DRAFT VERSION

URBANITE

Supporting the decision-making in urban transformation with the use of disruptive technologies

Deliverable D4.2

Implementation of strategies and algorithms for data modelling and visualisations

Editor(s):	Maj Smerkol, Žiga Kolar, Zdenko Vuk, Inaki Olabarrieta
Responsible Partner:	Jožef Stefan Institute
Status-Version:	Final - v1.0
Date:	07. 04. 2022
Distribution level (CO, PU):	Public

Project Number:	GA 870338
Project Title:	URBANITE

Title of Deliverable:	Strategies and algorithms for data modelling and visualisations
Due Date of Delivery to the EC:	31/03/2022

Workpackage responsible for the Deliverable:	WP4 – Algorithms and simulation techniques for decision-makers
Editor(s):	Jožef Stefan Institute
Contributor(s):	JSI, TEC
Reviewer(s):	Heli Ponto (FVH)
Approved by:	All Partners
Recommended/mandatory readers:	WP3, WP5, WP6

Abstract:	This document presents an overview of methods for data modelling and visualisations and their use in the URBANITE domain. This deliverable is the result of tasks T4.1 and T4.4.
Keyword List:	Data analysis, data visualisation, data modelling, exploratory data analysis
Licensing information:	The license under this component is offered, is not decided yet, and depends on the licences of the different components integrated in it. This document is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein.

DRAFT VERSION

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	03/03/2022	Draft ToC	JSI
v0.2	05/03/2022	Updates to various sections	JSI
v0.3	06/03/2022	Updates to Section 2.5	TEC
v0.4	09/03/2022	Updates to the document structure	JSI
v0.5	14/03/2022	Updates to Sections 2, 3 and 4	JSI
v0.6	16/03/2022	Updates to various section	JSI
v0.7	17/03/2022	Submitted for internal review	JSI
v0.8	24/03/2022	Reviewed and sent to the editors	FVH
V0.9	31/03/2022	Final edits	JSI
V1.0	07/04/2022	Final version	TEC

DRAFT VERSION

Table of Contents

DRAFT VERSION

DRAFT VERSION

List of Figures

DRAFT VERSION

DRAFT VERSION

Terms and abbreviations

EC	European Commission
sqm	Square metre
GTFS	General transit feed specification
GUI	Graphical user interface
API	Application programming interface
PCA	Principal component analysis
SVD	Singular value decomposition
SOM	Self-organising map
ITS	Intelligent transportation system
OD	Origin destination (matrix)
POI	Point of interest

DRAFT VERSION

Executive Summary

This is deliverable D4.2 Implementation of strategies and algorithms for data modelling and visualisation. This deliverable is the outcome of tasks T4.1 Methods for exploratory data analysis and user interaction, and T4.4 Advanced visualisation methods. It describes the data modelling and visualisation methods and techniques implemented during the first 24 months of the project.

This deliverable was preceded by deliverable D4.1 Strategies and algorithms for data modelling and visualisations. The deliverable D4.1 presented a review of state-of-the-art methods and algorithms for data modelling and visualisations within the domain of the URBANITE project and discussed their applicability to the project. The methods and algorithms have been developed and are presented in this document. The developed methods and algorithms are being integrated in WP5.

The document contains:

- an introduction, containing further information about this deliverable and overview of the document structure.
- a section on data modelling methods, describing the objectives of the methods and the context with descriptions of data available, introduces the Orange data modelling framework, and lists and describes the methods developed grouped into projection methods, clustering methods, self-organising maps, and prediction methods;
- A section on developed analysis components, describing the components implementing specific analyses, traffic flow prediction, bike analysis, bike trajectories and public transport origin-destination matrix estimation.
- A section on data visualisation methods, discussing the objectives of the visualisation, development of map-based visualisations and describing methods in detail.
- A conclusion including a brief discussion of the developed methods, next steps, and final thoughts.
- And the references mentioned in the document.

This deliverable is a report on the developed and implemented methods for data modelling and visualisation, supporting the final users in performing exploratory data analysis of heterogeneous data including interactive analysis and visualisations. The deliverable also overviews the framework for user interaction with the harvested data, using a visual programming machine learning and data mining tool, Orange.

Some of the methods described here are also used by other components of the URBANITE system, such as the Decision support system and Policy simulation system, described in deliverables D4.3 and D4.4, due M24. The final versions of the components will be presented in deliverable D4.6, due at the end of tasks T4.1, T4.2, T4.3, and T4.4 in M30.

DRAFT VERSION

1 Introduction

Deliverable D4.2 presents an overview of methods for data modelling, analysis and visualisations that are applicable to the URBANITE domain and have been used in the project.

This document is part of work package WP4 "Algorithms and simulation techniques for decision-makers" and is an outcome of tasks T4.1 "Methods for Exploratory Data Analysis and User Interaction" and T4.4 "Advanced visualisations methods".

It presents the methods for modelling, analysis, and visualisations of data, which have been used in the development or implemented for various analysis and data processing. The methods are discussed in context of urban mobility and policy support, with examples of use within the URBANITE platform.

1.1 About this deliverable

This document reports the data modelling methods and data visualisation methods that are used to process the data, present the data and the results, or provided as tools to the end user. The methods identified are described and their use in the context of the project justified.

The methods used are based on the content of deliverable D4.1, which overviewed the state-of-the-art methods and their applicability to the project. Additional research has been performed in some cases as the work progressed and several specific analyses have been done during the refinement of the use cases.

Furthermore, this document presents the framework for user interaction with the data and visualisations using Orange, a toolset for data analysis and machine learning.

This document covers the work planned out in deliverable D4.1, performed up M24. Some of the methods presented are still work in progress and are described as such.

1.2 Document structure

This document is organised into the following sections:

- "Introduction" explains the rationale of this document and explains the document structure in more detail.
- "Data modelling methods" contains the overview of methods developed for exploratory data analysis, including a discussion of their use in the context of the URBANITE solution, and the framework for interaction with the data and visualisations, as well as specific analysis components developed.
- "Data visualisation methods" provides the overview of visualisation methods in the domain of urban mobility. It includes discussion of their role in the URBANITE solution with examples.
- Then, for each of the main modules, the dedicated sections 5, identify their installation instructions, a brief user manual, licensing information and the repository URL for downloading the source code.
- "Conclusion" presents the overview of the work performed and next steps planned until the end of the project.

DRAFT VERSION

2 Data modelling methods

This section describes motivation for using relevant data modelling methods, explaining the rationale for usage and evaluation of the methods. The following subsections cover the objectives of data modelling in the context of the URBANITE project, describe the harvested data and other relevant data, and introduce the Orange data modelling framework, followed by discussion of projection methods, clustering methods, self-organising map, and prediction and regression methods. Finally, components developed for specific analysis are covered.

2.1 Objectives

The main objectives of the data modelling tools are to enable the policy makers to interactively explore available heterogenous data, discover related data attributes that may indicate causal relations, and enable the deeper understanding of complex and so-called 'wicked' mobility-related problems [1]

To facilitate interactive exploratory data analysis, the data modelling methods should be simple to use for non-technical users, allow the use of advanced data modelling techniques appropriate for use on different data sets, and support interactive visualisations during the analysis process.

DRAFT VERSION

2.2 Context

Harvested data consists of attributes, allowing for selection, filtering, and sorting of the data:

- model (Available values: trafficFlowObserved, daySpecification, calendar, airQualityObserved, weatherObserved, event, censusObserved, originDestinationMatrix) describes the data models of different data sets harvested.
- city (Available values: Bilbao, Messina, Helsinki, Amsterdam) informs the platform which of the pilot cases the URBANITE platform is deployed for,
- filters (Different data model fields)
- returnFields (Comma separated list of names (text) of Data Model fields to be returned. If not set, all fields will be returned)
- limit (Number of documents to retrieve)
- sort (Sort results by date in ascending or descending order).

Harvested data includes datasets including:

- trafficFlowObserved: time series data of traffic counts at different locations,
- airQualityObserved: time series data of air quality measurements at different locations,
- weatherObserved: time series data about weather, including attributes such as temperature, precipitation, and atmospheric pressure,
- censusObserved: provides anonymized data about persons in the city,
- originDestinationMatrix: origin destination matrices provide data about movements between city districts,
- daySpecification: different attributes of specific days by date, such as whether it is a workday, public holiday etc.

Each of the pilot cities provides certain data types that are not common to all of them. An overview of all data sources provided by different use cases is out of the scope of this document, however as an example, the additional data provided in the Messina use case are described. The Messina use case specifically is interesting as the use case developed new data access capabilities.

Data for Messina is divided into many different categories: areas of districts, district populations, vehicle stats, mobility, average values of homes in a district, number of students at different education levels, points of interest, bus stops and cycling paths. Data layers, describing statistics at district level include:

- number of residents,
- building outlines on a map,
- number of families,
- percentage of men,
- percentage of women.

Statistics about population in Messina at the city level include:

- total number of inhabitants,
- population by gender,
- population by age,
- population by district (residents, families, males, females, by age),
- education levels in the city (percentage),
- working people (percentage of employment, average income, average retirement pension).

Among vehicle statistics there are:

- total cars, motorcycles, average number of cars per person, registrations by gender,
- fuel types classification,
- air pollutant emissions by Euro emission standard.

Systematic mobility is represented by:

- work and study, describing the share of trips made for work or for education,
- internal and external movements, comparing the shares of trips made inside the city with the share of trips originating from outside the city.

These data sources describe the movements of people in more general terms, e.g. share of trips made for educational purposes and work purposes and the share of movement within the city and external movements out of the city, into the city or through the city.

Additional statistical information about the districts include:

- total number of inhabitants,
- market value (min, max in €/sqm),
- rent value (min, max in €/sqm per month).

Other data available describes schools, e.g. high school students includes the number of students per school, locations of interest, including business location, schools and other types of facilities are represented by name and location and bus stops, map the locations of bus stops derived from public transport data available as GTFS. Finally, cycling paths are described by name, description and LineString of cycling paths in the city in GeoJSON format.

2.3 Orange data modelling

Orange[1] is a free, open-source software for data visualisation, machine learning, data mining, and data analysis. It has been developed by the University of Ljubljana from Slovenia [2].

[1] More information about Orange is available on the official website: https://orangedatamining.com.

