

Data analytics is a broad field. There are four primary types of data analytics: descriptive, diagnostic, predictive and prescriptive analytics [11]. Each type has a different goal and a different place in the data analysis process.

- **Descriptive analytics** helps answer questions about what happened. These techniques summarize large datasets to describe outcomes to stakeholders. The output provides essential insights into past performance.
- **Diagnostic analytics** focuses on why something happened. These techniques supplement more basic descriptive analytics. They take the findings from descriptive analytics and dig deeper to find the cause. The performance indicators are further investigated to discover why they got better or worse. This generally occurs in three steps: Identify anomalies in the data, data that is related to these anomalies is collected and statistical techniques are used to find relationships and trends that explain these anomalies.
- **Predictive analytics** helps answer questions about what will happen in the future. These techniques use historical data to identify trends and determine if they are likely to recur. Its techniques include a variety of statistical and machine learning techniques.
- **Prescriptive analytics** suggests what should be done. By using insights from predictive analytics, data-driven decisions can be made.

These types of data analytics provide the insight that businesses need to make effective and efficient decisions. Used in combination they provide a well-rounded understanding of a company’s needs and opportunities.

Companies can use the insides of data analytics to improve their decision making and can do more effective marketing. Additionally, costs can be saved because customer needs are better known[35].
3.1.2 Overview of Artificial Intelligence (AI)

Artificial intelligence (AI) is the overarching discipline that covers anything related to making machines smarter. It is a program that can sense, reason, act and adapt. Machine Learning (ML) is often mistakenly called AI when in fact it is a subset of AI. ML refers to systems that can learn by their own. It involves algorithms whose performance improves when exposed to more data over time. A machine learning algorithm can be trained on a small sample of data, and the system will continue to learn as it gathers more data, becoming more accurate as time goes on. Deep Learning (DL) is a subset of ML in which multi-layered neural networks learn from vast amounts of data.

![Hierarchy of different AI Model-Set](image)

Figure 1 Hierarchy of different AI Modell-Sets

3.2 Real-time Streaming Analytics

Streaming Analytics has been a hot terminology in the business domain for the last decade as companies seek to harness benefits and add value from the inexorably growing volume of data. As the value of data is very often time critical and correlated with a business value (the earlier situations are detected, the earlier a proper reaction can be automatically triggered which in-turn minimizes loss and maximises gain) [9].

Real-time computing provides the system with the ability to learn automatically and improved from experience without being explicitly programmed. This means a system’s learning model is updated whenever new data is added to the system.

With real-time data analytics, errors can be known instantly. The knowledge about errors helps organizations to respond to such faults more quickly and increase the operational efficiency of the enterprise. Additionally, companies can progress with customer trends. Real-time data analytics will let them know the competitors’ strategy, promotions, customer preferences and gives information about the recent trends in the market. This information will help to make changes to the product as per the needs of the customers. On the other hand, the output is just as good as the input. An ML model cannot predict accurately if there is not enough data. To develop a model that delivers the desired business outcomes, the data science team will have to ask for sufficient, relevant data.

Complex Event Processing uses events and their pattern to filter, aggregate and query discrete events and event streams [10].

As shown in Figure 2 below events are monitored by the correlator which implements discrete event matching which could be a temporal or spatial matching or the matching of a longer sequence of events. In addition, aggregation of the event stream is realized using time windows, an object...
bounding box (OBB) or potentially any custom aggregation function. The correlator further provided correlation of event queries and apply detection of relationships of long running temporal pattern.

Figure 2 Different Correlator Queries for Real-time Streaming Analytics

### 3.2.1 IoT and Real-world Streaming Analytics

Streaming analytics gained increasingly attention in the business world for the past few years. As companies seek to harness, and benefit from the inexorably growing volume of data, these data volumes can hardly be managed in reasonable time using conventional approaches. Especially, as our world is accelerating continuously, the benefit and added value from data can only be realized if it is captured in time, instead of hours later [36][37]. Streaming Analytics enables to access, analyse, and act on both historical and real-time/fast-moving live data to determine if there is an issue relating to equipment and to prevent future problems. There several examples and the list can be arbitrary extended if needed:

- It enables to spot problems before they become an issue and hereby early detect signs for possible equipment failure.
- Analyse this information, learn from it and act
- Prevent certain events from happening in the first place. You can predict and detect significant risk and maximize your gain.
- Companies are relying on real-time analytics as big and fast data proliferates and more and more data streams are generated in real time from the IoT as well as markets, mobile devices, clickstreams, and internal transactional systems.

Streaming real-time analytics allows to design, develop and deploy sophisticated analytics that monitor any number of event streams and event data of any kind. They are often used to detect and analyse patterns from many sources at the same time and react with countermeasures. As there is a continuous monitoring in place, response to events may happen at the time they occur—or even before, when using predictive models. They further allow to automate responses to take intelligent actions instantly, without human intervention or Spot significant patterns of events, like a change in pressure or temperature, which could indicate pending equipment failure.

In addition, using machine learning models, streaming analytics can supplement or replace manual processes with automated systems using statistically derived actions in critical processes. No matter where your applications are running — whether in IoT, a distributed environment, the cloud or on a mainframe machine — you can execute, optimize and scale models without allocating dedicated IT resources using Machine Learning Models [37].
Using streaming analytics different deployments can be realized that also allow edge, on-premise, or cloud set-ups. To analyse and filter data at a local level before passing it to the back-end for more processing, data can be analysed in edge deployments reducing network congestion at a higher level and hereby increase overall system’s efficiency.

### 3.3 Different Types of Machine Learning

In the area of Machine Learning there exist a tremendous number of different models. However, in general there is no “the best model” as every model has its strengths and weaknesses. Therefore, choosing the right model highly depends on the use case. In the process of choosing a model, the principle of Occam’s Razor [13] applies, which says that simpler solutions are more likely to be correct than complex ones. This means in terms of model selection, if two models perform equally in the evaluation, the less complex is the better one.

In the following section the most prominent examples of machine learning algorithms will be discussed. The algorithms are divided in two different categories, supervised and unsupervised learning.

![Diagram of Types of Machine Learning](image)

#### 3.4 Supervised Machine Learning

Supervised learning means, that the machine is trained using data which is well labelled. This means, some data is already tagged with the correct answer as a “training set”. A supervised learning algorithm learns from labelled training data, that helps to predict outcomes for unforeseen data. In concrete terms, this means there is a set of n observations \( \{(x_i, y_i)\}_{i=1}^{n} \), each tuple consist of an...
input variable $x_i$ and a corresponding label $y_i$. All methods have in common, that they try to find a function $f \in F := \{ f : \mathbb{R}^d \rightarrow \mathbb{R}, x \mapsto \langle w, x \rangle + b \mid w \in \mathbb{R}^d, b \in \mathbb{R} \}$, which minimizes a certain criterion.  

One of the algorithms is used to learn the mapping function from the input to the label, $y_i = f(x_i)$. The goal is, to approximate the mapping function $f$ so well that when there is new input data $x_i$, the output variables $y_i$ is predicted for that data.

### 3.4.1 Regression

Regression algorithms attempt to estimate the mapping function $f$ from their input variables $x_i$ to numerical or continuous output variables $y_i$. Regression models a target prediction value based on independent variables. It is mostly used for forecasting and finding out cause and effect relationship between variables. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering, and the number of independent and dependent variables being used.

#### 3.4.1.1 Linear Least Squares Regression

Linear Regression is a supervised learning technique, which is used to find a linear relationship between the input variables and the labels. The variable $w$ is called weight Vector and $b$ bias. Note, this does not mean, that only linear functions can be learned, since a linear combination of nonlinear features still yields a nonlinear function. The following plot illustrates the general idea:

![Sample Plot to exemplify the idea of linear regression](image)

One often used variant of linear Regression is the so called ordinary linear least squares regression (OLLSR). This approach tries to fit a function which minimizes the squared Euclidean distance between the prediction and the actual data points. The optimization objective can be defined as follows:

$$
\min_{f \in F} \sum_{i=1}^{n} (y_i - f(x_i))^2 = \min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2} \sum_{i=1}^{n} (y_i - (\langle w, x_i \rangle + b))^2
$$
16

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 861540

For convenience one defines a new matrix $X'$, which is the same as $X$ except that a column of ones is added. Furthermore, one adds a new parameter to the weight vector. As a result, the objective can be written in vector notation as follows:

$$
\min_{w} \frac{1}{2} \| Y - X'w \|_2^2
$$

The objective function is obviously convex; hence it is necessary and sufficient for a global minimum to set the derivative with respect to $w$ equal to zero.

$$
\nabla_w \frac{1}{2} \| Y - X'w \|_2^2 = -X'^T(Y - X'w) = 0
$$

$$
\Rightarrow X'^TX'w = X'^TY
$$

If $X'^TX$ is invertible then the solution for $w$ is unique, in the case where $X'^TX$ is not invertible one has to solve a linear equation of the form $Ax = b$, but the minimizer is not unique anymore. Linear least squares regression has earned its place as the primary tool for process modelling because of its effectiveness and completeness. It is simple to implement and the output coefficients are easy to interpret. On the other hand, the main disadvantage of linear least squares regression are limitations in the shapes that linear models can assume over long ranges, possibly poor extrapolation properties, and sensitivity to outlier [14].

3.4.2 Classification

Classification algorithms attempt to estimate the mapping function $f$ from their input variables $x_i$ to discrete output variables $y_i$. The main goal is to identify in which class the new data will fall into; Classes are sometimes called as targets, labels, or categories. Classification can perform on structured or unstructured data. Apart from Bayesian Networks (BN), k-nearest neighbours algorithm (k-NN), and Support Vector machines (SVM), also other clustering methods exists like decision trees or random forest which will be treated in this document.

3.4.2.1 Bayesian networks (BNs)

Bayesian networks (BNs) [164], also known “as belief” networks(or Bayes nets for short), belong to the family of probabilistic graphical models (GMs). These graphical structures are used to represent knowledge about an uncertain domain. In particular, each node in the graph represents a random variable, while the edges between the nodes represent probabilistic dependencies among the corresponding random variables. These conditional dependencies in the graph are often estimated by using known statistical and computational methods. Hence, BNs combine principles from graph theory, probability theory, computer science, and statistics.

A BN reflects a simple conditional independence statement [164]. Namely that each variable is independent of its non-descendants in the graph given the state of its parents. This property is used to reduce, sometimes significantly, the number of parameters that are required to characterize the JPD of the variables. This reduction provides an efficient way to compute the posterior probabilities given the evidence.
3.4.2.2 k-nearest Neighbour Algorithm (k-NN)

The k-nearest Neighbour Algorithm is a type of supervised ML algorithm mainly used for classification of predictive problems although it could also be applied on regression problems. It represents a non-parametric classification which is very useful for non-linear data because there is no assumption about data needed and it delivery relatively high accuracy. Today there are much better supervised learning models than KNN which make the algorithm often obsolete. Especially as its computation is memory-wise expensive as it stores all training data which makes prediction slow with increasing data size.

![Figure 5: Decision Boundary on a training set](165)

3.4.2.3 Support vector machines

A support vector machine is a binary linear classification model (binary in this context means that all labels are either +1 or -1 [1]). First it is assumed that the data is linearly separable, that means it exists a hyperplane which perfectly separates the two classes from each other. A hyperplane can be described by the following equation

\[ \langle w, x \rangle + b = 0, \quad w \in \mathbb{R}^d, \quad b \in \mathbb{R}. \]

Support Vectors are the data points closest to the separating hyperplane and the aim of Support Vector Machines (SVM) is to orientate this hyperplane in a way such that the minimal distance between any data point and the hyperplane is maximized, that is why SVM’s are also called maximum margin classifiers.
State of the art in optimization and machine learning algorithms applied to last mile logistic

The plot implies that one wants to find \( w \) and \( b \) such that:

\[
\langle w, x_i \rangle + b \geq 1, \text{ if } y_i = +1 \quad (1.1)
\]
\[
\langle w, x_i \rangle + b \leq -1, \text{ if } y_i = -1 \quad (1.2)
\]

Combining these two equations yields:

\[
\forall i, y_i ((\langle w, x_i \rangle + b) - 1 \geq 0
\]

The hyperplanes H1 and H2 on which the support vectors lie (shown in circles), can be described by the following equations:

\[
\langle w, x_i \rangle + b = +1 \quad \text{for H1}
\]
\[
\langle w, x_i \rangle + b = -1 \quad \text{for H2}
\]

As described above the goal is maximize the minimal distance between any data point and the hyperplane, this is equivalent to maximizing the distance between H1 and H2. By geometry, this distance can be computed by \( 2/\|w\| \).

As a result, the optimization objective can be formulated as follows:

\[
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \|w\|
\]

s.t. \( \forall i, y_i ((\langle w, x_i \rangle + b) - 1 \geq 0 \)

Since minimizing \( \|w\| \) is equivalent to minimizing \( \frac{1}{2} \|w\|^2 \), the objective above can be rewritten as a quadratic program:

\[
\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2} \|w\|^2
\]

s.t. \( \forall i, y_i ((\langle w, x_i \rangle + b) - 1 \geq 0 \)
This constrained optimization problem can be transformed into an unconstrained optimization problem by taking advantage of the Lagrange Multiplier technique. So, let $\alpha_i \geq 0, \forall i$ then the objective can be rewritten as:

$$O_p(w, b, \alpha) = \min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2} \|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i((w, x_i) + b) - 1]$$

To find the optimal values for $w$ and $b$ it is sufficient to set the derivatives equal to zero.

$$\nabla_w O_p \nabla 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad (*)$$

$$\frac{\partial O_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \quad (**)$$

Substituting (*) and (**) into $O_p$ yield the so called dual Lagrangian formulation:

$$O_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

So, the constrained optimization problem for the dual formulation can be written as:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

s.t. $\forall i, \alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

$\alpha$ can be determined by any QP-solver, as a result $w$ can be computed as stated in (*). The next goal is to compute $b$. This can be achieved by noting that for any support vector $x_s$, $y_s((x_s, w) + b) = 1$ holds. Consequently, $b$ can be determined by solving for $b$ in the equation above. Instead of using an arbitrary Support Vector, it is better to take an average of all Support Vectors.

In all derivations above it was always assumed that the data is linearly separable, but in practice this is nearly never the case. To address this problem, so called slack variables $\xi_1 \ldots \xi_n \geq 0$ are introduced, which allow misclassifications of some training samples. This approach is called soft margin SVM. The equations (1.1) and (1.2) change to:

$$\langle w, x_i \rangle + b \geq +1 - \xi_i, \text{ if } y_i = +1 \quad (1.1.s)$$

$$\langle w, x_i \rangle + b \leq -1 + \xi_i, \text{ if } y_i = -1 \quad (1.2.s)$$

This can again be combined to:

$$\forall i, y_i ((w, x_i) + b) - 1 + \xi_i \geq 0$$
The idea of the soft-margin SVM is to penalize data points which are misclassified according to distance between the wrongly classified point and the decision boundary, therefore the objective can be adjusted as follows:

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

s.t. $\forall i, y_i((w, x_i) + b) - 1 + \xi_i \geq 0$

The parameter $C$ controls the trade-off between the slack variable penalty and the size of the margin. The derivation of $w$ and $b$ is similar to the hard margin case, but a bit more involved, therefore it is omitted.

Both described approaches above use the data only through inner products, because of that observation one can extend these linear classifiers to model non-linear decision boundaries.