DRAFT VERSION

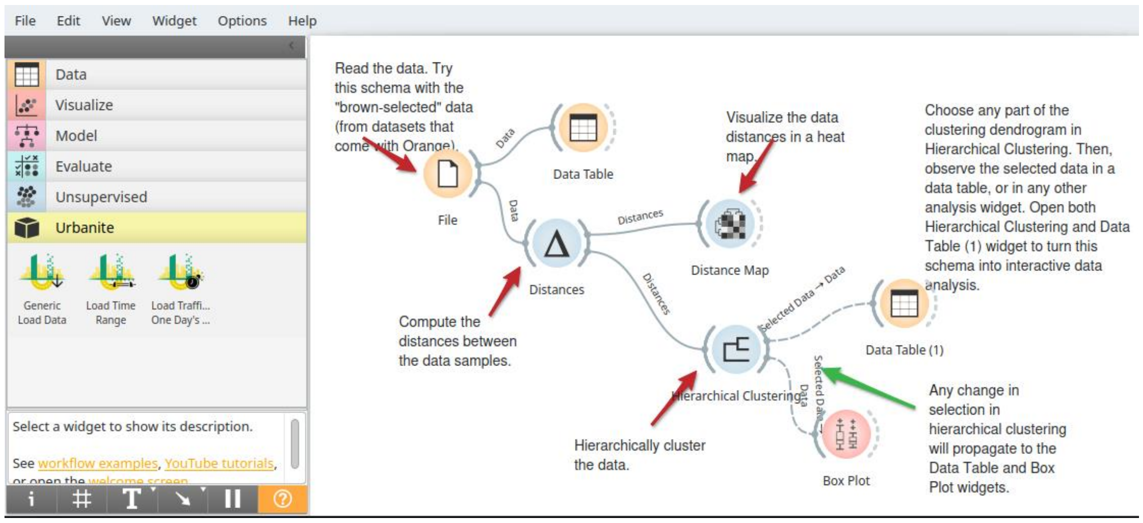
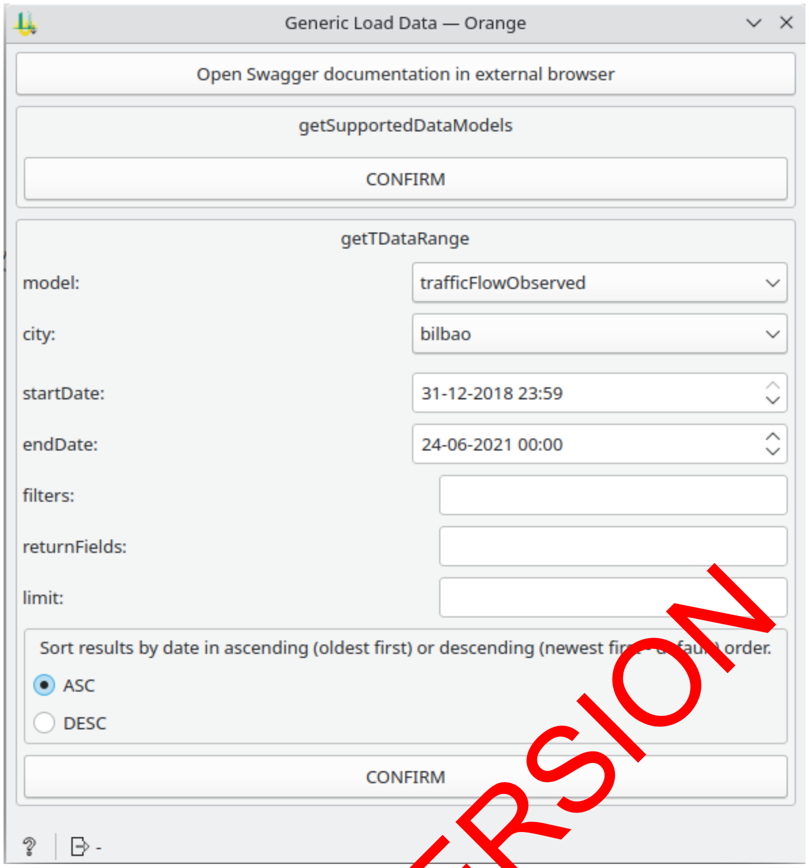

File
Edit
View
Widget
Options
Help
Data
Visualize
Model
Evaluate
Unsupervised
Urbanite
Generic Load Data
Load Time Range
Load Traffi... One Day's ...
Select a widget to show its description.
See workflow examples, YouTube tutorials,
Read the data. Try this schema with the "brown-selected" data (from datasets that come with Orange).
Data
Data Table
File
Data
Distances
Distances
Visualize the data distances in a heat map.
Distance Map
Choose any part of the clustering dendrogram in Hierarchical Clustering. Then, observe the selected data in a data table, or in any other analysis widget. Open both Hierarchical Clustering and Data Table (1) widget to turn this schema into interactive data analysis.
Compute the distances between the data samples.
Distances
Selected Data → Data
Data Table (1)
Hierarchical Clustering
Selected Data
Data
Any change in selection in hierarchical clustering will propagate to the Data Table and Box Plot widgets.
Hierarchically cluster the data.
Box Plot

Figure 1 Example of a workflow in Orange

It allows for visual programming, without required programming skills, with a few mouse clicks. It is very useful for peeking into data and interactive analysis. It implements data mining in Python in addition to a GUI. It offers a multitude of machine learning algorithms. Its workflow consists of linking components (widgets), as shown in Figure 1. Many widgets are provided, and new ones can be developed by the user by programming them in Python and Qt. It comes with some predefined datasets as input data. Input data can be merged from different datasets with a respective widget.

Until now we have built three widgets for Urbanite. All three of the widgets we built serve to load and visually analyse the URBANITE data. This data comes from our partner's servers through API calls. Data from these APIs comes in a form of various values in conjunction with one or more datetime value(s).

DRAFT VERSION



Generic Load Data — Orange

Open Swagger documentation in external browser

getSupportedDataModels

CONFIRM

getTDataRange

model:	trafficFlowObserved
city:	bilbao
startDate:	31-12-2018 23:59
endDate:	24-06-2021 00:00
filters:	
returnFields:	
limit:	

Sort results by date in ascending (oldest first) or descending (newest first - default) order.

[✓ ASC] DESC

CONFIRM

DRAFT VERSION

Figure 2 One of our custom-built widgets. The widget allows users to access the harvested data and perform further operations on the data.

Orange uses the concept of a workflow to present complex processes, performing analyses, machine learning or data mining tasks. In Figure 3 an example of a workflow is shown, created by connecting three widgets. This workflow loads data from a file into a data table, from which only certain columns (attributes) are selected for further analysis. This utilises a widget that reads the input file, a widget that constructs the data table from the raw data and a widget that allows for selection of the columns. The two lines, continuing to the right from the last widget, tell us that at least two more widgets exist processing the selected data.

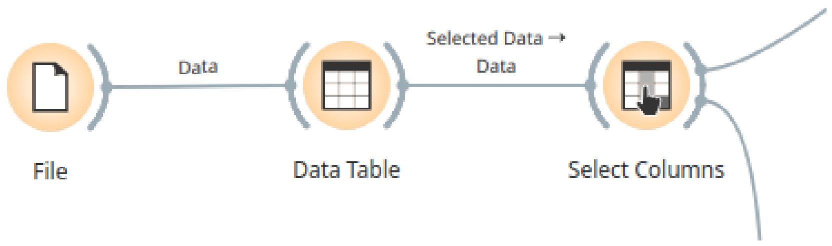
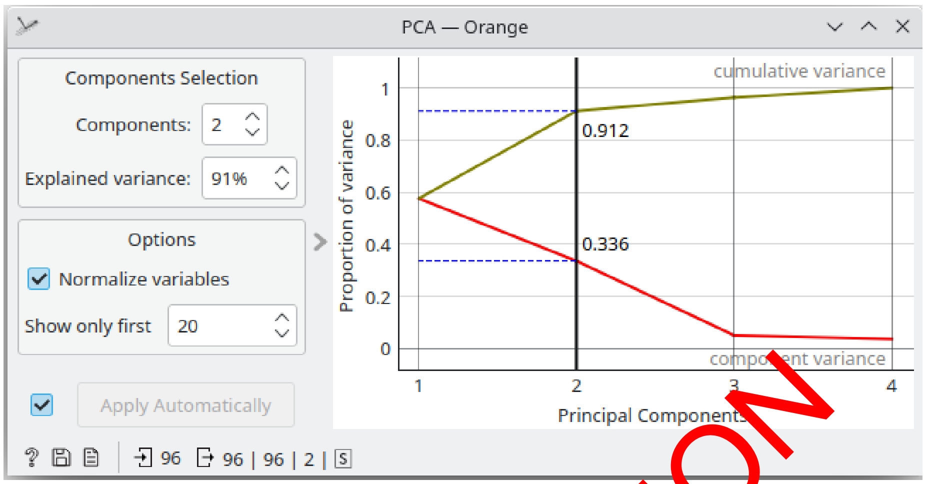


Figure 3: Three widgets connected in a pipeline create a workflow. Each widget performs an operation and triggers the next one when the results are ready.

One of these two widgets performs PCA. The widget's UI and results of principal component analysis are shown in Figure 4. With the results satisfactory, we proceed to pipe the output to a k-Means widget, shown in Figure 5, and furthermore to a Scatter Plot widget which plots the data points in the space of first two principal components, as shown in Figure 6. This extended workflow is shown in Figure 7.



PCA — Orange

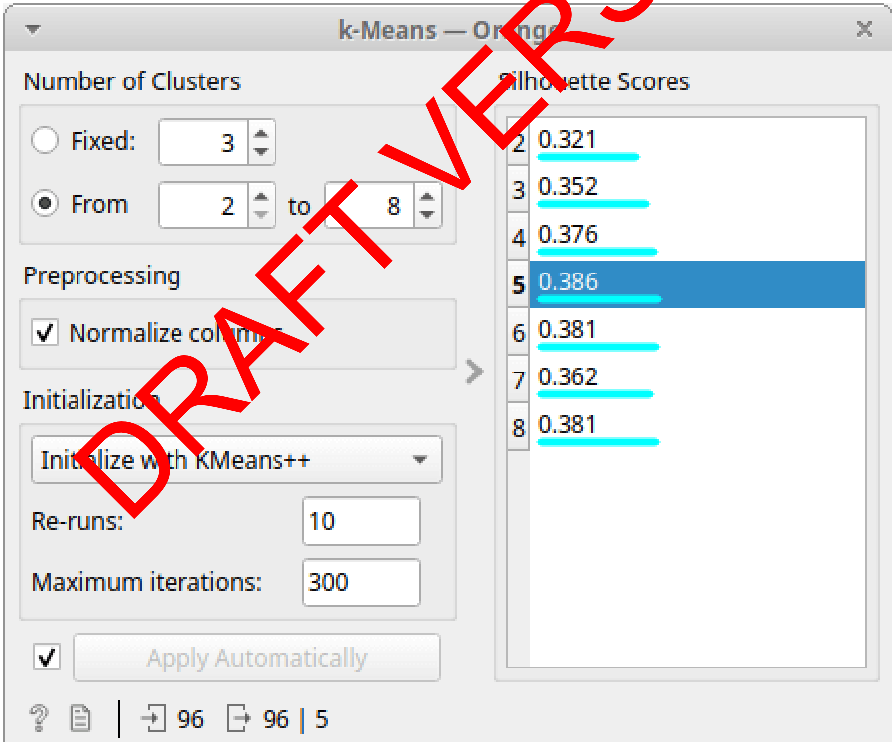
Components Selection
Components: 2
Explained variance: 91%

Options
☑ Normalize variables
Show only first 20

☑ Apply Automatically

96 | 96 | 96 | 2

Figure 3 PCA result, showing the variance explained by different numbers of components.



k-Means — Orange

Number of Clusters
○ Fixed: 3
● From 2 to 8

Preprocessing
☑ Normalize columns

Initialization
Initialize with KMeans++
Re-runs: 10
Maximum iterations: 300

☑ Apply Automatically

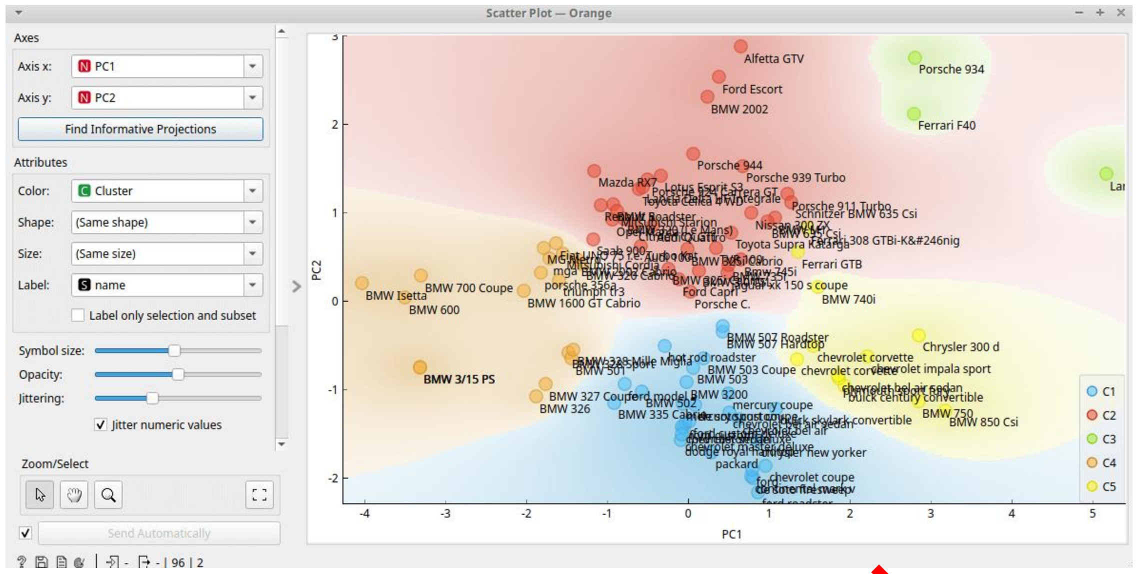
96 | 96 | 5

Silhouette Scores

2	0.321
3	0.352
4	0.376
5	0.386
6	0.381
7	0.362
8	0.381

Figure 4 K-Means widget, showing silhouette scores for different numbers of clusters.