The best way to understand this method is to look at an example. The data set, which is depicted in Figure 2-6, is obviously not separable by a hyperplane, but if one maps the data to a three-dimensional space using the following feature mapping the data becomes linearly separable:

$$\Phi: \mathbb{R}^2 \to \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$$

The application of this mapping is shown on the right of Figure 2-6.
The decision boundary in the transformed space will be of the form \( \langle w, z \rangle + b = 0 \), which reduces to

\[
w_1 x_1^2 + w_2 \sqrt{2} x_1 x_2 + w_3 x_2^2 = 0.
\]

So, the linear classifier can be used without any modifications to the actual algorithm to classify a dataset which is not linearly separable in the original space. The reason for this result is that the algorithm only uses the Gram Matrix \( K \) of the dataset, so after the computation of this matrix no explicit information of the data is used anymore.

\[
K = \begin{bmatrix}
\langle x_1, x_1 \rangle & \ldots & \langle x_1, x_n \rangle \\
\vdots & \ddots & \vdots \\
\langle x_n, x_1 \rangle & \ldots & \langle x_n, x_n \rangle
\end{bmatrix} = XX^T
\]

The Gram Matrix of the transformed data has a similar structure.

\[
K = \begin{bmatrix}
\langle \Phi(x_1), \Phi(x_1) \rangle & \ldots & \langle \Phi(x_1), \Phi(x_n) \rangle \\
\vdots & \ddots & \vdots \\
\langle \Phi(x_n), \Phi(x_1) \rangle & \ldots & \langle \Phi(x_n), \Phi(x_n) \rangle
\end{bmatrix}
\]

Let \( r \) and \( s \) be vectors in \( \mathbb{R}^3 \) corresponding to \( a \) and \( b \) respectively, then the dot product of \( r \) and \( s \) can be computed as follows:

\[
\langle r, s \rangle = r_1 s_1 + r_2 s_2 + r_3 s_3 = a_1^2 b_1^2 + 2 a_1 a_2 b_1 b_2 + a_2^2 b_2^2 = \langle a, b \rangle^2
\]

So instead of mapping the data to another space and then computing the inner product one can combine these two steps in one operation so the mapping becomes implicit. In fact, it is not even
necessary to know $\Phi$, it suffices to know how to compute the dot product between two transformed data points, and this operation is called a kernel $K(x, y)$.

Support Vector Machines work relatively well when there is a clear margin of separation between classes and they are more effective in high dimensional spaces. They are also effective in cases where the number of dimensions is greater than the number of samples. On the other hand, Support Vector Machines are not suitable for large data sets and do not perform very well when the data set has more noise [15].

### 3.5 Unsupervised Machine Learning

Unsupervised learning is a machine learning technique, where is no need to supervise the model. This means, there is input data $x_i$ but no corresponding output variables. Instead, the model works on its own to discover information and group the data by finding similarities, differences, and patterns. The goal for unsupervised learning is to model the underlying structure or distribution in the data to learn more about the data.

Apart from Clustering other unsupervised fields are being research such as dimensionality reduction, association rules but they are out of scope of this deliverable and will hence not be illustrated in the following.

#### 3.5.1 Clustering

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects based on similarity and dissimilarity between them. Clustering can take large datasets and, without instruction, quickly organize them into something more usable.

##### 3.5.1.1 K-means

K-means is a simple and widely used technique of clustering. It is completely based on partitioning methodology [4]. It partitions n-data items into k groups where k indicates the number of clusters specified by the user. Clusters are formed such that each item in the cluster has minimum distance from the centroid. To calculate the distance between item and centroid, k means algorithm uses the Euclidean distance measurement. It aims to minimize the sum of squared distances between all points and the cluster centre. This procedure consists of following steps:

**Input:** $K$: the number of desired clusters.

**Output:** A set of $k$ clusters

**Algorithm:**

1. Choose several clusters $k$
2. Randomly select $k$ objects as initial centroids $c_1 \ldots c_k$
3. Calculate the Euclidean distance between each data point $x_i$ and each centroid $c_j$ and assign each data point to its closest centroid.
4. Calculate the mean of every cluster to create new centroids
5. Repeat step 2 and 3 until the new computed centroids do not change anymore.
The following plot shows a visual execution of the above described algorithm, the triangles represent the centroids (read from left to right).

![Image of K-means evolution]

Figure 9 Plot Showing K-means evolution (plot created with [5])

The main advantages of K-means consist in fast convergence of the algorithm and interpretability of the results, but different execution will probably yield different results, since the initial centroids are chosen randomly. Furthermore, the algorithm is sensitive towards outliers and biased towards spherical clusters, due to usage of Euclidean distance as a measure, as a consequence K-means will produce nonsense clusters if the data is of non-spherical shape, which can be seen in the plot below.
K-means is fast, simple, relatively easy to implement and scales to large data sets. Additionally, with many variables, the computationally is faster than other clustering algorithms, if the K is small. However, it is difficult to find the optimal k, and centroids can be dragged by outliers. K-means has trouble clustering data where clusters are of varying sizes and density and the order of the data has an impact on the result [16][17].

3.5.1.2 Hierarchical clustering

The results of applying K-means or clustering algorithms depend on the choice for the number of clusters to be searched and on the starting configuration assignment. In contrast, hierarchical clustering methods do not require such specifications [18]. Instead, they require the user to specify a measure of dissimilarity between (disjoint) groups of observations, based on the pairwise dissimilarities among the observations in the two groups. As the name suggests, they produce hierarchical representations in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level. At the lowest level, each cluster contains a single observation. At the highest level there is only one cluster containing all data.

Strategies for hierarchical clustering are divided into two basic paradigms: agglomerative (bottom-up) and divisive (top-down). Agglomerative strategies start at the bottom and at each level recursively merge a selected pair of clusters into a single cluster. This produces a grouping at the next higher level with one less cluster. The pair chosen for merging consists of the two groups with the smallest intergroup dissimilarity. Divisive methods start at the top and at each level recursively split one of the existing clusters at that level into two new clusters. The split is chosen to produce two new groups with the largest between-group dissimilarity. With both paradigms there are N - 1 levels in the hierarchy. Each level of the hierarchy represents a particular grouping of the data into disjoint clusters of observations. The entire hierarchy represents an ordered sequence of such groupings. It is up to the user to decide which level (if any) actually represents a “natural” clustering in the sense that observations within each of its groups are sufficiently more similar to each other than to observations assigned to different groups at that level. Recursive binary splitting/agglomeration can be represented by a rooted binary tree. The nodes of the trees represent groups. The root node represents the entire data set. Each one of the N terminal nodes represent one of the individual observations (singleton clusters). Each nonterminal node (“parent”) has two daughter nodes.
For divisive clustering the two daughters represent the two groups resulting from the split of the parent; for agglomerative clustering the daughters represent the two groups that were merged to form the parent.

All agglomerative and some divisive methods (when viewed bottom-up) possess a monotonicity property. That is, the dissimilarity between merged clusters is monotone increasing with the level of the merger. Thus, the binary tree can be plotted so that the height of each node is proportional to the value of the intergroup dissimilarity between its two daughters. The terminal nodes representing individual observations are all plotted at zero height. This type of graphical display is called a dendrogram. A dendrogram provides a highly interpretable complete description of the hierarchical clustering in a graphical format. Cutting the dendrogram horizontally, at a particular height, partitions the data into disjoint clusters represented by the vertical lines that intersect it. These are the clusters that would be produced by terminating the procedure when the optimal intergroup dissimilarity exceeds that threshold cut value. Groups that merge at high values, relative to the merger values of the subgroups contained within them lower in the tree, are candidates for natural clusters. Note that this may occur at several different levels, indicating a clustering hierarchy: that is, clusters nested within clusters. Such a dendrogram is often viewed as a graphical summary of the data itself, rather than a description of the results of the algorithm. However, such interpretations should be treated with caution. Different hierarchical methods (see below) as well as small changes in the data can lead to quite different dendrogram. Also, such a summary will be valid only to the extent that the pairwise observation dissimilarities possess the hierarchical structure produced by the algorithm. Hierarchical methods impose hierarchical structure whether such structure exists in the data.

The main advantage of hierarchical clustering is that the clusters are not assumed to be globular and it is easy to implement. It is not possible to undo the previous step: once the instances have been assigned to a cluster, they can no longer be moved around [17].
3.6 Other Machine Learning Models/Types

The previously presented models are relevant for SENATOR, but there are more well-known models. Some are presented below, with a focus on anomaly detection techniques.

3.6.1.1 Bayesian belief networks

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. It is also called a Bayes network, belief network, decision network, or Bayesian model. A Bayesian belief network can be defined as “a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph.” [19] Bayesian networks are probabilistic, because these networks are built from a probability distribution, and also use probability theory for prediction and anomaly detection.

Real world applications are probabilistic in nature, and to represent the relationship between multiple events, a Bayesian network is needed. It can also be used in various tasks including prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction, and decision making under uncertainty. Bayesian Network can be used for building models from data and experts’ opinions, and it consists of two parts: Directed Acyclic Graph and Table of conditional probabilities.

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an Influence diagram. A Bayesian network graph is made up of nodes and Arcs using directed links, where:
- Each node corresponds to the random variables, and a variable can be continuous or discrete.
- Arc or directed arrows represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.
- These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other.

The Bayesian network has mainly two components: Causal Component and Actual numbers. Each node in the Bayesian network has condition probability distribution \( P(X_i | \text{Parent}(X_i)) \), which determines the effect of the parent on that node.

3.6.1.2 Hidden Markov model (HMM)

Hidden Markov model (HMM) is a statistical Markov model in which the system being modelled is assumed to be a Markov process \( X \) with unobservable (“hidden”) states [20]. HMM assumes that there is another process \( Y \) whose behaviour depends on \( X \). The goal is to learn about \( X \) by observing \( Y \).

HMM stipulates that, for each time instance \( n_0 \), the conditional probability distribution of \( Y_{n0} \) given the history \( \{X_n = x_n\}_{n \leq n_0} \) must not depend on \( \{x_n\}_{n \leq n_0} \). Hidden Markov models are known for their applications to thermodynamics, statistical mechanics, physics, chemistry, economics, pattern recognition and bioinformatics.

3.6.1.3 K-nearest neighbour (KNN)

K-nearest neighbour (KNN) is a density-based technique. It is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems [21][21]. In both cases, the input consists of the k closest training example in data set. The output depends on whether KNN is used for classification or regression: in KNN classification, the output is a
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 861540

class membership; in KNN regression, the output is the property value of the object. This value is the average of the values of k nearest neighbours.

KNN is a type of classification where the function is only approximated locally, and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbours, so that the nearer neighbours contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbour a weight of \(1/d\), where \(d\) is the distance to the neighbour.

The neighbours are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the k-NN algorithm is that it is sensitive to the local structure of the data.

### 3.6.1.4 Fuzzy logic-based outlier detection

The basic idea of Fuzzy logic-based outlier detection lies in the consideration that in a dataset an outlier is markedly smaller or larger than other values, i.e. that is far from the dataset centroid, has few points in its own neighbourhood and has a quite low degree of membership to the cluster to which it could be assigned from a clustering algorithm [22].

Thus, the proposed approach exploits both the consideration of neighbouring points and the capability of clustering algorithms of pointing out outliers.

The calculation of the centroid, that is simply the mean vector of the data distribution, and the clustering operation, that is performed by means of the Fuzzy C Means algorithm (FCM), must be computed preliminary. After this, for each pattern the following four features are extracted:

1. the distance between each element and the centroid of the overall data distribution, normalized with respect to the average value.
2. the fraction of the total number of elements that are near to the pattern itself.
3. the mean distance between the considered pattern and a fraction (usually 20%) of the remaining patterns, normalized with respect to the maximum value. The data belonging to such fraction are randomly chosen, by eventually discarding those that have been already marked as outliers.
4. the degree of membership of the patterns to the cluster to which it has been assigned by the preliminary FCM clustering stage

The four above listed features are fed as inputs to a Fuzzy Inference System (FIS) that gives as output an index in the range \([0,1]\) that provides an indication on the risk that the considered pattern is an outlier.

### 3.7 Neural Networks

Artificial Neural Networks, or shortly Neural Networks (NN), have been successfully applied to many diverse fields. Pattern classification/recognition; system modelling and identification; signal processing; image processing; control systems and stock market predictions are some of those main
fields of engineering and science. This, of course, can be attributed to the many useful aspects of neural networks, such as their parallel structure, learning and adaptive capabilities. Furthermore, Very Large Scale Integrated (VLSI) principles are relatively easy to implement and fault tolerant.

3.7.1 Feed-forward Neural Networks (FFNN)

NN, as the name implies, are inspired from their biological counterparts, the biological brain, and the nervous system. Biological brain is entirely different than the conventional digital computer in terms of its structure and the way it processes information. In many ways, biological brain (or human brain as the example attributed with the highest level of intelligence) is far more advanced and superior to conventional computers. The most important distinctive feature of a biological brain is its ability to learn and adapt, while a conventional computer does not have such abilities. The basic building block of neural networks is a neuron. A neuron can be perceived as a processing unit. In a neural network, neurons relate to each other through weights. Each neuron in a network receives weighted information via these synaptic connections from the neurons that it is connected to and produces an output by passing the weighted sum of those input signals (either external inputs from the environment or the outputs of other neurons) through an activation function. The most used activation functions can be seen in the Figure below.

![Figure 12 Commonly used activation functions](image)

Feed-forward neural networks fall into two categories depending on the number of the layers, either single layer or multi-layer.
Although the representation above is good to intuitively understand how a Neuronal Network (NN) is designed, this is not how NN’s work in a mathematical sense. In the following the mathematical representation of a NN will be briefly explained using the one-hidden-layer NN in the figure above as an example. The activation function that will be used for both the hidden and the output layer is the sigmoid function, denoted by \( \sigma \).

First some definitions are needed, let:

- \( x \in \mathbb{R}^3 \) be the input vector
- \( b^h \in \mathbb{R}^4 \) be the bias vector of the hidden layer
- \( b^o \in \mathbb{R}^4 \) be the bias vector of the output layer
- \( w^h_{ij} \) be the weight from the \( i \)-th input neuron to the \( j \)-th hidden neuron
- \( w^o_{ij} \) be the weight from the \( i \)-th hidden neuron to the \( j \)-th output neuron

The output of the \( i \)-th hidden unit is computed by:

\[
a_i = \sigma(x_1 w^h_{1i} + x_2 w^h_{2i} + x_3 w^h_{3i} + b^h_i).
\]

The output of the \( i \)-th output unit is computed by:

\[
a_i = \sigma(a_1 w^o_{1i} + a_2 w^o_{2i} + a_3 w^o_{3i} + a_4 w^o_{4i} + b^o_i)
\]

This procedure can be written in matrix notation by defining:

- \( (W_h)_{ij} = w^h_{ij}, i = 1 \ldots 3, j = 1 \ldots 4 \)
- \( (W_o)_{ij} = w^o_{ij}, i = 1 \ldots 4, j = 1 \ldots 2 \)

Then the output vector \( o \) can be computed as:

\[
a = \sigma(W_h^T x + b^h)
\]
\[
o = \sigma(W_o^T a + b^o)
\]

So, the neural network can be described by the following function:
State of the art in optimization and machine learning algorithms applied to last mile logistic

\[ f(x) = \sigma(W_o^T \sigma(W_h^T x + b^h) + b^o) \]

To make a prediction with a neural net one must perform the above described computation, this procedure is also called forwards-pass. So far it was always assumed that the weights were given, but one of the main challenges of designing a NN is to determine these weights in a way that the neural network makes accurate predictions. First one must decide what “accurate” means i.e. choose a function (Loss function) which evaluates the quality of a prediction. So, the optimization object of the above NN can be written as follows, where \( L \) is an arbitrary Loss function:

\[
\min_{W_\sigma \in \mathbb{R}^{3x4}, W_h \in \mathbb{R}^{4x2}} \sum_{i=1}^{n} L(f(x_i), y_i)
\]

For regression tasks one often uses the objective of OLLSR and for classification Cross-Entropy-Loss. The values of the weights are determined by a technique called backpropagation and a flavour of gradient descent which is explained in more detail in [12].

3.7.2 Long Short-term Memory (LSTM)

Recurrent neural networks with Long Short-Term Memory (LSTMs) have emerged as an effective and scalable model for several learning problems related to sequential data. Earlier methods for attacking these problems have either been tailored towards a specific problem or did not scale to long time dependencies. LSTMs on the other hand are both general and effective at capturing long-term temporal dependencies [3]. They do not suffer from the optimization hurdles that plague simple recurrent networks and have been used to advance the state-of-the-art for many difficult problems. This includes handwriting recognition and generation language modelling and translation, acoustic modelling of speech, speech synthesis protein secondary structure prediction analysis of audio and video data among others.

The central idea behind the LSTM architecture is a memory cell which can maintain its state over time, and non-linear gating units which regulate the information flow into and out of the cell.

A schematic of the vanilla LSTM block can be seen in Figure (c.f. Figure 14). It features three gates (input, forget, output), block input, a single cell (the Constant Error Carousel), an output activation function, and peephole connections. The output of the block is recurrently connected back to the block input and all the gates.
3.7.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) are analogous to traditional NNs (Neural Network) in that they are comprised of neurons that self-optimize through learning [2]. Each neuron will still receive an input and perform an operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs.

CNNs are needed because, one of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data. Common machine learning benchmarking datasets such as the MNIST database of handwritten digits are suitable for most forms of ANN, due to its relatively small image dimensionality of just 28×28. With this dataset a single neuron in the first hidden layer will contain 784 weights (28×28×1 ,1 since MNIST is normalized to just black and white values), which is manageable for most forms of ANN. If one considers a more substantial coloured input image of 64×64x3 (RGB-channel), the number of weights on just a single neuron of the first layer increases substantially to 12.288.

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers, and fully connected layers. When these layers are stacked, a CNN architecture has been formed. A simplified CNN architecture for MNIST classification is illustrated below.
The basic functionality of the example CNN above can be broken down into four key areas.

1. As found in other forms of ANN, the input layer will hold the pixel values of the image.
2. The convolutional layer will determine the output of neurons which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume.
3. The pooling layer will then simply perform down-sampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation.
4. The fully connected layers will then perform the same duties found in standard ANNs and attempt to produce class scores from the activations, to be used for classification. It is also suggested that ReLu may be used between these layers, as to improve performance.

In the figure below one can see the activations taken from the first convolutional layer of a simplistic deep CNN, after training on the MNIST database of handwritten digits. If one looks carefully, one can see that the network has successfully picked up on characteristics unique to specific numeric digits.
As the name implies, the convolutional layer plays a vital role in how CNNs operate. The layers parameters focus around the use of learnable kernels. These kernels are usually small in spatial dimensionality, but spreads along the entirety of the depth of the input. When the data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a 2D activation map. The following plot illustrates how the kernel moves over an input of size 7x7 with stride 1 and produces an output of size 5x5.

Convolutional layers are also able to significantly reduce the complexity of the model through the optimization of its output. These are optimized through three hyper-parameters, the depth, the stride and setting zero-padding. The depth of the output volume produced by the convolutional layers can be manually set through the number of neurons within the layer to the same region of the input. This can be seen with other forms of ANNs, where all the neurons in the hidden layer are directly connected to every single neuron beforehand. Reducing this hyper-parameter can significantly minimize the total number of neurons of the network, but it can also significantly reduce the pattern recognition capabilities of the model. One is also able to define the stride in which we set the depth around the spatial dimensionality of the input to place the receptive field. For example, if the stride were set as 1, then the receptive field will be heavily overlapping, as consequence the resulting activation map is
large. Alternatively, setting the stride to a greater number will reduce the amount of overlapping and produce an output of lower spatial dimensions. Zero-padding is the simple process of padding the border of the input and is an effective method to give further control as to the dimensionality of the output volumes.

Pooling layers aim to gradually reduce the dimensionality of the representation, and thus further reduce the number of parameters and the computational complexity of the model. The pooling layer operates over each activation map in the input, and scales its dimensionality using the “MAX” function. In most CNNs, these come in the form of max-pooling layers with kernels of a dimensionality of 2×2 applied with a stride of 2 along the spatial dimensions of the input. This scales the activation map down to 25% of the original size - whilst maintaining the depth volume to its standard size. Due to the destructive nature of information loss of the pooling layer, there are only two generally observed methods of max-pooling. Usually, the stride and filters of the pooling layers are both set to 2×2, which will allow the layer to extend through the entirety of the spatial dimensionality of the input. Furthermore, overlapping pooling may be utilized, where the stride is set to 2 with a kernel size set to 3. Due to the destructive nature of pooling a kernel size above 3 will usually decrease the performance of the model to a larger extent. It is also important to understand that beyond max-pooling, CNN architectures may contain general-pooling. General pooling layers are comprised of pooling neurons that can perform a multitude of common operations including L1/L2-normalisation, and average pooling.

The fully-connected layer contains neurons which are directly connected to the neurons in the two adjacent layers. This is analogous to way that neurons are arranged in traditional forms of feed-forward ANN.

3.7.4 Recurrent Neural Networks (RNN)

Recurrent neural networks (RNNs) are a family of neural networks for processing sequential, temporal data. Much as a convolutional network is a neural network that is specialized for processing a grid of values X such as an image, a recurrent neural network is specialized on processing a temporal sequence of values x_1, ..., x_n like in handwriting or speech recognition. As convolutional networks can readily scale to images with large width and height – and some convolutional networks can process images of variable size – recurrent networks can scale to much longer sequences than would be impractical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length. A simple RNN has three layers which are input [161], recurrent hidden and output layers, as presented in Figure below.

![Figure 18 A Simple RNN with three layers [161]](image)
The input layer has N input units. The inputs to this layer is a sequence of vectors through time $t$ such as $\{... , x_{t-1}, x_t, x_{t+1}, ...\}$, where $x_t = (x_1, x_2, ..., x_N)$. The input units in a fully connected RNN are connected to the hidden units in the hidden layer, where the connections are defined with a weight matrix $W_{IH}$. The hidden layer has $M$ hidden units $h_t = (h_1, h_2, ..., h_M)$, that are connected to each other through time with recurrent connections (c.f. Figure below).

![A Simple RNN with recurrent connections](image-url)

The initialization of hidden units using small non-zero elements can improve overall performance and stability of the network [9]. The hidden layer defines the state space or “memory” of the system as

$$h_t = f_H(o_t)$$  \hspace{1cm} (1)

where

$$o_t = W_{IH}x_t + W_{HH}h_{t-1} + b_h,$$  \hspace{1cm} (2)

$f_H$ is the hidden layer activation function, and $b_h$ is the bias vector of the hidden units. The hidden units are connected to the output layer with weighted connections $W_{HO}$. The output layer has $P$ units $y_t = (y_1, y_2, ..., y_P)$ that are computed as

$$y_t = f_O(W_{HO}h_t + b_o).$$ \hspace{1cm} (3)

where $f_O$ is the activation functions and $b_o$ is the bias vector in the output layer. Since the input target pairs are sequential through time, the above steps are repeated consequently over time $t = (1 ... T)$. Equation (1) and (3) show a RNN is consisted of certain non-linear state equations, which can be iterated through time. In each time step, the hidden states provide a prediction at the output layer based on the input vector. The hidden state of a RNN is a set of values, which apart from the effect of any external factors, summarizes all the unique necessary information about the past states of the network over many time steps. This integrated information can define future behaviour of the network and make accurate predictions at the output layer [5]. A RNN uses a simple nonlinear activation function in every unit. However, such simple structure is capable of modelling rich dynamics if it is well trained through time steps.
3.7.5 Deep Learning

Deep learning is an AI function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of ML that has networks capable of learning unsupervised from data that is unstructured or unlabelled. It is also known as deep neural learning or deep neural network, as it trains large neural networks. While DL was implied within the explanation of neural networks, it’s worth noting more explicitly. The “deep” in deep learning is referring to the depth of layers in a neural network. A neural network that consists of more than three layers- which would be inclusive of the inputs and the output- can be considered a deep learning algorithm [23]. Because of the close links, artificial neural networks and deep learning are often used interchangeably, which isn’t really correct. Not all neural networks are deep, meaning “with many hidden layers”, and not all deep learning architectures are neural networks [24].

3.8 Best Practices from Industrial Use Cases

3.8.1 Big Data Pipeline for AI

Data is one of the most fundamental parts of data analytics. The data pipeline processing sketched below follows best practice implementing Big Data Pipeline for data processing. This pipeline follows the following figure and is illustrated in the following

![Figure 20: Big Data Pipeline for Data Processing in SENATOR](image)

3.8.1.1 Data Sources

Starting with the data sources that feed the event streams, data events are shared using a common message bus. Components that either produce or consume data, subscribe to the respective topics to hereby form a first segregation of data types: in SENATOR data is grouped by data formats assigned to specific data topics.

3.8.1.2 Raw Data Storage

For feeding the data pipeline we use a simple event sniffer subscribed to all topics which can collect JSON events of any data format which is consequently persisted on disk and periodically uploaded into a database (event store). Hereby the data format is indifferent: As the event payload contains JSON distinct event fields can later be selected using a set of type-specific views on the payload fields.
3.8.1.3 Processing Data:
Pre-processing plays a huge role in Machine Learning, since every learning algorithm strongly depends on suitable data. There are different reasons why pre-processing may be beneficial, for example in practice one is often faced with data which is in a bad condition i.e. missing values or even wrong values. Another example is that some methods require the data to have unit variance or zero mean.

3.8.1.4 Handling Missing Values
Usually there are two different ways of dealing with missing values, either impute the missing values or delete the data points with missing values or even delete a complete feature. An advantage of imputation is that every data point is used. This is particularly interesting if one has a small dataset. On the other hand, if a lot of values of a certain feature are missing, imputing these values may bias the model. Another disadvantage for both methods is that, if the structure of the missing values contains predictive information, this information will be destroyed by deletion or imputation.

3.8.1.5 Detecting and Dealing with Outliers
Many Machine Learning algorithms are strongly influenced by outliers for example every algorithm that uses sample mean to build a predictor. Therefore, it is crucial for the performance of the model to detect these outliers and explain the origin of them. One simple method which yields reasonable results is the usage of boxplots (method to graphically depict groups of numerical data through their quartiles).

3.8.1.6 Scaling and Translation of Features
In practice one often encounters datasets with features of different scales; this is a problem for algorithms which use a similarity measure to produce a predictor. For example, if an algorithm uses Euclidean distance to determine similarity between points, features with large scale will have more influence than features with a low scale. This can result in a suboptimal choice of a classifier since the low scale of a feature does not imply that the feature has low predictive power. In practice the following two approaches are often used:

- **Standardization using Z-scores:** $x'=(x-\mu)/\sigma$, where $\mu$ is the mean and $\sigma$ is the standard deviation of the feature $x$, the new feature will have zero mean and standard deviation one.
- **Min-Max-Scaling:** $x'=(x-min(x))/(max(x)-min(x))$ all the values of the new feature are between zero and ones

3.8.1.7 Principal Component Analysis (PCA)
PCA is the typical example for a dimensionality reduction technique, which means that number of features per data point is reduced. One reason why this is useful is the fact that finding patterns in data of high dimensions requires exponentially more observations than finding patterns in lower dimensions. As consequence one will probably find a better model using a lower dimensional feature space, even though discarding features will destroy information.

3.8.1.8 Analysis Environment
In the area of Machine Learning there exist a tremendous number of different tools and models. Due to its popularity and its open-source character Jupyter Notebook is used throughout the process of data analysis as it is free and has a strong open-source developer community if facilitated. However,
in general there is no “the best model” as every model has its strengths and weaknesses. Consequently, choosing the right model highly depends on the use case. In the process of choosing a model, the principle of Occam’s Razor applies, which says that simpler solutions are more likely to be correct than complex ones. This means in terms of model selection, if two models perform equally in the evaluation, the less complex is the better one.

3.8.1.9 Data Serving

Zementis Predictive Analytics for Artificial Intelligence (AI) and machine learning dramatically reduces the complexity, expense and time required for enterprises to deploy, execute, and act on sophisticated predictive models. Zementis delivers massive improvements in the speed at which intelligent decisions are integrated into business processes and operational IT systems.

3.8.2 Lambda Architecture

Modern manufacturing- and shop-floor systems are equipped with sensors that constantly monitor production relevant KPIs by collecting huge amounts of data. Through the increase of performance of today’s computers and sensor devices over the past decade, this trend has been even increased as technical enablers become cheap, wireless communication reliable and embedded devices less resource hungry. Mainly driven through the advent of IoT this trend has even been increased, supporting technical solutions for real-time monitoring, event persistence in Big Data pools and smart algorithms which allow the digestion of large data collections the generation of business-critical insights. Especially KPI Monitoring, supply chain analytics, or production monitoring are to name in this context which are relatively well-known application areas of this technology.

However, the application of real-time monitoring technology paired with Artificial Intelligence in Human-Centred Manufacturing – as in the context of the SENATOR project – is a relatively new domain. This is the particular focus of the SENATOR project by developing a person-centred smart solution based on the above-mentioned technologies to support the employment and later retirement of older adults within their work fields.

Event messaging and complex event processing (CEP) are two of the main drivers that allow the detection of situations as they occur by monitoring multiple event streams in parallel. Data Analytics further allows predictions by looking into the past and hypothesizing “What might happen next?”. The concept of integrating Messaging, CEP and Data Analytics is very successful and motivates modern IT architectures using the concept of a Lambda-Architecture [6]. This architectural approach suggests the separation into two core layers such as Speed Analytics Layer and Batch Analytics Layer, which are facilitated by the Messaging Channel (see Figure below).
The difference between the Batch Analytics Layer and the Speed Analytics Layer is tracking capability, prediction accuracy and speed. On the Batch Analytics Layer, predictive models are built based on historic data using, for example, in-depth machine learning algorithms and neuronal networks. As this is a decoupled and off-line process, it is not executed in real-time but requires deep analysis of the collected data by data scientists and experts from the field. The data flow is following a stringent path by collecting raw data, data cleansing, data replenishment using additional sources of information, data aggregation, and eventually the application of machine learning algorithms which deliver the model behaviour in terms of Predictive Model Mark-up Language (PMML) models, a de facto standard. PMML is used as an input for SENATOR’s prediction engine ZEMENTIS [7] which executes the trained models at run-time on inputs received via the Messaging Channel to enable predictive insights and forecasts. Especially the model creation is hereby a tedious process which requires expertise of data scientists that explore and pre-process the data to eventually create models that reflect a high accuracy and precision rate, i.e. the error in forecasting is low (c.f., Figure below).
3.8.3 AI Model Management

While predictive analytics uses a model-based forecast based on historic data, the Speed Analytics Layer provides a real-time view on events. The module development is based on APAMA [8] which is watching real-time data streams for specific pattern to correlate, aggregate and monitor figures to higher level events. It is a rule-based (deterministic) approach where queries are defined using Event Processing Language (EPL) and they may be executed across multiple streams of events in real-time. All tools from above manipulate data originating from sources, such as the SENATOR Bridge which directly interlinks with the event message bus. Moreover, as events are generated from the Speed and Batch Analytics Layer tools, these are forwarded through the Messaging Bus to be further processed within SENATOR platform by different SENATOR service modules.