DRAFT VERSION



Axis x: PC1
Axis y: PC2
Color: Cluster
Shape: (Same shape)
Size: (Same size)
Label: name
✓ Jitter numeric values

Figure 5 Final result, displayed with Scatter Plot. The dataset used as an example represents cars by their technical specifications. Similarly, within the URBANITE project similar analysis was performed to cluster vehicles based on air pollutant emissions.

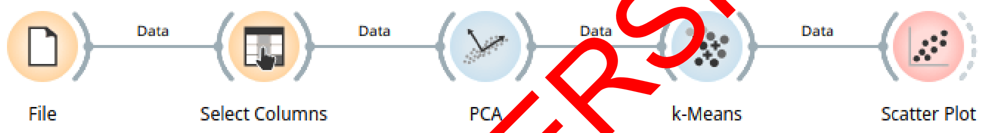


Figure 7: Workflow producing the results, shown in Figure 6.

DRAFT VERSION

2.4 Projection methods

Data projection methods are mathematical operations that transform the data in some space to a subspace. Several projection methods are appropriate for dimensionality reduction. Why would we want to reduce the number of dimensions? In predictive analytics, more columns normally mean more time required to build models and score data. If some columns have no predictive value, this means wasted time, or worse, those columns contribute noise to the model and reduce model quality or predictive accuracy.

Dimensionality reduction can be achieved by simply dropping attributes, for example, those that may show up as collinear with others or identified as not being particularly predictive of the target as determined with an attribute importance ranking technique. But it can also be achieved by deriving new attributes based on linear combinations of the original attributes. In both cases, the resulting transformed data set can be provided to machine learning algorithms to yield faster model build times, faster scoring times, and more accurate models.

In this subsection most common approaches for dimensionality reduction are described and examples of use in URBANITE given.

2.4.1 Principal Component Analysis - PCA

The principal components of a collection of points in a real coordinate space are a sequence of p unit vectors, where the i-th vector is the direction of a line that best fits the data while being orthogonal to the first i-1 vectors. Here, a best-fitting line is defined as one that minimises the average squared distance from the points to the line. These directions constitute an orthonormal basis in which different individual dimensions of the data are linearly uncorrelated. PCA is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest. PCA transforms a high number of features into a low number of uncorrelated PCA components, where each component is a linear combination of the original features.

PCA is used as a dimensionality reduction algorithm before running machine learning models on data, e.g. machine learning models used in the URBANITE DSS [3].

2.4.2 Singular Value Decomposition - SVD

SVD is one of several techniques that can be used to reduce the dimensionality, i.e., the number of attributes, of a data set. In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix. It generalises the eigendecomposition of a square normal matrix with an orthonormal eigenbasis to any m x n matrix. It is related to polar decomposition. While SVD can be used for dimensionality reduction, it is often used in digital signal processing for noise reduction, image compression, and other areas.

Within the URBANITE platform SVD is used to remove noise from time series data, e.g. traffic counts.

2.4.3 Factor analysis

Factor analysis is a statistical method used to describe variability among observed, correlated variables in terms of a potentially lower number of unobserved variables called factors. For example, it is possible that variations in six observed variables mainly reflect the variations in two unobserved (underlying) variables. Factor analysis searches for such joint variations in response to unobserved latent variables. The observed variables are modelled as linear combinations of the potential factors plus "error" terms, hence factor analysis can be thought of as a special case of errors-in-variables models.

Factor analysis is used to analyse the relation between weather and traffic patterns during the traffic simulation calibration dataset creation.

2.4.4 Independent component analysis - ICA

In signal processing, independent component analysis is a computational method for separating a multivariate signal into additive subcomponents. This is done by assuming that the subcomponents are, potentially, non-Gaussian signals and that they are statistically independent from each other. ICA is a special case of blind source separation.

A common example application is the "cocktail party problem" of listening in on one person's speech in a noisy room. Similarly, using ICA, traffic flows from different sources can be analysed.

2.4.5 Linear and quadratic discriminant analysis with shrinkage

Linear, quadratic discriminant analysis can be used to perform supervised dimensionality reduction, by projecting the input data to a linear subspace consisting of the directions which maximise the separation between classes. The dimension of the output is necessarily less than the number of classes, so this is in general a rather strong dimensionality reduction, and only makes sense in a multiclass setting.

This method has been applied to the machine learning methods used in the URBANITE DSS [3] to improve generalisation of the models.

DRAFT VERSION

2.5 Clustering methods

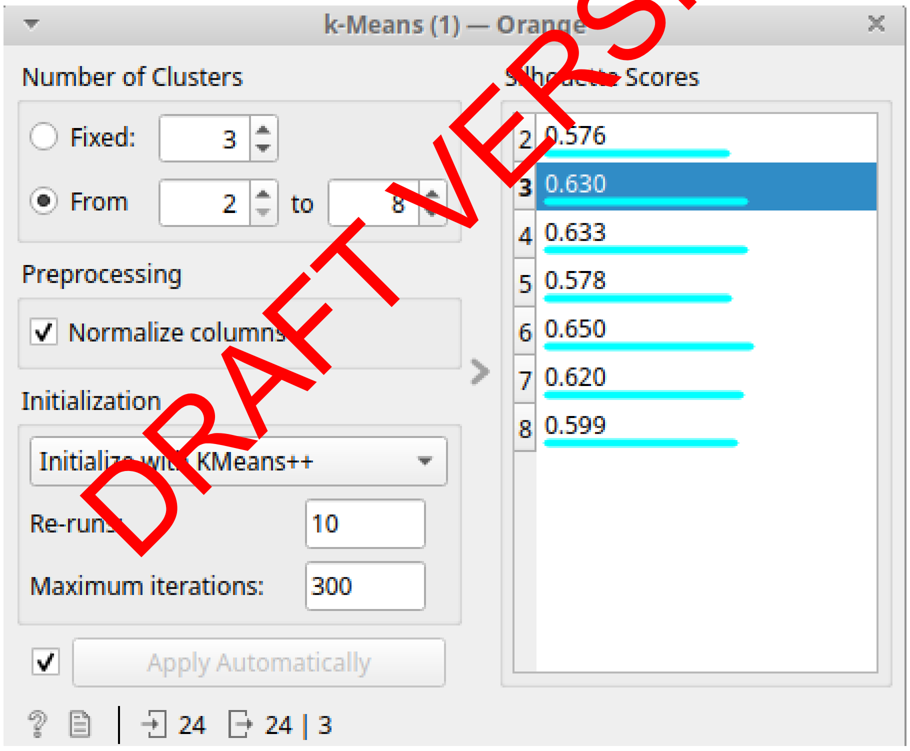
In this subsection several clustering methods are described and their applications within URBANITE.

2.5.1 K-Means

One of the common clustering methods is K-means. The k-means algorithm combines objects into a low number of clusters. Each cluster is represented with the cluster centre that is calculated as the mean of the objects that belong to that cluster. An object belongs to the cluster with the nearest centre. As a result, the k-means clustering divides the space of objects into Voronoi cells, one for each cluster. The problem is computationally difficult (NP-hard), however, efficient heuristic algorithms converge quickly to a local optimum. The algorithm is appropriate for bigger data sets. An example of K-means clustering can be seen in Figure 8, where there are 5 clusters, each shown with a different colour.

Here we plot traffic flow observed data from one day from Bilbao and we get the graph as in Figure 9. X axis shows the time of day from midnight to midnight. This plot tells us that in the morning and in the afternoon, traffic flow observed (traffic volume), is at its high. In the middle of the day it is average, while during night hours it's low.

DRAFT VERSION



k-Means (1) — Orange

Number of Clusters

Fixed: 3

[✓ From] 2 to 8

Preprocessing

✓ Normalize columns

Initialization

Initialize with KMeans++

Re-runs: 10

Maximum iterations: 300

✓ Apply Automatically

24 24 | 3

Silhouette Scores

2	0.576
3	0.630
4	0.633
5	0.578
6	0.650
7	0.620
8	0.599

Figure 6 K-Means widget. The displayed silhouette scores show that 6 clusters represent the data best.

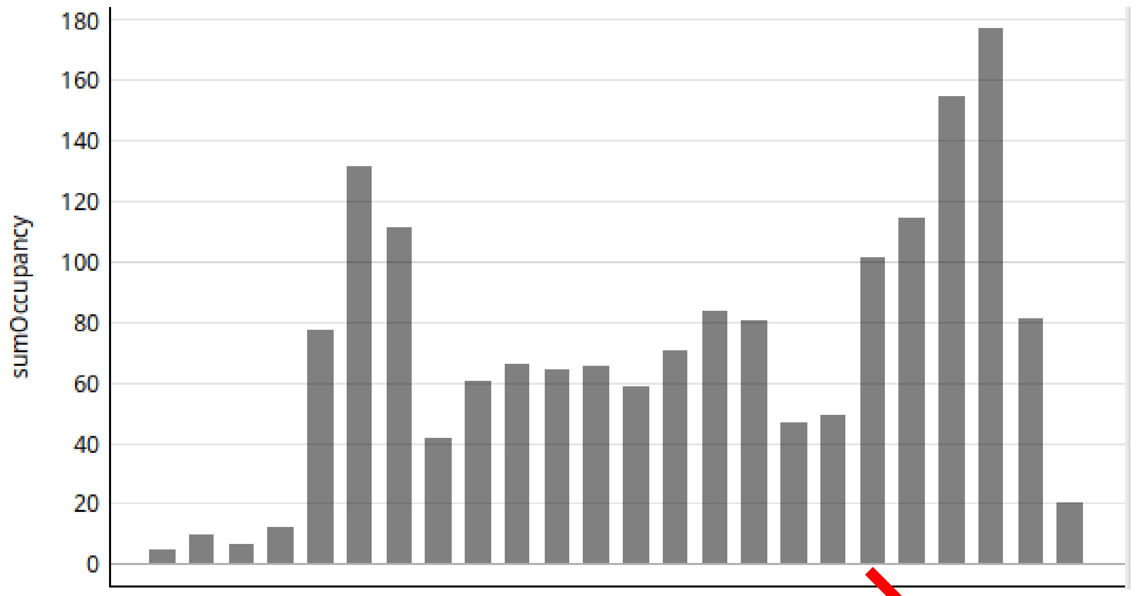


Figure 7 Bar plot, showing traffic volume at a specific location over one day.

DRAFT VERSION

For this data Orange's k-Means suggests that six would be the best number of clusters as can be deduced from Figure 9 as we can see from the Silhouette Scores, but for the sake of simplicity we will choose to have three clusters.

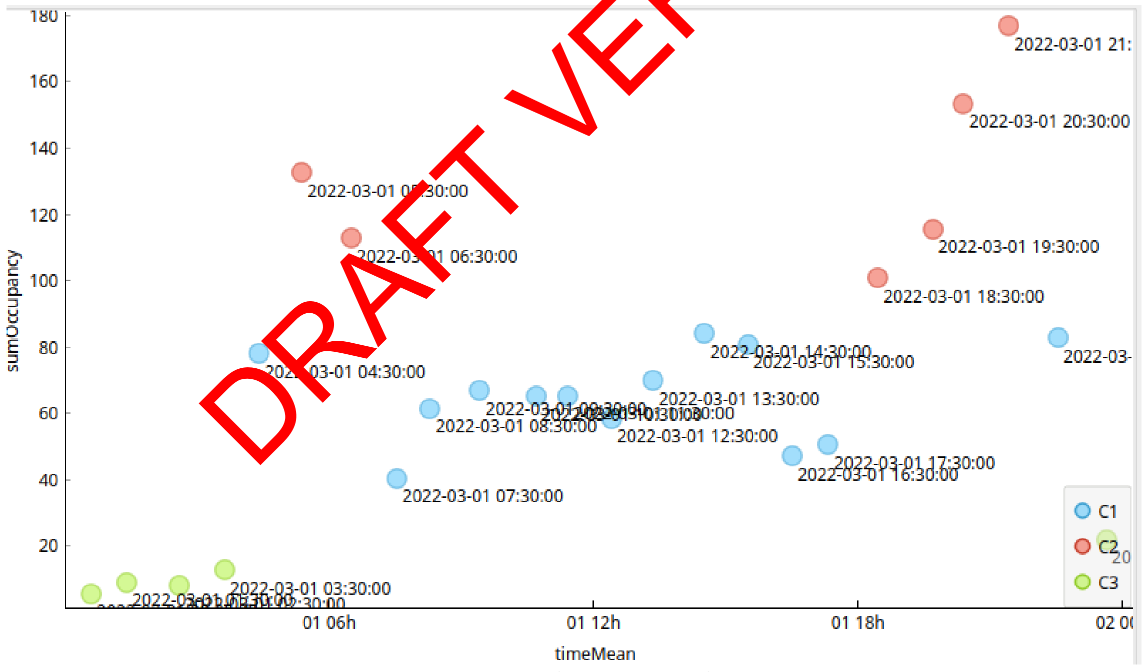


Figure 8 Resulting scatter plot.