3.9 Review of most recent applications domains

Nowadays, delivery traffic is steadily increasing because of ongoing urbanization and changes in consumer behaviour (such as e-groceries, e-commerce, etc.). The reasons for this include the strong growth in online retailing, customers' increasing quality expectations, and the advancing digitalization in all areas of life. Statistics show that in 2020, for all individuals aged 16-74 in the EU, the share of online shoppers equalled 64% [49]. Due to this, demographic trends point to an increasing concentration of last mile deliveries in urban areas. This is especially true in Europe, where three quarters of the population live in cities. The pandemic emergency linked to COVID-19 has exacerbated this problem due to government restrictions that have led to the possibility for some retailers of exclusive home delivery to sell their goods. The COVID-19 pandemic, indeed, has been a “rare catalyst”
for logistic innovations [45]: on the one hand, the new technologies have proved to be effective in addressing the challenges generated by the emergency situation; on the other hand, companies had to manage new risks with resilience measure; an emblematic situation is that of scheduled air transport, for which many companies have repurposed seating areas for cargo transport to offset the rapid decline in passengers. Notwithstanding the recent efforts done to improve the sustainability of logistics, issues related to the whole process are still debated, in particular in cities: while urban logistics vehicles account for up to 15% of traffic only, they contribute significantly to urban air polluting emissions (25% of CO2, and 30 to 50% of PM and NOx) and are disproportionally involved in fatal collisions [61].

Innovative digital technologies can come into help when dealing with urban deliveries. While individuals are familiar with online purchasing processes, the intelligent control of the whole supply chain is a perfect field for exploiting the potential of new technologies. In this respect, in recent years, logistic supply chain management based on soft computing has become an ongoing hot topic: Artificial Intelligence, and in particular Machine Learning (ML), have gained increasing relevance, mainly concerning on how to integrate and allocate resources efficiently to reduce cost and improve efficiency (e.g. improving demand forecasting accuracy and reducing the time and money spent on tracking the goods). Moreover, ML-based algorithms have the potential to automatically inspect the logistic processes to detect anomalies [34]. A scheme of the domains of application of ML to logistics is reported in figure 22. Data related to production and warehouse logistics and those relating to transport can be analysed through ML algorithms and the anomalies are detected. Then, ML algorithms can be used to improve the processes, as detailed in the following sections.

![Information Flow for selected SENAATOR data sources](image)

Based on these premises, in the following, an analysis of most recent application of Machine Learning models in logistics will be presented, by clustering them in three domains: smart logistics, urban logistics and anomaly detection.

### 3.9.1 Smart Logistics

#### 3.9.1.1 SMART Logistics – definition and related concepts

Smart logistics is a recent concept which embraces Smart Service as well as Smart Products within Logistics; it is therefore based on a technology driven approach enabling traditional techniques (e.g., material handling systems, information, billing) to act in a smart manner [52], [87]. Smart logistics are
often referred to as “Intelligent Logistics”, or “Logistics 4.0”, deriving from the concept of “Industry 4.0”, which is based on ubiquitous combination of machines, products, systems and people, which enable them to communicate, and form a mutual management network [90]. The term “e-logistics” is also used when the Internet has a key role in logistics processes and supply chain management [89]. The Internet, indeed, is playing a fundamental position in the evolution of logistics: on the one hand through the Internet of Things (IoT), which allows an easy processing of data and transferring information to the devices; on the other hand, through the birth of the “digital client” (or “connected customer”), which causes higher pressure on suppliers, willing for a better adaptation of products to customers’ needs and preferences and quick and trouble-free deliveries [44]. The supplier, therefore, in order to satisfy customers’ demand and maintain a competitive advantage over the competitors, must use intelligent technologies in its favour, using data analysis to support advanced decision making and increasing the functionality and efficiency of logistics processes through the interaction between people and machines.

3.9.1.2 Smart logistics domains and application of ML to use cases

The use of intelligent technologies in logistics can be clustered in four different domains [44]: (1) Transport, including security systems, route planning, navigation systems; (2) Warehouse, including RFID, intelligent forklift, smart racking, picking automation; (3) Production, including quality control systems, intelligent assembly systems, production control; (4) Supply chain, including e-commerce, virtual supply network and the global process.

Comprehensive reviews of the application domains of ML in Smart logistics can be found in the recent works of [93][92][41]. The systematic reviews show that several applications can be found for the production cluster (e.g. see [86][76]); however, in this report we will neglect a deep analysis of the applications related to this cluster since it is extraneous to the topics of the SENATOR project, which basically deals with urban deliveries.

The cluster in which ML has seen several recent applications is certainly the one related to transport; in particular, in the case of freight transport, the route optimization is certainly among the most investigated topic. Several scholars address the VRP through Machine Learning techniques, and in particular through Neuro-Dynamic Programming (NDP), a technique which uses neural network and other approximation architecture to overcome the limits of DP due to the huge size of the underlying state space (the so-called “curse of dimensionality”). The methodology allows systems to learn about their behaviour through simulation, and to improve their performance through iterative reinforcement [60][70][73][81][88]. Combinatory approaches of ML techniques with optimization algorithms are also frequent [42][58][68][62]. A deeper analysis of VRP optimization techniques is provided further on.

Other works deals with trips forecasting, such as travel time and traffic flow prediction. In this respect[82] apply the ML algorithms Extremely Randomized Trees (ExtraTrees), Adaptive Boosting (AdaBoost), and Support Vector Regression (SVR) to build several models for travel time prediction, by using tracking data from a real-world multimodal container transport relation from Germany to the USA. Results from this study show that SVR provides the best prediction accuracy. [91] applies the Inferential Theory of Learning in multiagent-based simulation environment in the case of autonomous logistics.
Two use cases are presented:
1) creation of models to predict future traffic flows, and
2) learning in evolutionary transport planning; in both cases agents with learning abilities outperform inexperienced agents in performing their tasks.

This type of self-learning system is common in ML applications within the Warehouse cluster. ML algorithms are used to read handwritten documents (e.g., labels on letters or packages) and to detect frequent events which can then be incorporated in specific rules. A literature application that can be useful for these purposes is the one by [58], who propose an entity-matching approach and system for validating transportation and logistics entities, based on Word Embedding and Supervised Learning techniques; in particular, they verify the entities existence through their addresses in France, by matching company names and addresses, using several data repositories. Results show high accuracy of their algorithms (0.9 evaluated through the F1 score). In another recent work, [40] propose a fully automated process control solution for container logistics, based on deep neural networks and trained from process steering decisions made by employees. More in detail, a container detection framework based on supervised machine learning is used for the steering of empty container processes in the internal supply chain of an engine manufacturing plant. A fully automated labelling framework allows operators at the plant to integrate new container types by teaching the machine learning model with automatically labelled images from the observed container routing workflow.

Several recent applications of ML algorithms apply to the entire logistic supply chain. One of the most spreading is that of anticipatory logistics, i.e., the forecasting of demand trends. Through anticipatory logistics, indeed: (i) manufacturers can predict their production levels; (ii) transport operators can plan the appropriate vehicle capacity; (iii) retailers can order and store appropriate stocks and plan their personnel schedule. Moreover, anticipatory logistics help to detect risks at an early stage in the whole process, including the recent pandemic emergency effects. Several authors propose theoretical approaches for predicting future inbound logistics processes using ML techniques [64][91]. In a recent application [39] explore the use of K-nearest neighbours (KNN – see Section 3.6.1.3), Random Forests and Support Vector Machine (SVM, see Section 3.4.2.3) machine learning algorithms to support inbound logistics planning; however, they do not explore which model would result in a more successful performance in different use cases. An interesting recent suggestion is the one provided by [67], which explore the use of Machine Learning Algorithms for Classification Tasks in Reverse Logistics. More in detail, the paper is a preliminary study in which a set of criteria to compare supervised learning algorithms was introduced. The study shows that, for all criteria, requirements could be identified regarding the exemplary case and the data available.

Finally, [41] after analysing the state of the art in optimization and machine learning in logistics planning and control, propose a vision for 2030. According to the authors, the main challenges are: (i) the scalability of the algorithms, since huge size of the problems will challenge the classical notion of efficient algorithms; (ii) the need of “Model-and-Run Features”, since last-mile logistics amounts to dispatch requests in real-time as well as to schedule delivery staff and vehicles; the ultimate goal will be to automatically generate metaheuristics that may provide “human competitive results”, i.e., results comparable to (or even better than) those produced by tailored algorithms entirely designed by humans; (iii) the improvement of Learning Capabilities: the authors imagine a future class of algorithms that will be able to recognize the type of instance to be solved.
3.9.2 Urban Logistics

3.9.2.1 Definition and existing regulatory framework

Urban Logistics, also known as “city logistics” or “last-mile delivery”, is “the means over which freight distribution can occur in urban areas and the strategies that can improve its overall efficiency while mitigating externalities such as congestion and emissions” [72]. The need to analyse city logistics separately from the more general supply chain of smart logistics is linked precisely to the greater impacts in the urban environment (and therefore on the communities), essentially generated by the retailers supplying on a just-in-time basis, the demand for express transport and the development of innovative courier services. These assumptions emphasize more the link between city logistics, transport and land use in urban areas. The coexistence of the phenomenon of city logistics with all the other phenomena that occur within the urban space generates various problems, both for freight transport operators and for the public. On the one hand, indeed, operators may face problems related to delays in deliveries (and consequently customers unsatisfaction) due to traffic congestion and access restrictions for heavy vehicles in central areas of the cities. On the other hand, the community sees an increase in pollution (with higher noise and air emissions) and of the congestion and occupation of the public spaces by the vehicles intended for the delivery of the goods.

The European Commission started to deal with urban logistics issues since the 2011 White Paper, setting up a strategy for CO2-free city logistics in major urban centres by 2030, through electrification, low-carbon fuels, shift to rail and water, and modern IT systems and infrastructure [46]. Then, in the 2013, the EC developed the Urban Mobility Package (UMP), with Urban Logistics as one of the four key topics. The UMP Communication, ‘Together towards competitive and resource-efficient urban mobility’, set out that Member States should consider logistics in their urban mobility approaches, particularly in SUMPs, and establish platforms for co-operations, data exchange, and training for all actors. The Communication was accompanied by the Staff Working Document, ‘A call to action on urban logistics’, which pointed out several of the key challenges of urban logistics [47], and among them:

- A lack of focus and strategy on urban logistics, and only a few cities with someone in authority responsible for urban logistics.
- A lack of coordination among actors involved in urban logistics, and in many cases insufficient dialogue between city authorities and the private actors who operate there.
- A lack of data and information, which makes it difficult to improve operational efficiency and long-term planning. The topic of informatization was further stressed in the 2016, in a new EC publication called ‘A European Strategy for Low Emissions Mobility’ [48], which highlighted the importance of digital mobility solutions for integrated logistics.

The SENATOR project aims to close all these gaps by developing an urban logistic technologic platform focused on four urban layers, including the end-receiver (citizens & freight receivers), transport (modes for delivery accomplishment), logistics operators (operational logistics) and infrastructure (policy makers and urban infrastructure owners). In this respect, ML algorithms can come in handy on several fronts: (i) they can help to understand peoples’ needs and, therefore, guarantee customers’ satisfaction; (ii) they can estimate future demand to ensure better planning of operations; (iii) they can improve the global last-mile delivery process, by aiding technological innovations. Based on these premises, in the following section an analysis of main challenges for urban logistics and recent ML-based solutions will be presented.
3.9.2.2 Main challenges and solutions and use case applications

Yet in 1990, Sam Walton, Founder of Walmart, stated the famous quote “There is only one boss. The customer. And he can fire everybody in the company from the chairman on down, simply by spending his money somewhere else.”. Customer satisfaction is not a novel issue in purchasing sector; however novel technologies, such as ML algorithms, can help in pursuing it in urban logistics. Example of how this can be achieved can be found in recent literature. In their study [84] tested the general feeling of people towards urban logistics. They show how the analysis of social media content could help in understanding the public perception of City Logistics collecting content from Twitter and implementing machine learning techniques (Unsupervised Learning and Natural Language Processing), to perform content and sentiment analysis. The proposed methodology is applied to more than 110 000 tweets containing City Logistics key-terms. Results allowed the building of an Interest Map of concepts and a Sentiment Analysis which showed that that the overall view of City Logistics is more positive than negative. With more detail on customer satisfaction, [85] propose a blockchain-based evaluation approach in the context of urban logistics. Four criteria affecting customer satisfaction in urban logistics are identified: Cargo damages rate; On-time delivery rate; Cost performance; Information transparency. A machine learning algorithm Long Short-Term Memory (LSTM, c.f. Section 3.7.2) is adopted to predict customer satisfaction in the future period with efficient results. Moreover, the authors design a smart contract for compensation and/or refund to customers when their satisfaction with the delivery services is at a low level.

ML techniques find applications also in one of the most challenging issues in urban freight delivery problems, i.e. demand forecasting, which is essential for delivery companies to schedule their operations, especially in these times of unpredictable changes. Some studies relating to this topic have already been dealt with in the previous section and are valid also in the case of urban logistics. In particular, some dedicated studies are the ones conducted by [53] and [59]. The first authors yet in 2009 provided a modelling and forecasting method of urban logistics demand based on SVM (Support Vector Machine – see Section 3.4.2.3) regression; more in detail, the authors propose a self-adaptive parameter adjust iterative algorithm to confirm SVM parameters, thereby enhancing the convergence rate and the forecasting accuracy through ability of self-learning of SVM. The application of the model to an actual forecasting case study show that the method is feasible. Recently [59] propose an approach incorporating both classical forecasting and machine learning methods and adapt model evaluation and selection to typical demand: intermittent with a double-seasonal pattern. The application to an empirical study (i.e. a meal delivery platform) shows that an exponential smoothing based method trained on past demand data achieves optimal accuracy, if at least two months are on record. Results shows that with a more limited demand history, machine learning can get to more accurate prediction results than traditional methods. A real-life case of using ML for the improvement of food delivery is the one by a famous worldwide food delivery company ([75]); they developed a dispatch engine, called ‘Frank’, which is constantly calculating and matching the best combination of riders with respective customer orders. These calculations and predictions are based on a ML algorithm trained on historical data to predict rider time, food preparation time and all the other variables related to meal delivery.

Another traditional transport-related issue is the one of safety. With a focus on urban logistics, [94] summarizes in their work the risk factors to public security in the process, including pick up, warehouse storage, transport, and the end distribution. They use generalized regression neural network (GRNN) combined with particle swarm optimization (PSO) to predict accidents, and the a priori algorithm to
analyse the combination of high-frequency risk factors. The results show that the combining GRNN with PSO is effective in accident prediction, improving the ability of relevant departments to deal with emergency incidents, and minimizing the impact of urban logistics accidents on social and public safety.