In the results plot, shown in Figure 10, we can see that the results the algorithm yielded make sense. It made three groups, one composed of samples from the night's very low traffic (green), another from midday samples with about average values (blue), and another one from traffic spikes from the morning and evening (red).

2.5.2 Affinity propagation

The second clustering method is affinity propagation. Affinity propagation is a clustering algorithm based on the concept of "message passing" between data points. Unlike clustering algorithms such as k-means or k-medoids, affinity propagation does not require the number of clusters to be determined or estimated before running the algorithm. Similar to k-medoids, affinity propagation finds "exemplars", members of the input set that are representative of clusters. It considers all data points as potential cluster centres and identifies a set of cluster centres that represent the dataset. The algorithm needs two pieces of information. First are the similarities between data points and second are the preferences. Similarity s(i,k) indicates how well the data point k is suited to be the exemplar for data point i. Preferences (k,k) influence the number of clusters. Affinity propagation recursively transmits real-valued messages along edges of the network until a good set of examples and clusters emerges. It works better with smaller data sets. It is unbalanced and is applicable to time series.

Affinity propagation is used to cluster traffic flow data together with preprocessed related data about weather, football match schedules and other data.

2.5.3 Spectral clustering

The third clustering method is spectral clustering. In multivariate statistics, spectral clustering techniques make use of the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions. The similarity matrix is provided as an input and consists of a quantitative assessment of the relative similarity of each pair of points in the dataset. This method is especially useful for clustering air pollutant emission data and other time series data.

The air pollutant emissions are represented as time series data from different locations in the city. The data is clustered based on shapes of the time series data, e.g. locations with high air pollutant emissions in the morning are in one cluster while locations with high emissions in the evening are in a different cluster. This allows for identification of locations that are problematic only under specific traffic conditions.

2.5.4 Hierarchical clustering

Finally, the last method for clustering similar objects together is hierarchical clustering. It starts with each object being an independent cluster. Afterwards, it combines the two most similar clusters together and repeats this step until only one cluster remains. The resulting representation of the hierarchically organised clusters is a dendrogram, where each cluster is represented by a horizontal line. The line lengths represent the differences between the clusters---the longer the distance, the more different the clusters, the smaller the distance, the more similar clusters. Hierarchical clustering is appropriate for population segmentation.

Here we are showing an example of this type of clustering on one day of Bilbao traffic count data. Before channelling the data into the Hierarchical Clustering widget, we will need first to pass it to Distances widget, as can be seen in Figure 1. For explanatory purposes, we selected two subsets on the dendrogram, C1 (blue) and a very small set of data C2 (red). Both of them differ largely from all the rest of the data.

DRAFT VERSION

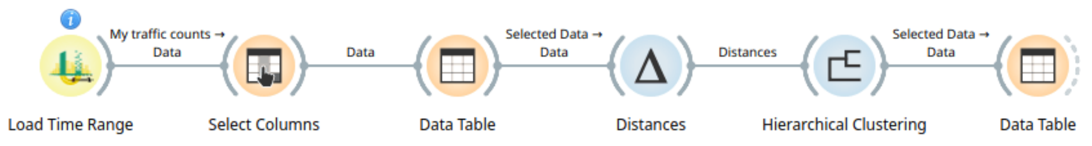


Figure 11: Workflow starts with loading the selected data by time range, selection of appropriate columns, tabularization of the data, calculation of distances between data points, hierarchical clustering step, and tabularization for presentation of results.

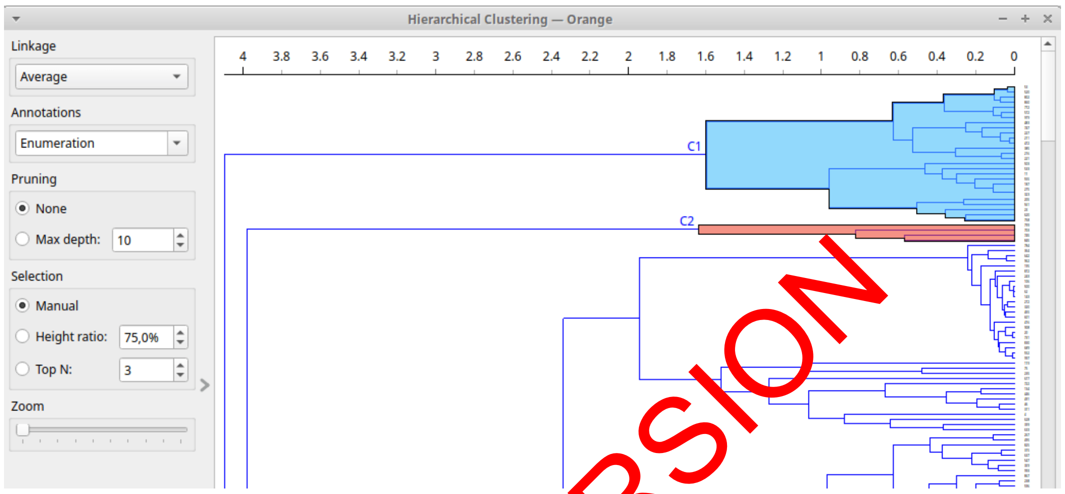
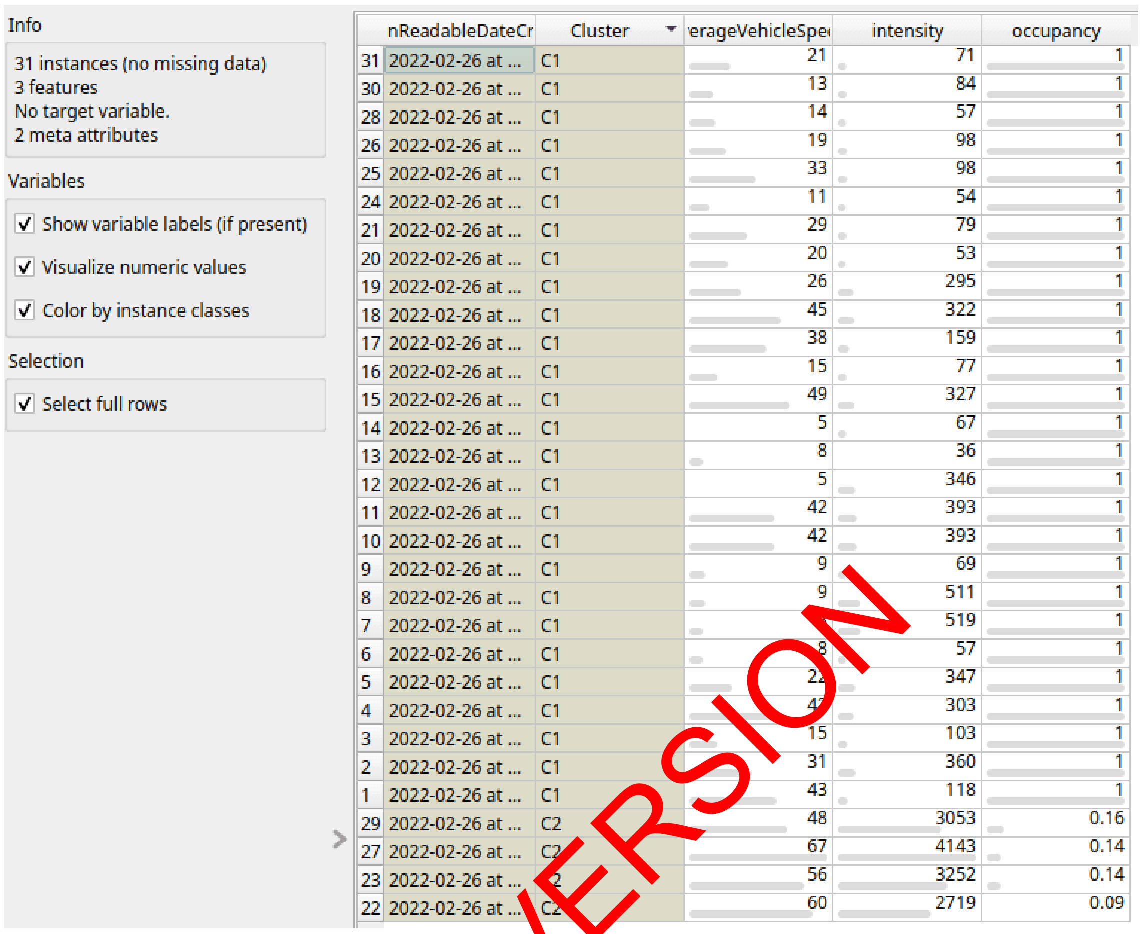


Figure 9 Visualisation of dendrogram displaying the hierarchical clusters.

Upon examining C1 and C2 in the Data Table widget, we can see that C1 is special because it has a high value for occupancy. The C2 samples are special because they have a very large value for intensity (we took a look at the other sample's data's intensity value in order to be able to determine that).

DRAFT VERSION



Info

31 instances (no missing data)
3 features
No target variable.
2 meta attributes

Variables

☑ Show variable labels (if present)
☑ Visualize numeric values
☑ Color by instance classes

Selection

☑ Select full rows

	nReadableDateCr	Cluster	erageVehicleSpe	intensity	occupancy
31	2022-02-26 at ...	C1	21	71	1
30	2022-02-26 at ...	C1	13	84	1
28	2022-02-26 at ...	C1	14	57	1
26	2022-02-26 at ...	C1	19	98	1
25	2022-02-26 at ...	C1	33	98	1
24	2022-02-26 at ...	C1	11	54	1
21	2022-02-26 at ...	C1	29	79	1
20	2022-02-26 at ...	C1	20	53	1
19	2022-02-26 at ...	C1	26	295	1
18	2022-02-26 at ...	C1	45	322	1
17	2022-02-26 at ...	C1	38	159	1
16	2022-02-26 at ...	C1	15	77	1
15	2022-02-26 at ...	C1	49	327	1
14	2022-02-26 at ...	C1	5	67	1
13	2022-02-26 at ...	C1	8	36	1
12	2022-02-26 at ...	C1	5	346	1
11	2022-02-26 at ...	C1	42	393	1
10	2022-02-26 at ...	C1	42	393	1
9	2022-02-26 at ...	C1	9	69	1
8	2022-02-26 at ...	C1	9	511	1
7	2022-02-26 at ...	C1	[illegible]	519	1
6	2022-02-26 at ...	C1	8	57	1
5	2022-02-26 at ...	C1	22	347	1
4	2022-02-26 at ...	C1	47	303	1
3	2022-02-26 at ...	C1	15	103	1
2	2022-02-26 at ...	C1	31	360	1
1	2022-02-26 at ...	C1	43	118	1
29	2022-02-26 at ...	C2	48	3053	0.16
27	2022-02-26 at ...	C2	67	4143	0.14
23	2022-02-26 at ...	C2	56	3252	0.14
22	2022-02-26 at ...	C2	60	2719	0.09

DRAFT VERSION

Figure 10 Viewing selected subsets of dendrogram.