The increase in the e-commerce phenomenon is also leading to a growing demand for urban logistics spaces to serve last mile deliveries and available sites for urban logistics. A solution is the locating of consolidation centres on the edge of cities which might help operators to guarantee the responsiveness to delivery (due to proximity to cities) while keeping heavy vehicle traffic (and the associated externalities) outside the urban area. In this respect, [50] developed a ML-based for zoning urban area in consolidation schemes context. They performed accuracy benchmarks of machine-learning algorithms for an efficient demand investigation tool to decrease logistics demand uncertainty that involves extra costs due to the system oversizing, considering how proximity and logistics demand behaviour affect such dimensioning problem. An application to experimental data shows the efficiency of using machine learning algorithms to this end.

As stated previously, the urban supply chain must face different challenges than the traditional one, with a particular focus on environmental pollution. ML algorithms can aid operation planners by developing comprehensive and sustainable solutions. In this [65] propose a ML based range prediction model which facilitates full exploitation of an EV fleet in urban logistics, including routing, traffic and weather data. These methods can reproduce consumption levels of learned real data sets with a relative error ranging below the 10%, predicting the energy loss of an electric vehicle in a reliable way. The requirement of creative delivery solutions in urban areas which could take advantage of ML is also related to the innovative Digital Twin (DT) concept, as discussed in [71]. A joint use of behavioural and simulation models should characterise a DT within a Living Lab approach so to stimulate effective, well-informed and participated planning processes, but also to forecast both behaviour and reactions to structural changes and policy measures implementations. More in detail, the paper highlights the use of ML within a DT platform for the described iterative learning processes.

3.9.3 Anomaly Detection

3.9.3.1 Introduction to anomaly detection

Anomaly detection (AD), or “outlier detection” is referred to as the process of detecting data occurrences that significantly deviate from most data instances. Typically, these anomalous items have the potential of getting translated into problems such as structural defects, errors, or frauds, affecting the efficiency of any technological logistic platform. Thus, AD can, for example, enhance the communication around the system behaviour or reduce threats to the software ecosystem. AD is often applied on unlabelled data (see [66]) which is known as unsupervised anomaly detection. It has two basic assumptions: (i) anomalies only occur very rarely in the data and their features differ from the normal instances significantly; (ii) outliers can come in different types, depending on the environment: point outliers, contextual outliers, or collective outliers (see [43]). Point outliers are single data points that lay far from the rest of the distribution. Contextual outliers can be noise in data, such as punctuation symbols when realizing text analysis or background noise signal when doing speech recognition. Collective outliers can be subsets of novelties in data such as a signal that may indicate the discovery of new phenomena. Causes of outliers on a data set can be for example human made
data entry errors, measurement errors, experimental errors (data extraction or experiment planning/executing errors) or intentional errors to test the detection methods [79].

In the current era of logistics 4.0 the use of high-quality data, in particular in the Intelligent Transportation Systems (ITS) is a fundamental prerequisite [74][63]; AD and related ML-based techniques can play a fundamental role in detecting anomaly events on the big stream of electronic data obtained from the ubiquitous processes. In this respect, SENATOR project aims at applying ML techniques to perform advanced data analysis and detect novel/unusual events in the logistic processes developed in the platform. Based on this premise, in the following section we will briefly introduce anomaly detection techniques and analyse some recent applications in the freight delivery sector.

3.9.3.2 Anomaly detection techniques

The scenario analysis for AD can be more or less complex. With reference to the number of variables to be examined, we distinguish between univariate anomaly detection and multivariate anomaly detection. In the first case, the outliers emerge by observing the distribution of the values relating to a single parameter. Univariate methods are easy to scale to many metrics and large datasets; however, relationships between the anomalies must then be established. In the case of multivariate detection, the outlier emerges from the observation of at least two variables, combined with each other; this means that outliers can be found in a n-dimensional space.

A detailed definition of AD and a review of the main techniques used can be found in [95]. According to the author, AD approaches for univariate and multivariate problems can be roughly classified along the following different axes:

- Global vs. Local Outliers. While some AD methods take the complete database into account, others consider only a local selection of database objects. Examples of this last type are the so-called density-based techniques (e.g. k-nearest neighbour, local outlier factor, isolation forests);
- Labelling vs. Scoring Methods. Labelling techniques lead to a binary decision of whether or not a given object is an outlier whereas the scoring methods are rather assigning a degree of “outlierness” to each object;
- Supervised vs. Unsupervised Methods. A supervised approach is based on a set of observations where the status of being an outlier or not is known and the differences between those different types of observations are learned. Labelling approaches are generally supervised methods.
- Parametric vs. Non-parametric Methods. Parametric approaches assume a particular family of distributions to describe the (normal) data and fit the presupposed model to the data by learning the parameters of the model.

3.9.3.3 Anomaly Detection in freight delivery and logistics

Several general applications of AD in smart logistics can be found in recent literature. An example is the one developed by [80], which propose to integrate a DL-based AD as a service into the 3GPP mobile cellular IoT architecture. The proposed architecture embeds autoencoder based AD modules both at the IoT devices and in the mobile core network, balancing between the system responsiveness and accuracy. They custom-designed a novel NB-IoT device platform for a Smart Logistics use case, where NB-IoT devices are connected to shipping containers in a factory supply chain, to collect data, deploy
and test the modules. Results of the application, based on a small-scale real-world trial, emphasize that autoencoders represent a suitable choice for ML anomaly detection.

An intuitive application of AD in freight delivery field is the one related to the detection of anomalies in vehicles movements. In this respect, [77] developed an algorithm for anomaly detection in trajectories, handling spatial and temporal data shifts and dealing with trajectories of unequal lengths, with the main aim to extract a mean path that is “normal” for the monitored route and detect the anomalies. The algorithm, tested on a real data set containing trajectories of freight ships traveling through the English Channel, proved to be effective. Another study is the one conducted by [51], which deals with the incorporation of three ML algorithms, namely Bayesian belief network, decision tree and random forest to detect activity types from GPS traces. In a more recent work, [78] use image processing and machine learning techniques to detect vehicular flow directions on a highway, using adapted k-nearest neighbour algorithm for unsupervised learning and proving the method to be reliable in the case of a single lane analysis.

Finally, AD plays an important role also when it comes to technological advances addressing last mile transport which are still in the conception phase, such as the use of drones in urban deliveries. An explanatory example is the one provided by [83], which propose a novel anomaly detection framework for a fleet of hybrid aerial vehicles performing fast package pickup and delivery missions. The detection is based on machine learning models of normal flight profiles, trained on millions of flight log measurements of control inputs and sensor readings. The proposed algorithm for robust regression can simultaneously fit predictive flight dynamics models while identifying and discarding abnormal flight missions from the training set. The resulting unsupervised estimator has a high breakdown point and can withstand massive contamination of training data to uncover what normal flight patterns look like, without requiring any form of prior knowledge of aircraft aerodynamics or manual labelling of anomalies upfront. Results show that the method outperforms alternative robust detection methods on synthetic benchmark problems.

### 3.10 Benchmarks and Datasets

Today a vast number of different AI models exists which are continuously improved and enhanced through their authors. These algorithms are very dominant in areas like image recognition, image analysis, flow detection, language processing and even speech recognition. However, their use in the domain of AI is to the current day very scarce and companies in possession of such algorithms only rarely share them.

#### 3.10.1 Benchmarks in AI

A growing number of technologies and methods contributed to the evolution of AI over the past decade. Three factors mainly influence the advance of AI which are algorithms, data and the computation available for the model training [163]. The trend currently shows roughly an increase of factor 10 for each year that was mainly driven by the use of custom hardware (GPUs, TPUs) and the research on parallelism. The following figure depicts the increase of compute power for training AI Systems since 1960 starting with the perceptron.
However not only computation power is relevant. However, in industrial applications more factors impact the benchmarks of machine learning models. Results are quite often produced in laboratory-type settings, under ideal conditions and machine workloads to achieve potentially the highest possible performance score but which may differ from real-scenarios in set-ups with shared resources.

Eventually is not only the software or the hardware but a combination of libraries, systems, formats, etc. The following figure gives an overview of factors that influence the performance of machine learning models and their benchmarks. These factors range from the ML Application, used framework, up to the target hardware.
Making SENATOR results comparable with results from other ML projects is a difficult if not infeasible task as the likely all of the above factors will be different. However, available datasets from image processing are available and could be used to roughly estimate KPI related factors, like Error rate, Accuracy, Flow Outliers, etc. Although they will eventually be difficult to be compared with results from SENATOR, as they have been developed against different problems and their success criteria is also consequently measured in different using different units.

A list of different problem domains from image processing is listed with different KPIs in the following sections. The following subsections list the 3 exemplary problem domains including their top leading algorithms in the respective field. Their ranging is done according to individual benchmarks which vary between the problem domains.

### 3.10.2 SVHN – Dataset

SVHN (Street View House Numbers) is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data pre-processing and formatting. It can be seen as similar in flavour to MNIST (e.g., the images are of small cropped digits), but incorporates an order of magnitude more labelled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images.
Table 1 SVHN - Datasets and Benchmarks

<table>
<thead>
<tr>
<th>Method</th>
<th>Authors</th>
<th>Year</th>
<th>Error [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoAugment: Learning Augmentation Strategies from Data [25]</td>
<td>Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le</td>
<td>2018</td>
<td>1.02</td>
</tr>
<tr>
<td>Improved Regularization of Convolutional Neural Networks with Cutout [26]</td>
<td>Terrance DeVries, Graham W. Taylor</td>
<td>2017</td>
<td>1.30</td>
</tr>
<tr>
<td>Drop-Activation: Implicit Parameter Reduction and Harmonious Regularization [27]</td>
<td>Senwei Liang, Yuehaw Khoo, Haizhao Yang</td>
<td>2018</td>
<td>1.46</td>
</tr>
</tbody>
</table>

3.10.3 CIFAR-100 - Dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Table 2 CIFAR-100 -- Datasets and Benchmarks

<table>
<thead>
<tr>
<th>Method</th>
<th>Authors</th>
<th>Year</th>
<th>Accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [30]</td>
<td>Mingxing Tan, Quoc V. Le</td>
<td>2019</td>
<td>91.70</td>
</tr>
<tr>
<td>GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism [31]</td>
<td>Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, Zhifeng Chen</td>
<td>2018</td>
<td>91.30</td>
</tr>
</tbody>
</table>

3.10.4 KITTI Optical Flow

This benchmark consists of 200 training scenes and 200 test scenes (4 colour images per scene, saved in loss less PNG format). It comprises dynamic scenes for which the ground truth has been established.
in a semi-automatic process. The evaluation is accomplished by computing the percentage of bad pixels averaged over all ground truth pixels of all 200 test images. For this benchmark, we consider a pixel to be correctly estimated if the disparity or flow end-point error is <3px or <5% (for scene flow this criterion needs to be fulfilled for both disparity maps and the flow map). All methods are required to use the same parameter set for all test pairs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Authors</th>
<th>Year</th>
<th>Flow Outliers [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic Refinement Networks [33]</td>
<td>Anne S. Wannenwetsch, Stefan Roth</td>
<td>2020</td>
<td>6.06</td>
</tr>
<tr>
<td>MaskFlownet: Asymmetric Feature Matching with Learnable Occlusion Mask [34]</td>
<td>Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I-Chao Chang, Yan Xu</td>
<td>2020</td>
<td>6.11</td>
</tr>
</tbody>
</table>
4 Optimization Models and Techniques for Dynamic Planning in Logistics

4.1 Introduction

Among routing problems, probably the most well-known problem in this field is the Travelling Salesman Problem (TSP) [118]. The TSP, together with the Problem of Allocation of Routes to Vehicles, or VRP (Vehicle Routing Problem) [120], is one of the most well-known and widely studied problems in the history of computer science and operational research. In the TSP, there is a set of customers and a single vehicle. The objective of the problem is to find a route that, starting and ending at the same point, visit once every node minimizing the total cost of the trip [135].

In the case of the VRP, the main objective is to find a minimum set of routes with the minimum possible cost that (i) start and end in the established deposit, in which (ii) each client is visited only once and (iii) the total demand of customers visited on a route does not exceed the capacity of the vehicle that performs it. These problems fall within the field of combinatorial optimization and are NP-hard problems, whose optimal solution becomes computationally intractable to obtain, once the size of the graph increases [97][106].

In almost all real-world applications, uncertainty is an inherent characteristic of the problem. This uncertainty can be present, for example, in the demand, travel times, unexpected events (e.g. vehicle breakdown), etc. The stochastic VRP (SVRP) is a variant of the VRP where one or more parameters are stochastic, meaning that are given by random variables with a known probability distribution [104].

Apart from uncertainty, in real-world applications, it is common that one or more elements of the problem vary over time. In the dynamic VRP (DVRP), also referred to as real-time or online VRP, some input data is disclosed or changed during the dispatching phase of the distribution planning. The most common dynamic events in VRPs is the arrival of new customer pick-up/delivery requests, or variations in service and travel times [142].

In addition to the above, when solving VRP in real-world environments, in particular in last-mile environments, it is not enough to optimise the distance travelled or the working time, but there are several objective or performance measures to be optimised, such as cost or emissions. Moreover, these objectives are often conflicting, making their joint optimisation complex. Multi-objective VRP [125] models are a variant of the classical VRP in which two or more objectives or performance measures are optimised at the same time.

In parallel to the wide variety of VRP models, a multitude of optimisation method proposals [114] have also emerged, ranging from exact to approximate and hybrid methods.

Having said that, the objective of this section is to review the state of the art in terms of the VRP optimisation models and optimization methods most aligned with SENATOR’s objectives, as well as to assess their main strengths and weaknesses. With this objective in mind, the rest of this section is structured as follows. Section 4.2 reviews the state-of-the-art in the VRP variants that are most aligned with SENATOR’s objectives and in particular task 2.2. Section 4.3 shows the state-of-the-art regarding the most commonly used optimisation methods to solve these VRP variants. Section 4.4 reviews the main software platforms and tools available for vehicle routing optimisation. Section 4.5 assesses the
strengths and weaknesses of the different VRP optimisation models and methods. And finally, Section 4.6 describes how we expect to design the benchmark tests of the models and algorithms to be developed in tasks T2.2 and T2.3, respectively.

### 4.2 Mathematical Models for Vehicle Routing Optimization

The VRP is generally defined on a graph \( G = (V, E, C) \), where \( V = \{v_0, v_1, \ldots, v_n\} \) is the set of vertices; \( E = \{(v_i, v_j) | (v_i, v_j) \in V^2, i \neq j\} \) is the arc set; and \( C = (C_{ij}(v_i, v_j))_{(v_i, v_j) \in E} \) is a cost matrix defined over \( E \), representing distances, travel times, or travel costs. The VRP consists of finding a set of routes for \( K \) identical vehicles based at the depot, such that each of the vertices is visited exactly once while minimizing the overall routing cost. In the classic version of the VRP, there is a set of customers, a fleet of vehicles with a limited capacity and a warehouse [106].

From this basic VRP definition, a wide variety of variants of the problem have been proposed with the aim of tackling increasingly realistic and complex problems, which allow modelling the different causalities that typically occur in real environments. These mainly concern highly complex problems where a wide variety of constraints or vehicle types must be addressed, where dynamism and uncertainty are present, and where more than one objective needs to be optimised simultaneously. This need to model and solve new VRP typologies, as well as the importance that this type of problem has in many sectors (e.g. logistics, transport, mobility, etc.) has attracted the attention of an important part of the scientific community. As proof of this, the number of journal articles published in the field of vehicle routing has grown remarkably in recent years. This trend can be seen in Figure 26. Data were obtained from the Scopus bibliographic database.

![Figure 26](image)

Figure 26: Evolution of the number of publications containing “Vehicle Routing Problem” keywords on Scopus on April 27th, 2021.

In the next part of this subsection we will review the state of the art in each of the main families of VRP variants, and, those that are most aligned with SENATOR's objectives and Task 2.2, as mentioned above.
4.2.1 Rich VRP

VRPs are an extremely broad class of problems and the literature on VRPs shows a clear trend towards the study of more complex problems to reduce the gap with real-world applications. Problems may differ because of additional features a route must satisfy, such as constraints on time duration, time windows or pickup and delivery sequences of operations. In recent years, several variants of multi-constrained Vehicle Routing Problems have been studied, forming a class of problems known as Rich Vehicle Routing Problems (RVRP) [128][154].

The term Rich Vehicle Routing is associated with problems that represent some or all aspects of a real-world application including optimization criteria, constraints, and preferences. These problems deal with realistic (and sometimes multi-objective) optimization functions, uncertainty (i.e., stochastic or fuzzy behaviours), dynamism, along with a wide variety of real-life constraints related to time and distance factors, use of heterogeneous fleets.

The first reference to define the RVRP was made by Toth and Vigo [154]. The authors define the potential of extending the “vehicle flow formulations, particularly the more flexible three-index ones.” A taxonomy for RVRPs is shown in Figure 27. This taxonomy has been extracted from [128]. It is designed according to central concepts in routing that are often present in industrial applications. This is related to characteristics that alter the nature of the problem significantly. It focuses on the most important and relevant concepts while maintaining a moderate level of granularity [107][128]. These authors divide the features of the problem into two main blocks. The first block refers to the characteristics of the scenario, which includes strategic and tactical features of the problem definition such as input data, management, number of depots, type of operation, load distribution, planning horizon or the use of multiple vehicles. The second block refers to the physical characteristics of the problem itself and includes operational aspects such as vehicle features, time constraints, time window structure, incompatibility constraints, specific constraints and definition of the objective function.

![Figure 27 Main characteristics of RVRP (Extracted from [128])](image)

For the sake of simplicity and concreteness, in this deliverable, we will focus on those features of RVRPs most closely aligned with the requirements of the VRP models that are intended to be...
developed in SENATOR Task T2.2. Specifically, on the modelling of demand, heterogeneous vehicle fleets and constraints.

### 4.2.1.1 Demand

In the classic variant of the VRP, the demand of each customer is entirely served by one vehicle and a single commodity is considered. However, in some real problems, the location of the customer is known in advance, but the actual demand is revealed when the vehicle arrives at the customer location. Furthermore, it can occur that the required demand cannot be met, and the vehicle must return to the depot for replenishment before serving the customer. Examples of the modelling of these problems can be found in [152] and [111].

Another important characteristic of demand relates to the type of operation that takes place at the customer location. In this sense, in the classical version of the VRP all goods are considered to be loaded at the depot and distributed to the customers, or conversely, picked up at the customers and at the end of the route unloaded at the depot. However, in last-mile logistics, the common situation is to have both pick-up and delivery orders in the same route. This variant of the VRP is known as the VRP with pick-up and deliveries [127].

Another subvariant occurs when the pick-up and deliveries can be simultaneous. For example, in the distribution system of grocery store chains, each grocery store may have demand for both delivery (fresh food or soft drinks) and pickup (outdated items or empty bottles) and is served with a single stop by the supplier [157]. Another subvariant occurs when the consolidation of the goods is required and it must be done using cross-dock facilities or intermediate depots [99].

Recently, another element within last-mile problem modelling that has gained prominence in recent years concerns parcel lockers that are devices that aim at concentrating the demand for delivery or pick up of small parcels in specific locations. An Automated Parcel Locker System (APLS) [137] is a parcel collection service that allows customers to have their parcels delivered to SPs and pick them up at any time of the day using digital codes. The APLS are located usually in public places or premises such as public transport stations, shopping centres, gas stations, etc. The increasing popularity of APLSs is creating opportunities to improve the efficiency and the sustainability of the distribution of small deliveries. Examples of the modelling of these problems can be found in [121] [137].

Usually, VRP variants assume that customers can only be served once by a single vehicle, and load splitting is not allowed. Load splitting occurs when a vehicle can serve the customer demand in multiple trips by, for example, serving half of the demand, going back to the depot, filling up the vehicle and serving the other half of the demand, or simply, because it is served by two or more vehicles. This is the case for the VRP with split deliveries [122].

Another interesting demand related variant is the VRP with release and/or due dates [148]. The release date is defined as the date on which the order to be delivered to a certain customer is available in the warehouse. Due dates refer to the date by which the order should be delivered to the customer at the latest.

Finally, the combination of outsourcing and split delivery features has received increasing attention in the literature in recent years [98] [103] due to the rising of last-mile logistics. The VRP with outsourcing (VRPPO) models those problems in which a customer can be served using the own facilities and fleet, or by an external (outsourced) carrier. A solution to this problem consists of differentiating between the routes for the own vehicles and routes for customers for which the demand is outsourced.
4.2.1.2 Heterogeneous fleet

The Heterogeneous VRP (HVRP) [149] assumes that the vehicles have different capacities and/or other characteristics (e.g. capacity or travel times using bike, motorbike, van, truck). In this type of problems, mixed fleet of vehicles, having distinct capacities, fixed costs and travel costs, are used to serve a set of customers, minimizing the total costs, subject to the service duration constraints, capacity constraints, etc.

If we focus on the capacity, in the literature we can find variants where all vehicle have the same capacity, which are called VRP with homogeneous fleet; different capacity, which are a sub-class of the already mentioned Heterogeneous VRP; or load-specific capacity [115] which can only accommodate one or more specific load or loads, each of them with their corresponding capacities (e.g., multi-compartment vehicles where each compartment is dedicated to one specific good).

With the rise of e-commerce and last-mile logistics, many companies offering courier services often have a heterogeneous fleet increasingly composed of different types of vehicles [140] as walking, bike, cargo bikes, motorbikes, vans, trucks, etc. For this reason, we can find in recent years more and more variants of the heterogeneous vehicle routing problem that incorporates these greener vehicles [108][141].

4.2.1.3 Constraints

In addition to the allocation of demand to vehicles, the order in which demand is served and the type of vehicle used, there is another very important element that influences the applicability of the models in real environments and that strongly affects their complexity. These elements are the constraints and relate to additional features a route must satisfy, such as constraints on time duration, time windows or pickup and delivery sequences of operations. Constraints can be classified according to the company decision levels [107]: operational, tactical, and strategic (see Figure 28). The difference between the three levels depends on the types of decision involved. Below we describe more in details each of these levels of constraints.
The operational level is associated with distribution planning including the vehicle and the driver schedules. Some of the most important sub-categories of constraints related to this level are:

- Vehicle fleet composition
- Vehicle Capacity
- Driver’s regulations

The tactical level includes constraints that do not affect the daily activities but that have an important impact on the routing plans. Some sub-categories of constraints within this level are:

- Time windows
- Order type
- Multiple visits/split deliveries
- Customer capacity

Finally, at the strategic level, the constraints are related to different functions of the supply chain or transportation planning. This level includes the sub-categories of constraints such as:

- Multi-depot
- Depot related constraints
- Constraints related to temporal and spatial characteristics of the start and end of the routes.
4.2.2 Stochastic VRP

The Stochastic VRP (SVRP) is the variant in which one or several components of the problem are random and follows a probability distribution. SVRP can be divided into the VRP with stochastic demand (VRPSD), the VRP with stochastic customers (VRPSC), the VRP with stochastic demands and customers (VRPSDC), and the VRP with stochastic travel and service times (VRPSTS) [104][105].

4.2.3 Dynamic VRP

In last-mile logistics, customer requests need sometimes to be serviced as early as possible, requiring immediate re-planning of the current vehicle route. Apart from this, some events, as vehicle breakdown, may also imply the re-planning of vehicle routes. Furthermore, the dynamic routing of
vehicles may allow introducing greater opportunities to reduce operational costs, improve customer service, and reduce environmental impact.

Dynamic vehicle routing problems (DVRP) involve new elements that increase the complexity of their decisions (more degrees of freedom) and introduce new challenges. The most common source of dynamism in vehicle routing is the online arrival of customer requests during the operation. Travel time is also a dynamic component of most real-world applications [142][144]. This class of problems has also gained popularity in recent years because it is very well suited for modelling just-in-time supply systems and also because of recent technological advancements, such as mobile devices or sensors, that allow drivers to dynamically change their plan while executing the route [136].

In the classic DVRP, $m$ vehicles with fixed equal capacity, $q_i (i = 1, ..., m)$, depart from a depot to deliver products to $n$ number customers at demand points. Each customer has a known demand $d_i (i = 1, ..., n)$, where $n$ is the number of customers. It is assumed that the quantities demanded are less than the maximum capacity of the vehicles. Meanwhile, new customers with known demand emerge dynamically over time. A graphical example of the classic VRP is shown in Figure 30.

The DVRP has dynamic demands that arrive in the system at different times. These demands obviously affect the solution because they change both the problem and the solution when they arrive in the system. The challenge here consists of the construction of routes from a depot while minimizing the total distance travelled. Figure 31 shows the main components that may be changing over time in DVRPs [136][144].
DVRP can include the emergence of new customers, where additional customers arrive at an unknown location when the vehicles are “en route”. In this case, the usual objective consists of maximizing the probability that these additional customers can be served without violating time constraints [136]. Similarly, to the case of dynamic customers, demand can also vary over time [132]. The dynamism in service times is commonly related to the variation in demand, since if a client varies his demand, the service times in that client may also change. Nevertheless, other elements as the availability of resources in customer premises may also affect service time [159]. As mentioned before, another element of dynamism is travel times because, in real scenarios, they usually vary over time, possibly as a result of traffic congestion or other factors that may impact conditions along the route during the day [158].

4.2.4 Multi-objective VRP

The classic VRP variants aim to minimize a single objective that is not usually suitable for real-life instances, where we usually have different factors to be considered. Moreover, the objectives may be conflicting in nature. For instance, in some sectors like delivery of perishable foods, customer satisfaction and timely delivery is more important than minimizing the distance travelled. To this end, a family of VRP called multi-objective VRP (MOVRP) was proposed to deal with these real-life instances [157].

The purpose of this family of problems is to extend the classical VRP to increase their practical applications. As an example, we can consider some objectives like the driver workload, customer satisfaction, GHG emissions, etc. [110][109].

Another way to use MOVRP is to generalize classical problems by adding objectives instead of adding one or several constraints or parameters. More specifically, in the MOVRP, the overall optimization function is influenced by two or more objectives. These objectives, as mentioned before, are sometimes conflicting in nature i.e., there exist some trade-offs between them. Formally, MOVRP can be stated as:
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 861540

\[ \text{MOVRP} \rightarrow \max / \min F_x = (f_1(x), f_2(x), \ldots, f_k(x)) \quad \text{s.t } x \in D. \]

Where \( k \geq 2 \) is the number of objective functions to be optimized, \( x = (x_1, x_2, \ldots, x_k) \) are the decision variables, \( D \) is the search space of feasible solutions and \( F_x \) is the objective function [157]. Another important different w.r.t the classic VRP lies in the fact that in this category of problems there is no single optimal solution, but several. Concretely, the solution to the MOVRP is the set of non-dominated solutions called Pareto set or Pareto front [112]. The Pareto front is a set of non-dominated solutions, which fulfill the Pareto optimality property. The Pareto optimality property is fulfilled when no individual objective can be better off without making at least one individual objective worse. If they are represented in a graph as shown in Figure 32, the objective functions of the set of solutions will form what is called the Pareto Front and represent the boundary of the space between the feasible and non-feasible solutions.

![Figure 32 Example Non-dominated solutions called Pareto set](image)

The most common objectives used in the MOVRP are [150]:

- **Tour Related Objectives**: these objectives are considered from an economic point of view. Normally it is expressed in terms of total travel distance, the number of customers visited, and time needed [156]. This type of objectives is considered to provide equity between workers and customers [151].

- **Resource Related Objectives**: resources are mainly vehicles, personnel, and equipment. This class of objectives have both economic and environmental significance. For example, a smaller number of used vehicles requires less investment cost and thus fewer emissions of GHG [117]. The minimization of the use of human resources and/or equipment needed to serve demand also falls within this category [123].

- **Node/Arc Related Objectives**: in this class of problems, the hard fulfilment of time windows is replaced with minimization of the violated time window constraints [160]. Other objectives considered under this category are maximizing customer satisfaction [145].
4.3 Optimization techniques

VRPs variants can be classified into three levels according to the degree of realism of their associated models. Aligned with this realism in the modelling, and the class of optimization techniques used to solve them usually vary [97][107][114], as shown in Figure 33. Using this classification as a reference, we will review in the next part of this section the main categories of optimization algorithms used to solve the VRP.

![Figure 33 Model's classification and optimisation techniques [107]](image)

4.3.1 Exact algorithms

Since the classical VRP problem is NP-hard in the strong sense, all sub-variants are also NP. This means that there is no deterministic algorithm that guarantees to find the optimal solution within a computing time that is bounded by a polynomial in the input size. This means that only small size instances can be handled by exact methods.

Branch-and-cut is an exact method of combinatorial optimization for solving integer linear programs (ILPs), that is, linear programming (LP) problems where some or all the unknowns are restricted to integer values. Branch and cut involves running a branch and bound algorithm and using cutting planes to tighten the linear programming relaxations [102][113].

The set partition problem is the task of deciding whether a given multiset S of positive integers can be partitioned into two subsets S1 and S2 such that the sum of the numbers in S1 equals the sum of the numbers in S2. Although the partition problem is NP-complete, there is a pseudo-polynomial time dynamic programming solution, and there are heuristics that solve the problem in many instances, either optimally or approximately. Based on this approach, some authors define set partitioning formulations that are exact methods to solve VRPs [100]. The main idea of the formulation is that authors associate a binary variable with each feasible route to search for optimal solutions.

Another algorithm is 2-opt, a simple search algorithm for solving the travelling salesman problem. The main idea behind it is to take a route that crosses over itself and reorders the sequence of nodes to traverse so that it avoids the crossing. A complete 2-opt local search will compare every possible valid
combination of the swapping mechanism. This technique can be applied to the travelling salesman problem as well as many related problems. These include the vehicle routing problem (VRP) as well as the capacitated VRP, with minor modification of the algorithm [102].

4.3.2 Approximate algorithms/ Metaheuristics

Metaheuristics are widely recognized as efficient approaches for many hard optimization problems [143]. They represent a core research field in combinatorial optimization, the VRP is an NP-hard problem and furthermore, its real-life VRP applications are considerably larger in scale. Therefore, metaheuristics are often more suitable for practical applications. The main reference metaheuristics in the literature that are used to solve the VRPs are discussed below.

4.3.2.1 Simulated Annealing

Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space for an optimization problem. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of time, simulated annealing may be preferable to alternatives such as gradient descent [96].

It is implemented using the notion of a cooling temperature interpreted as a slow decrease in the probability of accepting worse solutions as the solution space is explored. Accepting worse solutions is a fundamental property of metaheuristics because it allows for a more extensive search for the global optimal solution. In general, the simulated annealing algorithms work as follows. At each time step, the algorithm randomly selects a solution close to the current one, measures its quality, and then decides to move to it or to stay with the current solution based on either one of two probabilities between which it chooses on the basis that the new solution is better or worse than the current one. During the search, the temperature is progressively decreased from an initial positive value to zero [96].