2.6 Self-Organising Map - SOM

A self-organising map is an unsupervised machine learning technique used to produce a low-dimensional (typically two-dimensional) representation of a higher dimensional data set while preserving the topological structure of the data. For example, a data set with p variables measured in n observations could be represented as clusters of observations with similar values for the variables. A SOM is a type of artificial neural network but is trained using competitive learning rather than the error-correction learning (e.g., backpropagation with gradient descent) used by other artificial neural networks.

Self-organising maps, like most artificial neural networks, operate in two modes: training and mapping. First, training uses an input data set (the "input space") to generate a lower-dimensional representation of the input data (the "map space"). Second, mapping classifies additional input data using the generated map.

In most cases, the goal of training is to represent an input space with p dimensions as a map space with two dimensions. Specifically, an input space with p variables is said to have p dimensions. A map space consists of components called "nodes" or "neurons," which are arranged as a hexagonal or rectangular grid with two dimensions. The number of nodes and their arrangement are specified beforehand based on the decision maker goals of the analysis and exploration of the data.

Each node in the map space is associated with a "weight" vector, which is the position of the node in the input space. While nodes in the map space stay fixed, training consists in moving weight vectors toward the input data (reducing a distance metric such as Euclidean distance) without spoiling the topology induced from the map space. After training, the map can be used to classify additional observations for the input space by finding the node with the closest weight vector (smallest distance metric) to the input space vector.

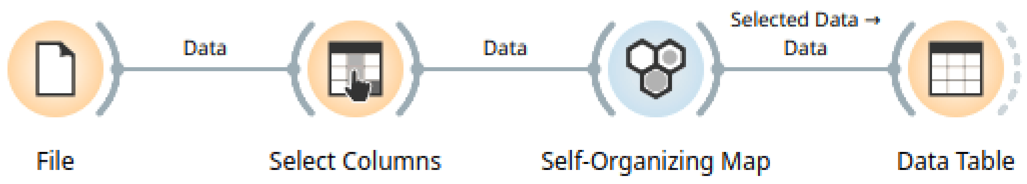
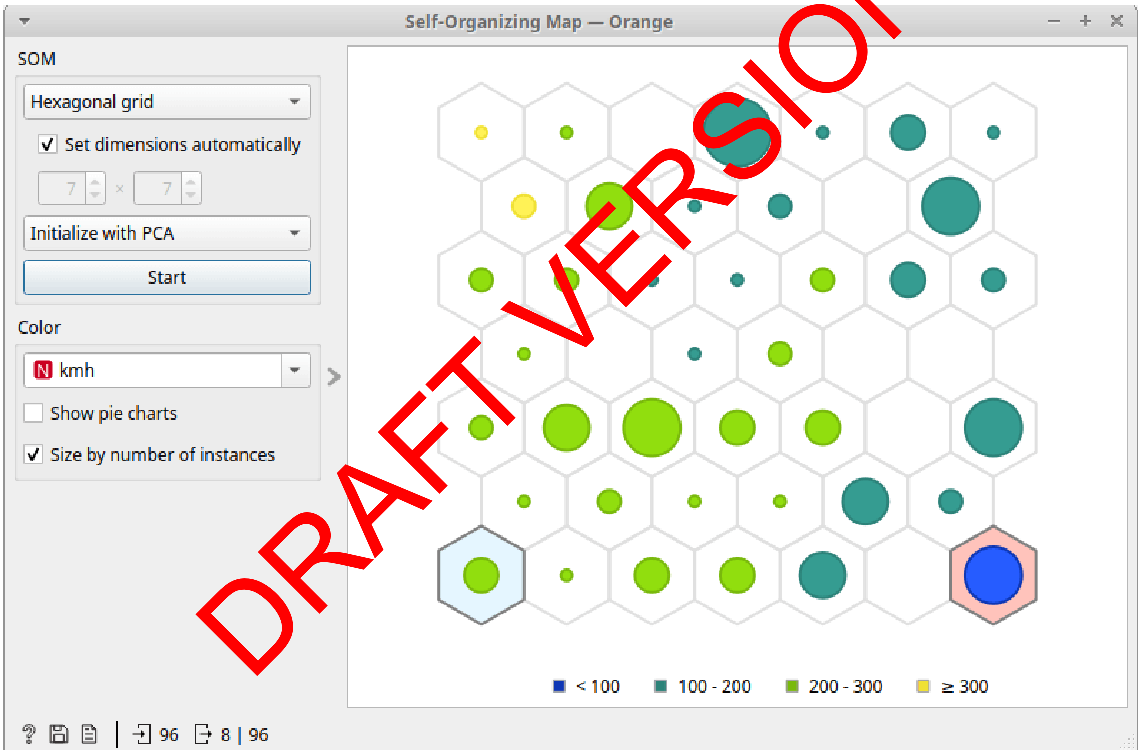


Figure 11 Example of a workflow.


DRAFT VERSION

Self-Organizing Map — Orange

SOM

Hexagonal grid

✓ Set dimensions automatically

7 × 7

Initialize with PCA

Start

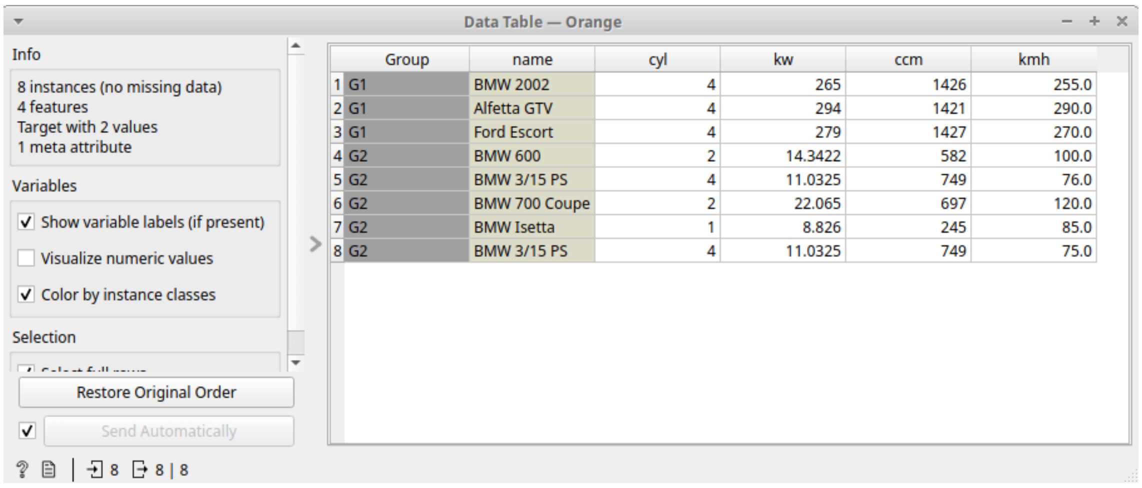
Color

N kmh

☐ Show pie charts

✓ Size by number of instances

Figure 12 SOM's result on a test data set. The colour shows a vehicle's max speed. We can see that the slowest vehicle, marked with blue, is located as far as possible from the fastest vehicle, marked with yellow.


Data Table — Orange

Info

8 instances (no missing data)
4 features
Target with 2 values
1 meta attribute

Variables

✓ Show variable labels (if present)
Visualize numeric values
✓ Color by instance classes

Selection

Restore Original Order
✓ Send Automatically

	Group	name	cyl	kw	ccm	kmh
1	G1	BMW 2002	4	265	1426	255.0
2	G1	Alfetta GTV	4	294	1421	290.0
3	G1	Ford Escort	4	279	1427	270.0
4	G2	BMW 600	2	14.3422	582	100.0
5	G2	BMW 3/15 PS	4	11.0325	749	76.0
6	G2	BMW 700 Coupe	2	22.065	697	120.0
7	G2	BMW Isetta	1	8.826	245	85.0
8	G2	BMW 3/15 PS	4	11.0325	749	75.0

8 | 8 | 8

Figure 13 Inspecting the two groups we selected. A clear difference between the two groups is the vehicle's power output and engine displacement volume.

2.7 Prediction/regression methods, or classification methods

In statistical modelling, regression methods are a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome' or 'response' variable) and one or more independent variables (often called 'predictors', 'covariates', 'explanatory variables', 'attributes', or 'features'). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables.

The prediction and regression methods are used in the URBANITE project for example to predict future traffic flows, estimate the expected values of scenario KPIs before simulating them, while classification methods are used to find congested road segments.

2.7.1 Linear regression

One of the most widely known modelling techniques is linear regression. It is represented by the following equation

$$Y = a + b * X + e$$

where a is intercept, b is slope of the line and e is an error term. This equation can be used to predict the value of target variable based on given predictor variable(s). It establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line). Regression line is the most common method used for fitting a regression line. It calculates the best-fit line for the observed data by minimising the sum of the squares of the vertical deviations from each data point to the line. Because the deviations are first squared, when added, there is no cancelling out between positive and negative values.

2.7.2 Logistic regression

Logistic regression is one of the types of regression analysis technique, which is used when the dependent variable is discrete. Example: 0 or 1, true or false, etc. This means the target variable can have only two values, and a sigmoid curve denotes the relation between the target variable and the independent variables. Logit function is used in Logistic Regression to measure the relationship between the target variable and independent variables. Below is the equation that denotes logistic regression:

DRAFT VERSION

$$logit(p) = ln(p/1-p) = b_0 + b_1 * X_1 + b_2 * X_2 + b_3 * X_3 + \ldots + b_k * X_k$$

where p is the probability of occurrence of the feature. For selecting logistic regression as the regression analyst technique, it is required that the size of data is large with the almost equal occurrence of values to come in target variables. Also, there should be no multicollinearity, which means that there should be no correlation between independent variables in the dataset.

Logistic regression can be used for **classification**. Classification is the process of recognizing, understanding, and grouping objects into preset categories or "sub-populations." Using pre-categorized training datasets, machine learning programs use a variety of algorithms to classify future datasets into categories.

Classification algorithms in machine learning use input training data to predict the likelihood that subsequent data will fall into one of the predetermined categories. One of the prominent uses of logistic regression in the URBANITE context is the classification of road segments into congested and not congested.

2.7.3 K-nearest neighbours

K-nearest neighbours (k-NN) is a pattern recognition algorithm that uses training datasets to find the k closest relatives in future examples. When k-NN is used in classification, the sample in classified with the category of its nearest neighbour. If k = 1, then it is classified the same as the nearest 1 sample, while when K>1, the sample is classified by a plurality poll of its neighbours.

This method is used to create different traffic simulation calibration datasets. The traffic volume data is enriched with corresponding data such as weather, ferry arrival schedules and similar data. K-NN is used to cluster this complex data into different clusters, such as rainy days with less traffic, sunny days with more cyclists, or days with specifically heavy traffic from and to the harbour.

2.7.4 Decision tree

A decision tree is a supervised learning algorithm for the classification problems, as it's able to order classes on a precise level. It works like a flow chart, separating data points into two similar categories at a time from the "tree trunk" to "branches," to "leaves," where the categories become more finitely similar. This creates categories within categories, allowing for organic classification with limited human supervision.

After the decision tree is inferred by fitting the learning data, more important branches (those that contain most training samples) can be extracted and converted to rules. This approach was taken in the first version of the URBANITE DSS. We trained a decision tree to predict the total emitted CO^2 using a vector representation of mobility policies and extracted representative branches. An example of a resulting rule is "IF share_of_bike_trips > 0.34 THEN total_CO2 < 2,260 t".

2.7.5 Random forest

Random forest or random decision forest is an ensemble learning method for classification and regression, which operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests overcome the decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

DRAFT VERSION

In the URBANITE project, random forests are used among other classification methods in the URBANITE DSS.