4.3.2.2 Variable Neighbourhood Search

Variable Neighbourhood Search (VNS) is a metaheuristic based upon systematic changes of neighbourhoods both in the descent phase, to find a local minimum, and in the perturbation phase to emerge from the corresponding valley. It was first proposed in 1997 and has since then rapidly developed both in its methods and its applications. VNS embeds a local search heuristic for solving combinatorial and global optimization problems. It allows a change of the neighbourhood structures within the search [114][136].

4.3.2.3 Ant Colony Optimization

Ant colony optimization (ACO) is a population-based metaheuristic for the solution of difficult combinatorial optimization problems. In ACO, each individual of the population is an artificial agent that builds incrementally and stochastically a solution to the considered problem. Agents build solutions by moving on a graph-based representation of the problem. At each step, their moves define which solution components are added to the solution under construction. A probabilistic model is associated with the graph and is used to bias the agents’ choices. The probabilistic model is updated
online by the agents so as to increase the probability that future agents will build good solutions [134][158].

4.3.2.4 Large Neighbourhood Search

Large Neighbourhood Search (LNS) is a meta-heuristic in which the neighbourhood of a solution is defined implicitly by destroying and repair operators. A destroying operator destroys part of the current solution while a repair operator rebuilds the destroyed solution. Typically, the destroy method contains some randomness such that different parts of the current solution are modified so that enabling exploration of the solution search space. This exploration technique enables larger neighbourhoods to be visited in comparison to standard neighbourhoods of classical local search methods. This property has made this method became the state of the art in many variants of the vehicle routing problem [54][55][57][69] and that is also why it is the method most commonly implemented in many software libraries and packages related to this field [96].

4.3.3 Hybrid algorithms

In recent years, for the resolution of VRP problems, hybrid algorithms have emerged as a powerful tool to solve them, especially for the most complex VRP variants [114]. Hybridization has become a very promising strategy for designing better metaheuristics methods, because of their greater flexibility, less strict mathematical formulations and higher robustness. In this way, they provide a very suitable tool to develop solvers for VRP, and as a matter of fact, they have become state-of-the-art in many variants of the rich VRP [114][119]. Following the well-known taxonomy of hybrid algorithms proposed in [147], we will review significant literature in this area by considering three classes of hybrid algorithms: metaheuristics hybridized with metaheuristics, metaheuristics hybridized with problem specific metaheuristics, and metaheuristics hybridized with other Operational Research (OR) or Artificial Intelligence (AI) techniques.

4.3.3.1 Hybridization of Metaheuristics and Metaheuristics

The combination of metaheuristics with metaheuristics is the most common category of hybrid algorithms that we can find for the VRP. For example, the paper in [130] proposed a hybrid algorithm of Simulated Annealing and TS. The algorithm combines the advantages of both SA and TS with local search with the idea of not restricting a move by the search algorithm within the solution space which can prove to lower the objective function. The long-run effect is to allow movement in the solution space which results in increasing objective function. In another paper [153], presented another hybrid algorithm to solve VPR by imposing a limit on the maximum allowable distance that each vehicle in the homogeneous fleet can travel. The problem was solved for a near-optimal solution using a hybrid swarm-based metaheuristic that integrates VNS (Variable Neighbourhood Search) within the particle swarm optimization (PSO). More recently, the authors in [133] introduced a hybrid heuristic consisting of an iteration of a particle swarm algorithm, a local search improving each particle found in the previous step and a large neighbourhood search to escape from local optima. In the same year, Avci et al. in [101] develop a hybrid local search algorithm in which a non-monotone threshold adjusting strategy is integrated with tabu search. The threshold function used in the algorithm has an adaptive nature which makes it self-tuning. Additionally, its implementation was very simple as it requires no parameter tuning except for the tabu list length. Another interesting approach was presented in [131] by the hybridization of Ant Colony Optimization and Tabu Search algorithms for solving the VRP with
the compatibility constraints. ACO is used to search for a globally promising area, and then TS continues to optimize it to obtain a high-quality solution. The initial solution of TS is provided by the final solution of ACO. Recently, the work in [146] proposed a hybridization of Gravitational Emulation Local Search (GELS) and Genetic Algorithm for solving Capacitated VRP. They use three standard benchmarks that include different test problems found in the literature, and compare the results with other metaheuristic algorithms, obtaining competitive results.

4.3.3.2 Hybridization of metaheuristics with problem-specific algorithms

As mentioned above, another important class of hybrid algorithm for solving the VRP is the combination of metaheuristics with problem-specific algorithms. One example of this type of techniques can be found in [155], where the authors used a hybrid genetic search metaheuristic with three components of assignment, sequencing and route evaluation. In the work presented in [109], a hybrid genetic algorithm is proposed which makes use of a route decomposition technique for chromosome decoding and a local search, to solve the multi-trip VRP with time windows and release dates. A more recent example of this class of methods can be found in [129] where a Hybrid Genetic Algorithm for solving the Capacitated VRP in the Internet of Things area was presented. The authors created an initialization algorithm solution that combines a sweep algorithm with random components. Then they propose a specific crossover operator that generates feasible solutions, checks the constraints of the problem and integrates with a neighbourhood search heuristic.

4.3.3.3 Hybridization of Metaheuristics and OR/AI methods

The last class of hybrid algorithms consist of the combination of metaheuristics and OR/AI methods. This category of hybrid algorithms is gaining more attention in recent years because of its excellent results, even though they started to be proposed more than a decade ago. For example, in [126], the authors combine a hybrid ACO algorithm for solving vehicle routing problem heuristically with an exact algorithm to improve both the performance of the algorithm and the quality of solutions. Another interesting example of these methods can be found in [116] where an Artificial Ant Colony (AAC) based on the 2_Opt local search algorithm to solve the dynamic pickup and delivery VRP is presented. In the AAC meta-heuristic, a set of agents(ants) build solutions to the given problem cooperating through pheromone update. The success principles of AAC consist in intelligent exploitation of the problem structure and in an effective interplay between the search space and the solution space elaborating with the local search.

More recently, Jabir et al. in [124] develop an ACO based meta-heuristic for solving small scale instances and large-scale instances. Moreover, they develop a hybrid ACO based meta-heuristic integrated with Variable Neighbourhood Search for solving large scale instances. Additionally, proposed Integer Linear Programming models for a multi-depot vehicle routing problem.

4.4 Current software tools for dynamic planning in logistic

This section aims to review and assess different examples of tools currently available to solve the VRP. For this purpose, we will focus only on the most well-known ones, which are more widely used in the academic and/or industrial field, and which offer functionalities in line with the objectives of the SENATOR project. Regarding this last point, in a more specific way, we have taken the following criteria into account:
- Generality, the possibility of solving various VRP types.
- Flexibility in the set-up of the definition of the optimization model and method.
- Computational efficiency, especially for large problem instances.
- Efficacy in terms of the obtained solutions.
- Availability and quality of the documentation of the tool.
- Ease of integration with external systems as well as GIS and visualization systems.

Below, we list and describe each of these tools, and after that, we show a comparative analysis among them.

### 4.4.1 JSprit

JSprit is a java-based, open-source vehicle route optimization engine. It is lightweight and flexible, and based on a generic Ruin & Recreate metaheuristic. Currently, JSprit has been successfully used to solve different VRP variants, such as:

- Capacitated VRP
- Multiple Depot VRP
- VRP with Time Windows
- VRP with Backhauls
- VRP with Pickups and Deliveries
- VRP with Heterogeneous Fleet
- Time-dependent VRP

It is also flexible in the sense that setting up the problem can be performed in several manners. For example, it allows the definition of ad-hoc constraints, the configuration and set-up of the optimization algorithms, and the visualization of the found solutions. JSprit is very appropriate for its modification and extension due to its modular design and a comprehensive set of unit and integration tests available.

The architecture implemented in JSprit to manipulate (create or modify) constraints is based on Java Reflection. The constraints can be divided into Soft and Hard, and, at the same time, these can be divided into route constraints and activity constraints. The constraint associated with the route refers to restrictions that the created route must comply with, for example, the maximum or minimum time that a vehicle can travel. In the case of constraint associated with activities, they refer to criteria that activities must meet when a route is modified, for example, a new pickup should fit in between Activity 1 and Activity 3.

It is important to mention that JSprit is designed and implemented in a flexible manner which makes it easy to change or evolve it for particular usage. The general schema of its design is shown in Figure 34. In this way, it facilitates, for instance, defining an arbitrary number of iterations for the solver, defining how an initial construction is formed, the overall solution search strategy, etc.
4.4.2 OR-Tools

OR-Tools\(^2\) is an open-source software suite for optimization, tuned for tackling hard problems in vehicle routing, flows, integer and linear programming, and constraint programming. The architecture provides programming language wrappers for operations research tools such as optimisation and constraint solving. OR-Tools was developed in the C++ language, but also provide wrappers in Python, C#, and Java.

OR-Tools includes a specialized routing library to solve different types of node-routing problems, such as:

- Traveling Salesman Problems (TSP)
- Vehicle Routing Problems (VRP)
- Capacitated Vehicle Routing Problems (CVRP)
- Vehicle Routing Problems with Time Windows (VRPTW)
- Vehicle Routing Problems with Resource Constraints
- Vehicle Routing Problems with Pickup and Delivery (VRPPD)

The route solver that was developed for OR-Tools includes different set-up parametrizations such as the maximum duration, the number of solutions found, the initialization heuristics, or the global solver (e.g. Greedy Descent, Guided Local Search, Simulated Annealing and Tabu Search). Moreover, in the search process, the state of the search can be printed, which returns an integer value that can have different interpretations.

---

1. [https://github.com/graphhopper/jsprit/tree/master/docs](https://github.com/graphhopper/jsprit/tree/master/docs)
2. [https://github.com/google/or-tools](https://github.com/google/or-tools)
4.4.3 VROOM

VROOM\(^3\) is open-source software written in C++ to solve vehicle routing problems arising in logistics and more widely in any context with geographically distributed tasks. VROOM offers out-of-the-box integration with different open-source route engines such as OSRM, Openrouteservice and Valhalla. VROOM uses several heuristics to find an initial solution depending on the problem and an adjusted version of the clustering heuristics using spanning trees for CVRP. Then the local search procedure is used to check for valid neighbour solutions (by using different operators as 2-opt) and improve the current solution iteratively.

The solving approach aims at providing high-quality solutions efficiently by using dedicated metaheuristics. This allows to get solutions very fast and/or to scale to big problem sizes. VROOM can solve several well-known types of vehicle routing problems (VRP).

- TSP (travelling salesman problem)
- CVRP (capacitated VRP)
- VRPTW (VRP with time windows)
- MDHVRPTW (multi-depot heterogeneous vehicle VRPTW)
- PDPTW (pickup-and-delivery problem with TW)
- Mix of the above problem types.

4.4.4 VRP Service from ArcGIS (ESRI)

The VRP service from ArcGIS on-line\(^4\) is a commercial service developed by ESRI to address different routing problems. This service can be accessed in different ways such as Java Script APIs, SDKs in different programming languages as Java, .NET, etc.

It is a highly customisable tool that allows a large number of VRP variants to be solved:

- Capacitated VRP
- Multiple Depot VRP
- VRP with Time Windows
- VRP with Pickups and Deliveries
- VRP with Heterogeneous Fleet
- VRP with job and vehicle skills

Another interesting aspect, which differentiates it from the other tools analysed, is its integration with the rich ArcGIS Geographic Information System. This means that, in addition to the VRP modelling itself, it is also possible to incorporate geographical or locational aspects, such as areas restricted to certain vehicles, areas with different speeds, etc. Another interesting feature of this tool that is not found in other ones, is the fact that it allows prioritizing a vehicle when it is assigned to a particular delivery zone.

However, this tool also has a series of disadvantages:

- It is a commercial solution with a pay per use policy which make it costly especially for large vehicle fleet and high-frequency use.

---

\(^3\) [https://github.com/VROOM-Project/vroom](https://github.com/VROOM-Project/vroom)

- The objective function only works with one objective mainly based on the minimization of distance and/or time, but it is not possible to adapt it to other objectives such as environmental impact.
- Although it is a flexible tool with a lot of different functionalities to customize VRP instances, its extension is not possible (e.g. to include parcel lockers).

4.4.5 Circuit

Circuit\(^5\) is a commercial tool, which is available on different platforms such as web service, Android and iOS. It allows you to combine current traffic conditions with the most up-to-date map data to plan and optimize the order of your delivery route, road trip or travel plan. Among the main features we can find optimization of up to 1000 stops, which allows to establishing stop time windows, establish the first and last stop, and priority levels for each stop.

The main use cases that it implements are:
- Driver Tracking
- Local Delivery
- Route Planning
- Proof of Delivery
- Courier Management

Moreover, Circuit is compatible with many third-party navigation apps, such as Google Maps. Like ArcGIS online, its custom configuration and extension are very difficult or impossible due to the licencing constraints.

4.4.6 LOCUS

LOCUS\(^6\) is a tool that allows the planning of routes and the assignation of vehicles for a certain number of orders and the route that the vehicles must take on each journey. It implements the following problems: Travelling salesman problem, Vehicle Routing Problem (constraints: capacity, resources, drops and pickups, time window), Knapsacking Problem. For delivery drivers, this tool can be downloaded as a mobile application for Android and iOS. In their solution, they implement exact algorithms, heuristic algorithms and hybrid algorithms but they do not give public details about the type of algorithms they include.

The main use cases that it addresses are:
- Last-Mile Delivery Routing
- Field Service Dispatch Planning
- Dynamic Route Planning and Optimization
- Territory-Based Route Planning
- Reverse / Returns Logistics

They implement a solution for Last-Mile Delivery Routing, considering that any modification in the last section of a shipment can lead to an exponential increase in delivery costs. Besides, it includes last-mile carrier tracking functionality to provide actionable delivery visibility and real-time tracking.

\(^5\) [https://getcircuit.com/](https://getcircuit.com/)

\(^6\) [https://locus.sh/](https://locus.sh/)
LOCUS includes also Dynamic Route Planning and Optimization in real-time, considering orders that may arrive at the last minute, and it also guarantees that the delivery plan is not interrupted by reorganizing the orders.
LOCUS includes an API to access its services through REST calls, making it easy to integrate into any application. It does not contain a free trial, so you can only access a demo to test a single service.

4.4.7 OptaPlanner
OptaPlanner\(^7\) is an open-source tool under the Apache License, and it is developed in Java. This tool includes the implementation of several optimization problems, among them, the Vehicle Routing Problem and the variants Capacitated VRP and VRP with Time Windows.
OptaPlanner implements optimization algorithms such as Tabu Search, Late Acceptance and Simulated Annealing. It also allows the development of new construction heuristics of the initial solution and metaheuristics as a solution method. It also allows the integration with Google Maps and OpenStreetMap.
They propose that part of the local search algorithms can be parallelized in several threads of execution of the same solver, which should improve the performance of the optimiser. They also have a benchmark of problems with which they have tested the different algorithms they implement, which can serve as a comparative baseline for users.