DRAFT VERSION

3 Analysis components

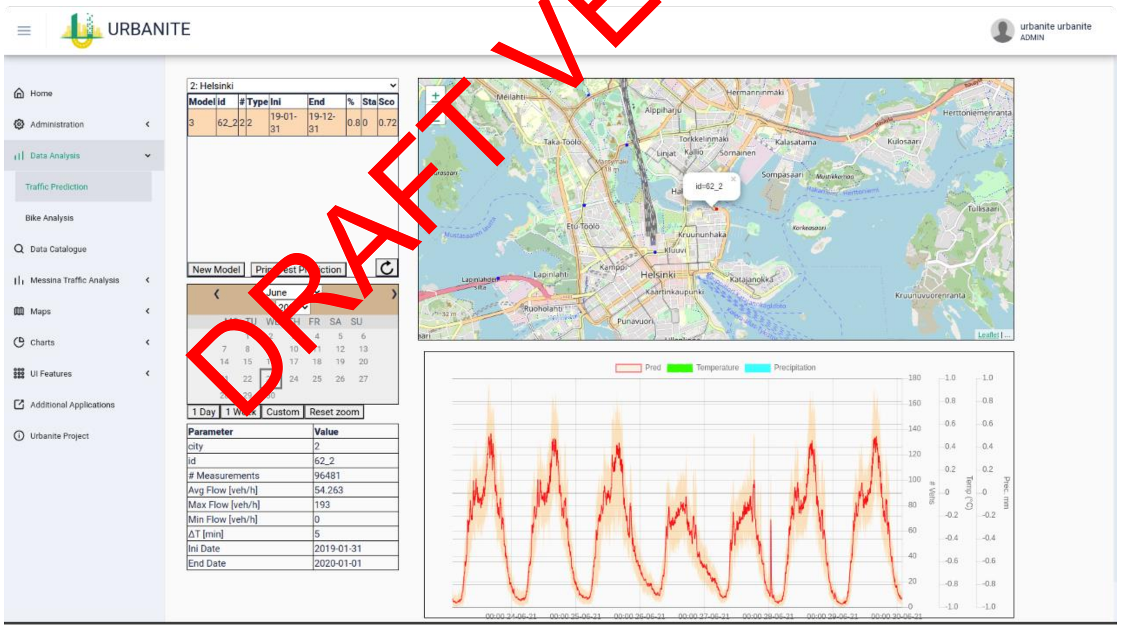
In this section we present the Analysis components that have been developed within the URBANITE project to address specific problems of the URBANITE use cases. Thus, the components described in the following sections implement specific analyses developed in cooperation with the use cases.

3.1 Traffic flow prediction

Road traffic forecasting has been a topic of study since the sixties [4] when time series analysis methods were mainly used [5] [6] [7]. In the last two decades, heuristic machine learning methods [8] started being used to find more complex relations within the traffic data. Nowadays the traffic prediction component has become one key tool for any ITS system. The component developed within the URBANITE project can forecast what is the traffic flow that a sensor within the city would measure for a given set of features.

The set of features at the time of this deliverable include the day of the week and the time of the day, but other ones are in the process of being incorporated. Some of these features include if the day of the forecast is a bank or school holiday, weather features (precipitation and temperature), the arrival of ferries to Helsinki port or sport events (football games) in Bilbao (using the method developed in [9]). Note that the approach within URBANITE is not to consider previous measurements as features since the data is not available in real time. Therefore, this approach can be considered as long-term prediction because the predicting horizon is only limited by the accessibility of external features (i.e., access to a weather prediction for example).

DRAFT VERSION



Model	id	#	Type	Ini	End	%	Sta	Sco
3	62_2	2	2	19-01-31	19-12-31	0.8	0	0.72

Parameter	Value
city	2
id	62_2
# Measurements	96481
Avg Flow [veh/h]	54.263
Max Flow [veh/h]	193
Min Flow [veh/h]	0
ΔT [min]	5
Ini Date	2019-01-31
End Date	2020-01-01

Figure 14 Integrated tool to perform traffic prediction showing the Helsinki use case.

The web portal to the integrated tool can be seen in Figure 17, the tool allows to train a new model, to perform a prediction, and to visualise the results.
The process of performing the training implies that the user needs to choose the following:

- The regression model type, two options are available: random forest [10] and distribution inference [11] (only for features with discrete values).

- The number of features to consider: 1. considers only the day of the week, 2. also considers the time and so on.
- The time resolution, typically either 5 or 15 minutes. This is the aggregation period on which the individual counts of vehicles moving over the sensor are combined to produce a time series.
- The traffic sensor, this is chosen by selecting the available sensors within a map.
- The period of the training data, the period can be chosen from the available data within the Data Management Platform, being able to change this period allows for instance to avoid choosing the anomalous period due to the restrictions due to COVID-19. In addition, a percentage of the training data can be reserved to test the goodness of the model, this percentage can also be specified.
-

Once a model is trained, this can be used to perform a prediction. There are different ways to perform this, one way is to use the URBANITE web visualisation tool to choose a given date and perform the prediction for either the following 24 hours or the following 7 days. Alternatively, a specific set of features can be fed to the model using the REST Web service in JSON format to obtain a result at a given instant of time.

An example of the result can be seen in Figure 18 where in addition to the prediction (red line) the confidence interval is shown (orange band). The details of how to compute the confidence interval are explained in [12]. In this figure the result of the prediction for a week is shown, where the peaks for the different days are clearly visible, including the difference in the pattern due to the weekend (fourth and fifth peaks in the series).

DRAFT VERSION

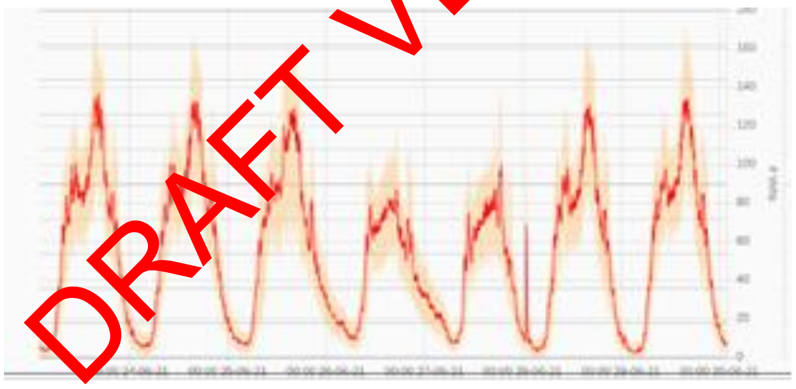
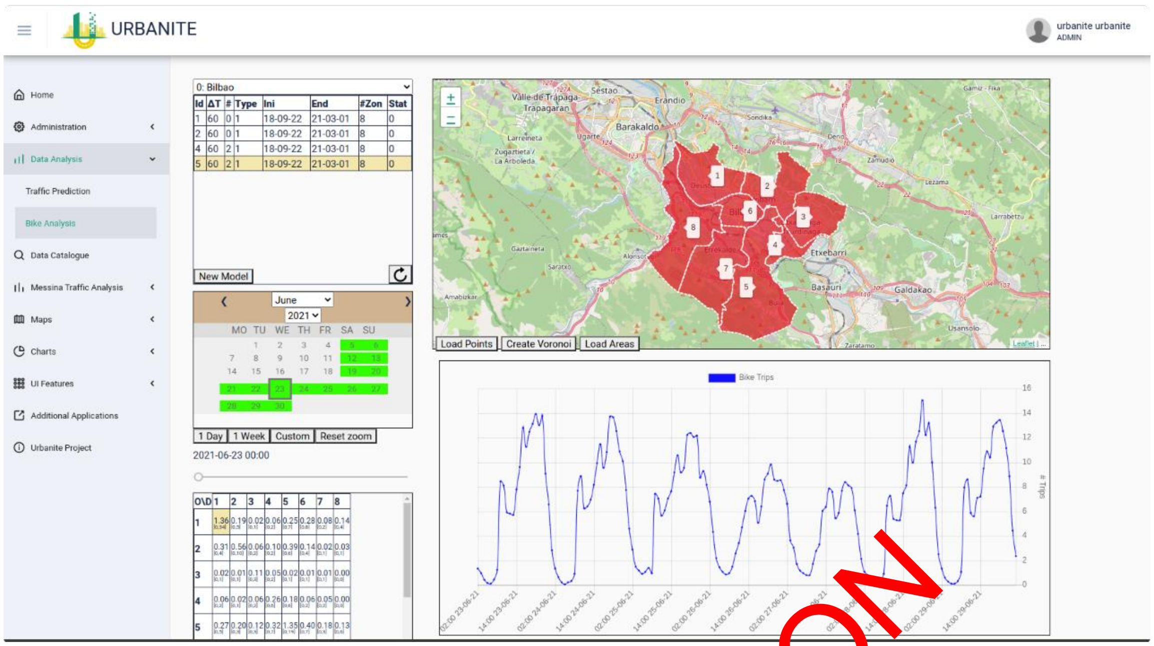


Figure 15 Detail of the visualisation for the result of the traffic flow prediction for 7 days including the confidence interval.

3.2 Bike analysis

The OD Matrix estimation works in a similar way than the prediction module, described in Section 3.1. In this case we use data from bike rental city service. Specifically, we consider the origins and the destinations of each one of the rentals. These are both temporally and spatially aggregated by providing the time resolution (the same way as for the traffic prediction) and by providing a set of geographic areas where to aggregate the origins and the ends of each rental. These areas can be specified either via a GEOJSON or by specifying a set of points; the URBANITE web tool can be used to obtain the Voronoi areas [13].



0: Bilbao

Id	ΔT	#	Type	Ini	End	#Zon	Stat
1	60	0	1	18-09-22	21-03-01	8	0
2	60	0	1	18-09-22	21-03-01	8	0
4	60	2	1	18-09-22	21-03-01	8	0
5	60	2	1	18-09-22	21-03-01	8	0

2021-06-23 00:00

O\D	1	2	3	4	5	6	7	8
1	1.36	0.19	0.02	0.06	0.25	0.28	0.08	0.14
2	0.31	0.56	0.06	0.10	0.39	0.14	0.02	0.03
3	0.02	0.01	0.11	0.05	0.02	0.01	0.01	0.00
4	0.06	0.02	0.06	0.26	0.18	0.06	0.05	0.00
5	0.27	0.20	0.12	0.32	1.35	0.40	0.18	0.13

Figure 16 Integrated tool to perform OD matrix estimation for the Bilbao use case.

Training a model to perform OD matrix estimation implies choosing a regression model type, the number of features, the time resolution, and the period of training data. Among the features that can be chosen in order to compute the OD matrix are the following:

1. Hour of the Day (HoD)
2. HoD and Day of the Week (DoW)
3. HoD, DoW, and School and Holidays (SaH)
4. HoD, DoW, SaH and Meteo (MET)
5. HoD, DoW, SaH, MET and Futbol (FUT)

These variables have been chosen after a study of the different factors that may have an effect on bike usage. Figure 20 shows the results of such a study, specifically this study was performed for the Bilbao use case and in graphs of the figure the y-axis corresponds to the total number of trips per hour (we have summed up the contribution from all the zones) while in the many x-axis different variables are being shown. It is noted that for those variables which do not take continuous values, i.e. that are categorical, noise has been added to its value in order to expand the distribution and have a better visualisation of it. This effect can be observed in all the graphs with the exception of the first two corresponding to temperature and precipitation level.

DRAFT VERSION

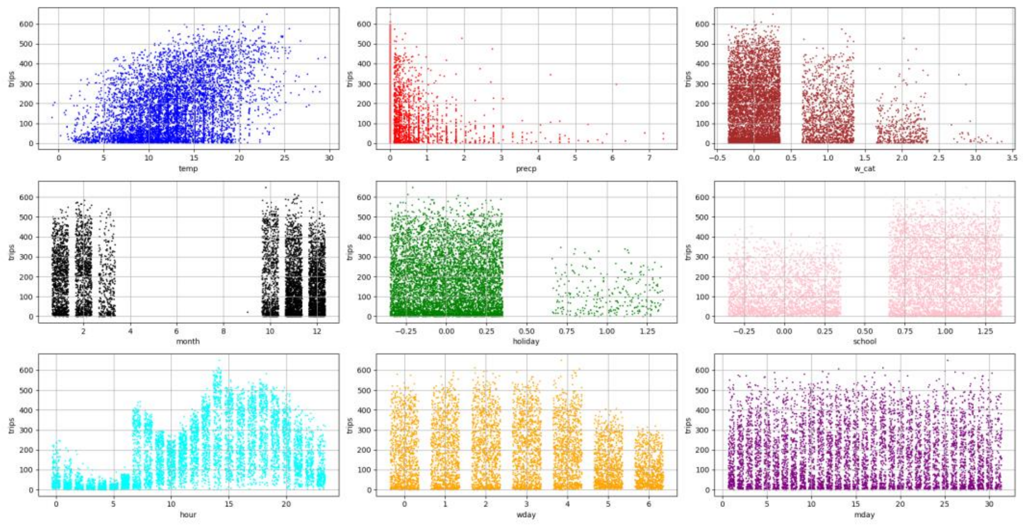


Figure 17 Dependence on the overall trips number per hour depending on the following variables (from left to right and top to bottom): temperature, precipitation, weather category, month, holiday index, school index, hour of the day, weekday and month day.