4.4.8 Here
HERE\(^8\) provides a route planning API to solve the vehicle routing problem, it is a proprietary tool, developed in Java. In addition, the services can be consumed by REST, and it has applications for Android and iOS. The variants of VRP that it has implemented are:
- Capacitated vehicle routing problem
- Vehicle routing problem with time windows
- Multi-depot vehicle routing problem
- Open vehicle routing problem
- Heterogeneous or mixed fleet VRP
- Pickup and delivery vehicle routing problem

It allows the calculation of the routes, using real-time and historical traffic information. Furthermore, it is possible the re-planning of routes in real-time if new orders appear. To do this, the generated routes are used as input to accommodate new jobs without interruption, based on the current locations of the vehicles.

4.4.9 GraphHopper
GraphHopper\(^9\) provides an API to solve a variety of vehicle routing problems, including the classic "Traveling Salesman" optimization problem. It is an open-source software tool developed in Java that uses JSprit as the route optimization engine. As it is based on JSprit, it has all the variants of VRPs that are implemented in this library described in Section 4.4.1. Among their main advantages, we can find

\(^7\) https://www.optaplanner.org/
\(^8\) https://developer.here.com/products/tour-planning
\(^9\) https://github.com/graphhopper/graphhopper
the possibility of designing vehicle types including aspects such as weight, volume, number of passenger seats, among others. In addition, the time windows and service times that drivers need to serve customers can also be defined. Furthermore, it allows you to create multimodal routing.

4.4.10 Carto

CARTO\textsuperscript{10} is the Location Intelligence platform, which through location data allows obtaining routes, the strategic location of facilities and analysing behavioural marketing. They have developed different variants of the vehicle routing problem, including on-demand last-mile transportation: real-time dynamic route optimization. To solve this variant of the problem, they used a greedy algorithm, which according to studies they have carried out, obtains a good solution in a fast time.

In addition, CARTO has an API developed in Python known as CARToframes, to integrate CARTO maps, analysis, and data services into data science workflows, which allows enriching the route planning calculation with additional data such as weather or traffic.

4.4.11 DELMIA Quintiq

DELMIA Quintiq has solutions for modelling, planning, and optimizing operations. They have a logistics planning tool that allows planners to optimize logistics plans through an intuitive planning interface. There is also a tool that displays real-time KPIs reports, giving the possibility that planners have control of the whole process.

DELMIA Quintiq Logistics Planner determines the best possible sequence of visits and their distribution in the routes, the algorithm used for the solution is a proprietary Large Neighbourhood Search (LNS) developed by this company. They use constraint programming to eliminate potential solutions through sophisticated constraint propagation, allowing for a wide variety of constraint types. In addition, they use their own language for optimizer configuration, called Quill, which includes the construction heuristics and the local search heuristics. The VRP variants available are:

- Vehicle Routing Problem with Time Windows
- Pickup and Delivery Problem with Time Windows
- Dynamic Vehicle Routing Problem

4.5 Assessment of different optimization models and techniques

The objective of this section is to assess the strengths and weaknesses of the different VRP models discussed in Section 4.2, the different optimisation methods described in Section 4.3, and the software tools described in Section 4.4 from the perspective of their application to last-mile logistics and according to the objectives of the SENATOR project. We will analyse them in the following subsections, we will analyse each of these two elements.

\footnotesize{\textsuperscript{10} https://carto.com/industries/logistics/}
4.5.1 Assessment of mathematical models for the VRP

We will begin by assessing the weaknesses and strengths of the main categories of VRP optimisation models:

- **Rich VRP.** The classical VRP as we have seen several times throughout Section 4.2 is not sufficient to model all the complexity that we find in last-mile logistics and that has been reviewed in Section 3.9. Therefore, in the context of SENATOR, it will be necessary to use Rich VRP models to address the different causalities that we will encounter within SENATOR. In the following, we will describe in more detail the most appropriate Rich VRP models from the point of view of demand, the vehicle fleet used and the constraints.

  o **Demand.** On the demand side, we will need to deal with VRP models that support simultaneous pick-up and deliveries, as this is the type of orders that are needed in last-mile logistics. In addition, the models generated must allow the use of the Automated Parcel Locker System (APLS), since it is one of the elements foreseen within WP1. Other functionalities that we will have to consider from the demand point of view is to allow release and due dates in the orders, and the possibility of outsourcing some of them. Finally, in our case, it will not be necessary to use load splitting as it is not foreseen as a requirement within WP1 and it is not common in last-mile logistics.

  o **Vehicle fleet.** As for the modelling of the vehicle fleet, this will probably be one of the key elements of the VRP models to be generated in SENATOR, as it will have a major impact from the point of view of their applicability. In this sense, it will be necessary to use heterogeneous and multi-modal vehicle fleets. Therefore, homogeneous VRP models or VRP models in which the heterogeneity of the fleet is only given by different capacities are not appropriate for SENATOR requirements.

  o **Constraints.** The constraints will also be a key element for the applicability of the models. Reviewing each of the different levels of constraints, we have to say that, at the operational level, VPR models will need to have constraints related to the composition of the vehicle fleet, vehicle capacity and driver regulation. At the tactical level, they will have to incorporate time windows and constraints associated with the type of orders. However, multiple visits/split deliveries and customer capacity will not be necessary.

- **Stochastic VRP.** If we analyse stochasticity, from the point of view of the applicability of the generated models it will not be a critical point. As shown in Section 4.4 it is a functionality that is not implemented by any of the tools reviewed. However, its modelling can help to improve the robustness of the solutions found by the optimisation algorithms, since if uncertainty is taken into account, it is possible to find solutions that are robust to variations in certain elements such as travel times or service times. These will be the only elements with uncertainty to be considered in SENATOR, i.e. we will not consider uncertainty in demand and customers.

- **Dynamic VRP.** Unlike stochasticity, the dynamism will be key in the VRP models that we generate for SENATOR, since, as described above, it is a common characteristic present in last-mile logistics, and it is also included in WP1. In this sense, we will consider dynamism
in demand, since in last-mile logistics it is common to receive new orders while vehicles are in dispatch, or to have to reallocate loads due to an unexpected event. We will also consider dynamic travel times but not dynamic service times.

- **Multi-objective VRP.** The multiple objectives will also be another important element to incorporate in the VRP models to be developed in SENATOR. The main reason for this is that several conflicting objectives such as economic cost, environmental impact and quality of service to customers must be considered when planning solutions. It is important to note that in the case of SENATOR we will not look for the Pareto Frontier, as this usually implies a high computational cost, as well as an excessive decision time for managers when determining which of the solutions compose the Pareto Front should be implemented at any given moment. Instead, what we will use is some mechanism of aggregation of objectives (e.g. weighting, using targets, etc.) that offers a single solution to managers considering their preferences and the importance they give to each of the objectives.

### 4.5.2 Assessment of optimization techniques for the VRR

Regarding the evaluation of the optimisation techniques reviewed in Section 4.3 first of all, we have to say that exact techniques are completely discarded for use in SENATOR. The reasons are mainly two: 1) the complexity of the VRP models to be treated is very high, which makes their design very difficult, and 2) their computation time, especially with large instances, is in the order of hours, which makes it hard to put them into practice in real last-mile optimisation environments.

As for approximate methods and metaheuristics, although from the point of view of their practical application they present enormous benefits with respect to exact algorithms (e.g. they allow dealing with models of higher complexity, they offer good solutions in a reasonable time using limited resources, etc.), given the high complexity of the VRP models that we intend to address in SENATOR, the evidence we have so far is that their performance is limited when tackling this type of problems. Finally, from our point of view, and based on the literature reviewed, the optimisation algorithms to be used in SENATOR will be based on the hybridisation of algorithms, since they are very competitive methods when dealing with highly complex VRP models. For the development of these algorithms, we will explore the different forms of hybridisation that we have reviewed in Section 4.3.3, such as the combination of metaheuristics with metaheuristics, with problem-specific heuristics, and with other AI and OR techniques. Furthermore, these hybridisations are likely to be based on Large Neighbourhood Search, as it has been shown to be one of the most robust algorithms for dealing with VRP.
### 4.5.3 Assessment of software tools

The objective of this section is to analyse the functionalities and features of the software tools described in Section 4.4, from the point of view of the desirable characteristics that the VRP models to be developed in SENATOR should have, which were reviewed in Section 4.5.1. This analysis is summarised in Table 4 and Table 5.

#### Table 4: Comparative analysis of software tools for dynamic planning in logistics according to license and Rich VRP features

<table>
<thead>
<tr>
<th>Software Tool</th>
<th>License</th>
<th>Demand</th>
<th>Rich VRP</th>
<th>Vehicle fleet</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Pick-up &amp; Delivery</td>
<td>Order types</td>
<td>APLS</td>
<td>Outsourcing</td>
</tr>
<tr>
<td>JSprit</td>
<td>Open-source</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OR-Tools</td>
<td>Open-source</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VROOM</td>
<td>Open-source</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VRP Service from ArcGIS (ESRI)</td>
<td>Commercial</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Circuit</td>
<td>Commercial</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LOCUS</td>
<td>Commercial</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OptaPlanner</td>
<td>Commercial</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Here</td>
<td>Commercial</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GraphHopper</td>
<td>Open-source</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Carto</td>
<td>Commercial</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DELMIA Quintiq</td>
<td>Commercial</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

11 Automatic Parcel Locker System
Table 5 Comparative analysis of software tools for dynamic planning in logistics according to Stochastic, Dynamic and Multi-Objective VRP features

<table>
<thead>
<tr>
<th>Software Tool</th>
<th>Stochastic VRP</th>
<th>Dynamic VRP</th>
<th>Multi-objective</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Demand</td>
<td>Travel times</td>
<td>Service times</td>
</tr>
<tr>
<td>JSprit</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OR-Tools</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VROOM</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VRP Service from ArcGIS (ESRI)</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Circuit</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LOCUS</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OptaPlanner</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Here</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GraphHopper</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Carto</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DELMIA Quintiq</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
4.6 Benchmarks and Datasets

To conclude this section on models and optimisation techniques for dynamic route planning, we will briefly describe the main guidelines that we will take into account when designing the benchmarks on which we will test the models and optimisation methods to be developed in SENATOR tasks T2.2 and T2.3.

Regarding the models, for the design of the benchmarks, we will rely on the design of other benchmarks of reference in the literature such as:

- **VRP-REP.** It is a collaborative open data platform that offers the possibility of sharing reference instances in the VRP problem and its variants, it also has solutions for those instances ([http://www.vrp-rep.org/](http://www.vrp-rep.org/)). It includes VRP variants such as:
  - Asymmetric Capacitated Vehicle Routing Problem
  - Capacitated Arc Routing Problem with Turn Penalties
  - Capacitated Vehicle Routing Problem
  - Consistent Multi-Compartment Vehicle Routing Problem
  - Capacitated Vehicle Routing Problem with Time Windows, Unmatched pickups and deliveries, Priority tasks, and Inventory restrictions
  - Carrier-Vehicle Traveling Salesman Problem

- **CVRPLIB.** It is a library to solve a Capacitated Vehicle Routing Problem ([http://vrp.atd-lab.inf.puc-rio.br/index.php/en/](http://vrp.atd-lab.inf.puc-rio.br/index.php/en/)) and has instances of the problem and solutions obtained by different authors.

- **Logistikmanagement.** It is a repository of instances of the VRP in the context of last-mile logistics ([https://logistik.bwl.uni-mainz.de/forschung/benchmarks/](https://logistik.bwl.uni-mainz.de/forschung/benchmarks/)).

In any case, although for the design of the benchmarks we will consider these reference repositories, they will be based on real data extracted from the two testing scenarios foreseen in SENATOR, i.e. Zaragoza and Dublin. Given that at the time of writing this deliverable, the use case definition and requirements are still in progress, and real-data from testing scenarios is not still available, we will only give general guidelines about the characteristics of the instances that we plan to use for the assessment and validation of the developed models and algorithms:

- **Demand:** The demand will cover the geographical area of the two cities and will include both pick-ups and deliveries. In case the complexity is too high, the possibility of splitting the demand by geographical areas will be considered. As for the number of customers, we expect the size to be in the range of 500 to 1000 customers, at least in the initial validation tests.

- **Vehicle fleet:** a heterogeneous, multi-modal fleet will be considered. The precise composition of the vehicles cannot yet be determined. As for the size of the fleet, we estimate it to reach around 50 vehicles. However, depending on the results obtained and the definition of the use cases, this number may increase.

- **Objectives:** different combinations of at least the following three objectives will be considered: cost of operation, quality of service and emissions.

- **Constraints:** these will be mainly set by the use cases and user requirements to be defined in WP1. Based on the information available, some of the constraints that will be incorporated in these instances are:
  - Vehicle capacity
State of the art in optimization and machine learning algorithms applied to last mile logistic

- Time windows
- Maximum route duration
- Vehicle/order type compatibility (e.g. refrigerated products)
- Depot related constraints
- Labour conditions related constraints (e.g. working shift, breaks, etc.)

Finally, regarding the baseline for the optimisation algorithms, we will use the Large Neighbourhood Search approach as a reference, since this method and its variations are currently considered state-of-the-art in a large number of VRP variants.
5 Conclusion

Technology and methods used within SENATOR will be aligned on the present State-of-the-Art (SotA) document which will shape the design and development of the dynamic planning module of SENATOR. The present document does so by providing an in-depth review of current technologies, methods, and algorithms from the field of dynamic planning and machine learning. It does further sketch mathematical models used to abstract the processes to be optimized and most employed optimization techniques including optimization methods, heuristics and meta-heuristics which impact optimization for dynamic planning in logistics. For the field of machine learning, it reviews existing models and applications domains and strives to assess best practices from industrial use cases which include for example Support Vector Machines, Neural Networks, Long-Short Term Memory and alike. Along with the SotA on algorithms it provides a practical aspect relevant for integration of algorithms and methodology into real-world scenarios. These include but are not limited to an architectural viewpoint (Lambda Architecture), a data integration aspect (AI Data Pipeline), and a standard process cycle definition for handling and improving machine learning models. An outlook on the current market for software tools that provide a dynamic planning solution concludes the study.
6 References


[8] The APAMA Platform, Software AG, Product White Paper, 21.11.2016, (Online), https://content.cdnwrk.com/files/aT04MjQ2NjcmkdjOxJmlzc3VITmFTZT1oGUUtYXBbWEtcGxhdGZvcm0mY21kPWQmc2nPTJkJTA3MmVjMzE5MjY1ZTA3MGJhYTcNGMwMzl2ZjFj


State of the art in optimization and machine learning algorithms applied to last mile logistic

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 861540


[94] Zhao, M., Ji, S., & Wei, Z. (2020). Risk prediction and risk factor analysis of urban logistics to public security based on PSO-GRNN algorithm. Plos one, 15(10), e0238443


State of the art in optimization and machine learning algorithms applied to last mile logistic


This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 861540


State of the art in optimization and machine learning algorithms applied to last mile logistic

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 861540


State of the art in optimization and machine learning algorithms applied to last mile logistic

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 861540


https://doi.org/10.1109/ACCESS.2019.2957722

https://doi.org/10.1007/11890584_1

https://doi.org/10.1287/ijoc.2017.0756

https://doi.org/10.1016/j.eswa.2018.07.034

https://doi.org/10.1016/j.proeng.2014.12.461

https://doi.org/10.1007/s00170-018-2346-6

https://doi.org/10.1016/j.trc.2010.05.018


https://doi.org/10.1137/1.9780898718515.ch1

https://doi.org/10.1016/j.cor.2014.10.019

https://doi.org/10.1109/TSMC.2018.2861879

https://doi.org/10.1109/TCYB.2015.2409837

https://doi.org/10.1155/2018/1295485