Special care has been taken to measure the effect of football on the usage of the service. It was chosen not to do the same analysis as it has been performed above for the rest of the variables due to the very small samples available in the dataset. One way to gauge the effect of football games is to train a model with data that contains no day with football games and apply it to exactly those days removed. Comparing the results of such a model can give a hint of if the football has an effect and how to incorporate it into the model. Figure 21 shows the results that have been obtained for the case of the Bilbao showing that there is no clear effect in the total number of trips per hour. It can be also observed that in the beginning of the dataset the predictions are higher that the real values and the second set of data is the other way around. More investigation needs to be done in order to measure the effect of football games, probably repeating this study to the trips between individual zones.

DRAFT VERSION

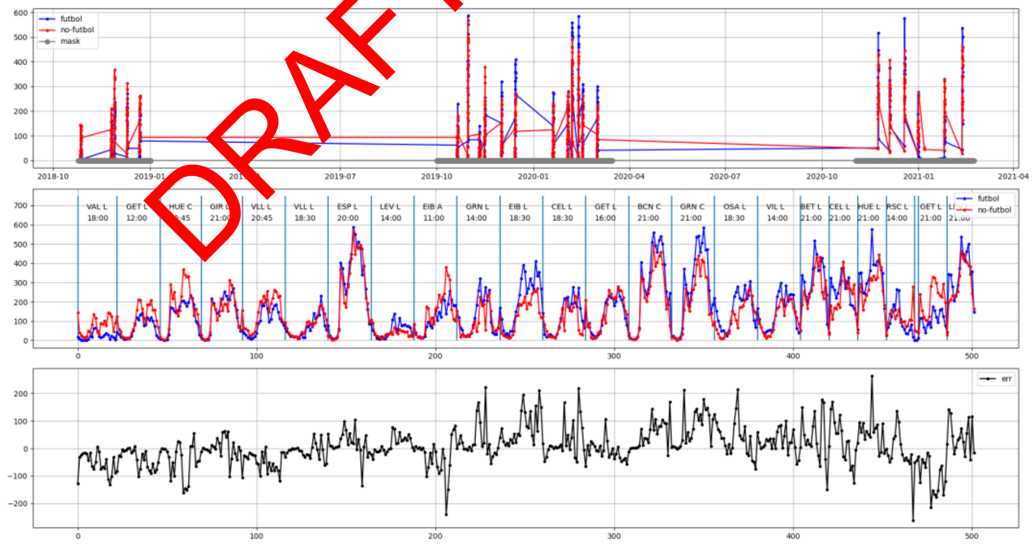
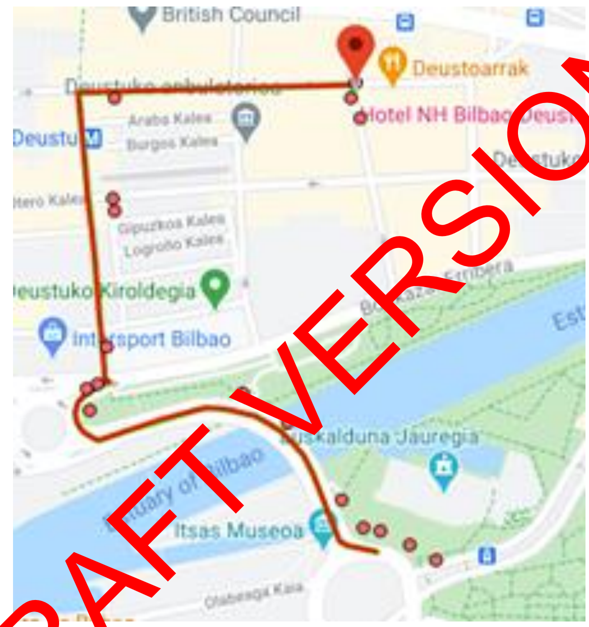


Figure 18 Dependence on the football games. Top graph shows the prediction without football (in red) and the real data with football plotted against the date. The middle figure shows the same data removing the days without data and the bottom one is the difference between the two-time series.

The result of the OD Matrix estimation, at a specific instant of time, consists of a square matrix of size *NxN* where *N* is the number of different areas considered for the spatial aggregation. The web tool within URBANITE allows to compute and visualise these estimations for all the instances within a period (typically a day or a full week). In Figure 19 a detail of the web tool is shown where it can be seen the result at a given instant in the form of a matrix (lower left hand side) and the time evolution of one of the matrix components for a whole week (lower right hand side). It is worth mentioning that this process to estimate OD matrices, by means of the use of regression algorithms, has the capability of generalisation obtaining results even in regions of the feature space where no values have been obtained yet.

3.3 Bike Trajectories

This component consists in a tool able to analyse not only the origin and the destination of trajectories but also what happens in between. More specifically, we can think of this tool's goal to be obtaining the points more popular to visit in a trajectory. The data processing is done in two different phases: the cleaning phase, and the aggregation phase.



DRAFT VERSION

Figure 19 Result of the cleaning phase for a set of GPS points obtained from a single bike city rental in Bilbao.

The cleaning phase is a crucial phase of processing GPS data obtained from affordable, not very accurate sensors or in areas with tall buildings (urban environment) where the multi-path of the satellite signal can increase the noise of the measurements. The purpose of this phase is to align the obtained measurements with the navigational road network, i.e., the possible allowed positions for the vehicles. In URBANITE, Hidden Markov models [16] are used in this phase. Moreover, this process provides an additional result, which consists in the most likely points between measurements.

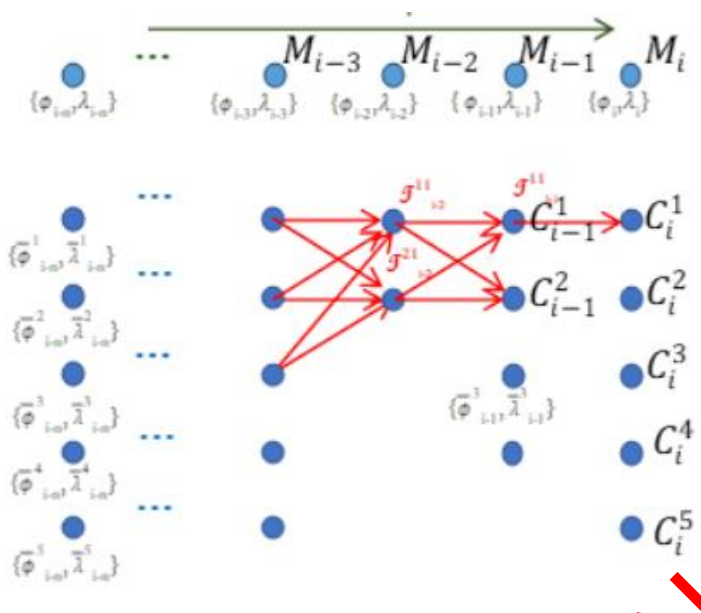


Figure 20 Trellis Diagram with the measurements on the top and the candidate points in the bottom.

One way to visualise the architecture of a Hidden Markov Model is by means of the Trellis diagram. In Figure 23 we show an example of such diagram in which for each GPS measurement M_i we assign a collection of K candidate points C_i^k. These candidate points are points on the road network and are the closer one to the measurement. A naive strategy could assign the measurement simply to the closest of the candidate points but in the HMM considered in URBANITE the distance to the measurement is considered as the emission probability. In addition to this probability, every pair of successive candidate points has a transmission probability given by the difference between the "on route" distance

$$D_R(C_{i-1}^k,C_i^l)$$

and the "straight line" distance

$$D_L(C_{i-1}^k,C_i^l).$$

$$\text{Emission Probability: } O^k_i \sim exp\left\{-\,{D_L(M_i,C_i^k)^2}\big/{2\,\sigma^2}\right\}$$

$$\text{Transmission Probability: } T^{k\,l}_{i\text{-}1\,i} \sim exp\left\{-\,\left[D_R(C_{i-1}^k,C_i^l)\;-D_L(C_{i-1}^k,C_i^l)\right]\big/{\beta\Delta T^2}\right\}$$

The model has a couple of free parameters (*β, σ*) that need to be loosely adjusted beforehand. Using this language, the problem of the HMM consists of finding the path in the Trellis diagram that maximises the overall probability, which locally does not need to correspond to the naive approach of maximising only the emission probability (i.e. closest point to measurement).

The second, i.e., the aggregation phase, compares the points obtained in the cleaning phase for all the trajectories. Probably the simplest of these aggregations is to compute the number of times a location is visited independently of the trajectory it belongs. The result of this process applied to the trajectories of the bike city service in Bilbao is shown in Figure 22.

DRAFT VERSION

Other types of aggregations can also be performed, as for example most likely points to be visited depending on the day of the week and the time of the day, the most popular chain of consecutive points visited, the longest route accomplished, etc.

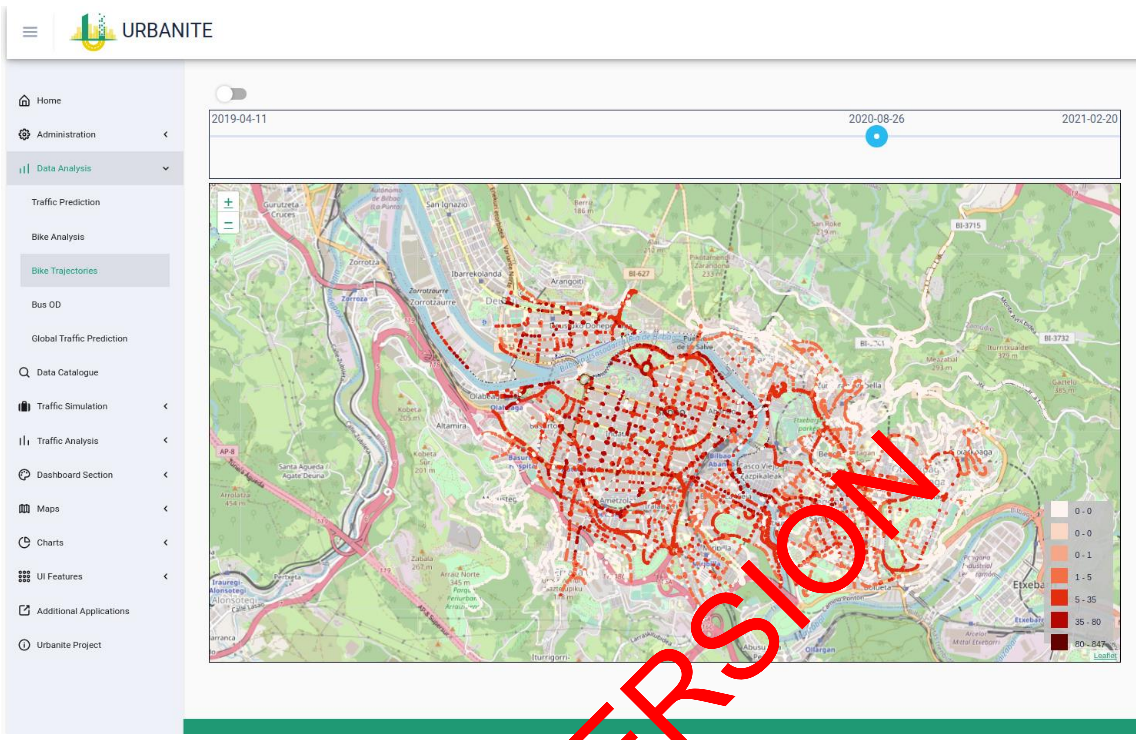


Figure 21 Visualisation of the most popular trajectory points for the bike city service.

3.4 Bus OD Matrix estimation

The purpose of the bus OD Matrix estimation component is to estimate the origin destination matrix for passengers using public transportation. This component is specifically designed for the Bilbao use case and the Bilbobus bus city service. The operation of this service is such that passengers only have to check when boarding the bus and not when leaving the vehicle. This implies that the service administrators do not have any knowledge of the location where the passengers get off.

This problem can be overcome by processing data obtained from the smart card system used across the city. This system has a very high penetration among the city users of the bus service and stores the information of when and where a given passenger checks when boarding a bus. There exist two types of cards: a personalised one that can only be used by a single person and a general one that can be shared and that can actually be used by several people at the same time. The user has to check the card every time it jumps into a bus, independently if it is a connection (a leg within a larger trip consisting of more than one bus trajectories and which may not produce an expense) or not. Although this card can be used on different public transportation systems, such as buses and railways within the region, we only have information about its usage in the Bilbobus service. Finally, another important aspect is that in some lines the passengers do not need to leave the bus at the end of the line and can continue their trip to some stop on the returning way.

DRAFT VERSION

The trip chaining algorithm consists of assuming continuation in the usage of the smart card by passengers. The basic assumption is that a smart card user uses the card at a location near to the location where he/she got off the bus the last time. Of course, there exists the possibility that this assumption is broken, and it basically amounts to the passenger, holding the card, has travelled without using the Bilbobus service. Displacements that can break this assumption can be done on foot, by car or by using a different public transportation and then use the smart card in a completely different place. Nevertheless, and before further examination against alternative measurements, the assumption seems to be reasonable and the best one that can be applied with the amount of information available.

This component has two different modes of usage: “per line” and “per trip” modes. In both modes of usage, the component computes an estimation for the location where every passenger has got off the bus at every leg in their trips. The data from the smart card system records a line every time a passenger checks at the entrance of a bus. Among other information, every line has the following data:

```
card_id, line_id, stop_id, time, trip_id, driver_id, direction
```

This last field is supposed to store the information about the direction of the bus (up vs. down the city) but unfortunately this field is not set in the data obtained and therefore needs to be estimated before continuing with the rest of the computation. The processing per line consist on the following:

- Compute the direction of the bus (up vs down): this step implies obtaining the stops from the lines with the same trip_id (this field denotes a specific trip for the bus) and comparing these stops with the stops in the bus schedule (i.e. the GTFS file). This process does not only assign the direction of this line but all the lines of the same trip_id.
- Compute the closest stop within the scheduled trip to the following bus entrance for the same card_id.

The difference between the modes of usage relies on how the information is aggregated. In the first mode, the aggregation of the information is performed on a per line basis. In this case the information the user of the component should choose is a specific line and a stop in that line. The component then shows the number of passengers that have got off the bus at the subsequent stops in the line.

DRAFT VERSION

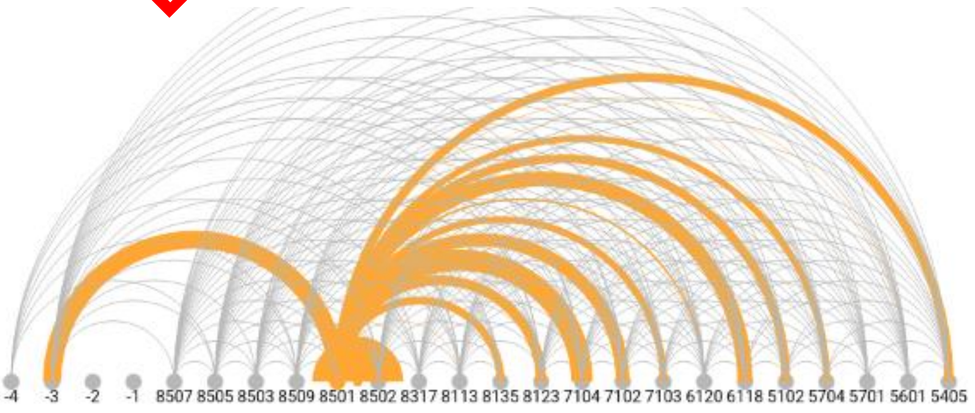
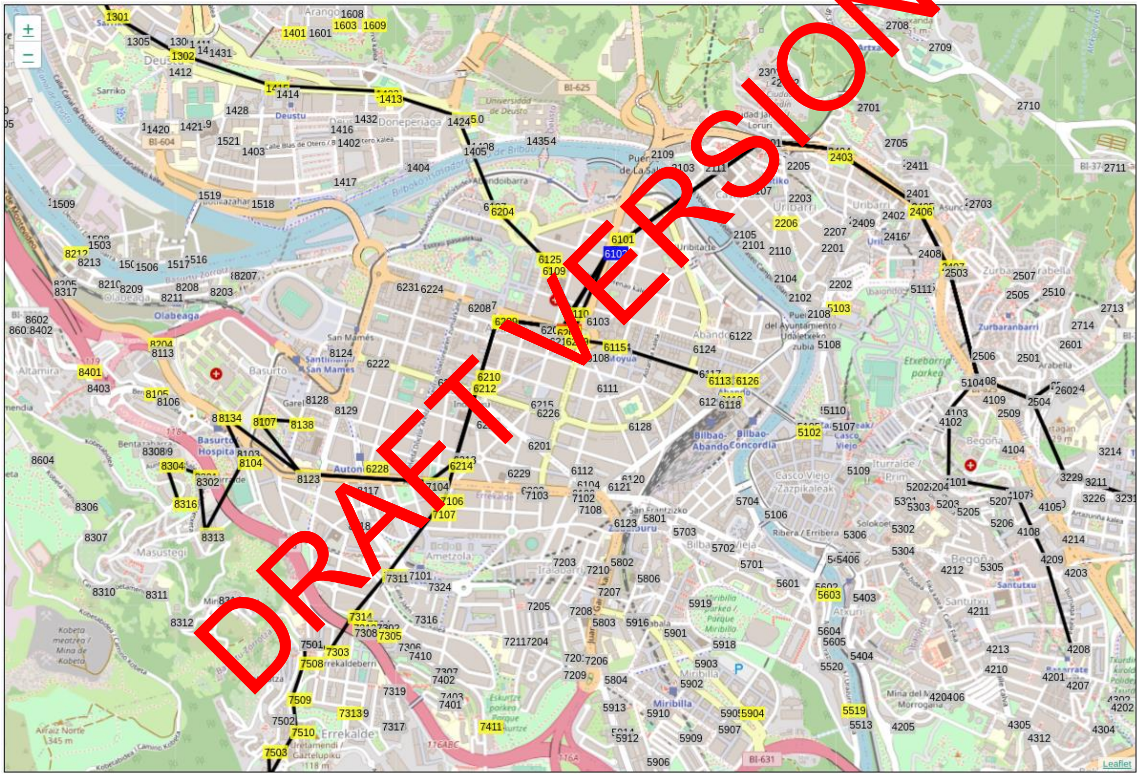


Figure 22 Visualisation of the distribution of the passengers according to the stop they get off. With the selected bus station numbered 8501, the orange lines' thickness visualises the number of people that entered at the bus station 8501 using the bus to travel to other bus stations.

In Figure 25 a visualisation of the results of the "per line" mode via a chord diagram shows the number of people that gets off on each stop by the width of the chord connecting the stops. One of the features that are not yet incorporated into the algorithm is the possibility of the passenger to remain in the bus after the final stop along one of the directions and continue to a stop in the reverse direction. This may happen due to passengers missing their stops or managing to catch the same bus in the other direction after doing some short activities.

Alternatively, the "per trip" mode joins legs in different buses by the same card_id and aggregates not the exit stop from each bus but the final destinations. This mode joins bus rides that are less than 45 minutes apart from each other. Potentially in this mode, every stop is connected with every other one and having a chord diagram to show the results does not result adequate. Instead a map is used showing the stops coloured with different tones (from yellow to red) depending on the number of people with that destination (see Figure 26).



DRAFT VERSION

Figure 23 Visualisation of the results for "per trip" mode. The origin stop is marked in blue and the final destination stops are denoted in tones from yellow to red. The visualisation also shows the bus lines that contain the origin.

In some cases, the algorithm fails to obtain a result for the estimation of the exit stop. There are several reasons why this can occur:

- The algorithm not being able to determine the direction of the line (due to a possible mismatch with the bus schedule file),
- Not finding a bus stop, along the bus line, after the entering stop that is close to the following checking for the same card.
- Not having any posterior checking for the same smart card.

4 Data visualisation methods

This section describes the visualisation methods developed to support the URBANITE exploratory data analysis module. The objectives of the visualisations are described with examples visualising some of the simulation results, such as simulated vehicle movements and traffic flows, and some of the geo-spatial data used for traffic simulation creation.

4.1 Objectives

The objectives of the data visualisation methods are to present the data in an understandable and clutter free way that does not cause cognitive overload. [14]

The visualisations in this section include simulation result visualisation, such as traffic flows, air pollutant emissions and vehicle movements. Other simulation-related data includes locations of points of interest and values of district attributes, such as demographics of the district residents.

The aspects we have focused on during the development of these visualisations are understandability, interactivity, and relevance. Only the visualisations deemed relevant are included. This section does not cover the data visualisation methods covered in Section 2 and its subsections.

4.2 Map-based visualisations

Multiple visualisations were developed to visualise certain data:

- Visualisation of the traffic flows is shown in Figure 28. Each street is overlaid with a line, coloured according to the traffic flow intensity. Less intensive flows are shown with blue hue and more intensive flows are shown in red. The colour scale consists of a five-colour ramp selected for best visibility on the base map.
- Visualisation of emissions of specific pollutants. Each street is overlaid with a line, coloured according to the amount of selected pollutant. Streets with less emitted pollutants are shown in blue and streets with more are shown in red.

These visualisations are implemented using JavaScript and based on maps provided by the library Leaflet.js.

The map layers are generated from the simulation results by aggregating road network links by street name and summing the selected attribute for the day or per hour, thus enabling time series visualisation of changes throughout the day or a static daily attribute visualisation.

A special case is the time series visualisation of simulated vehicle movements on the map, shown in Figure 27. Due to the large number of simulated vehicles, the visualisation is quite resource intensive and further optimizations are needed, as the current version needs about one second for updating the positions of the vehicles.

DRAFT VERSION

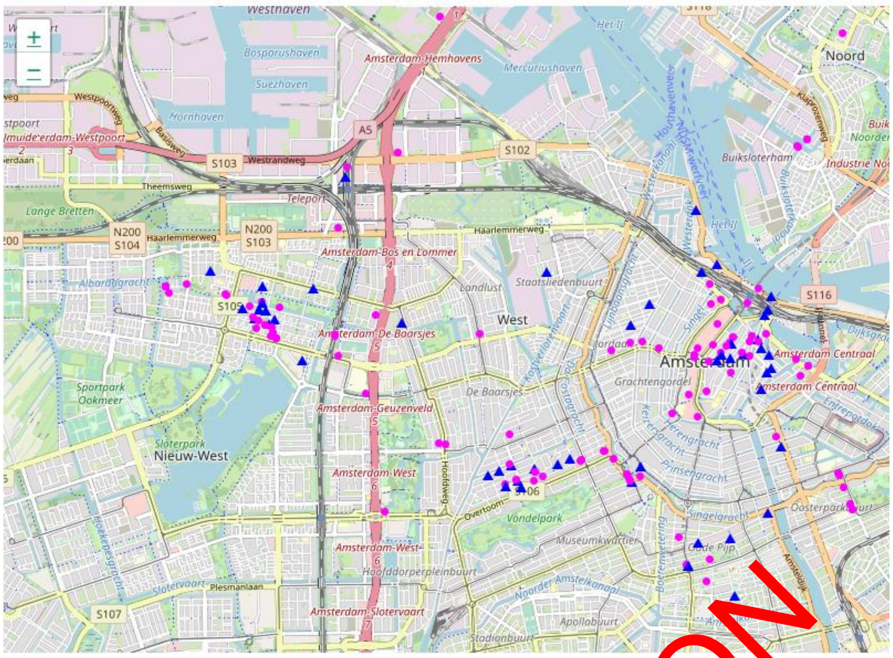


Figure 24 Locations of different simulated vehicles. This is a frame from an animated visualisation.

DRAFT VERSION

4.3 Traffic flow overlay on roads

Traffic flow is the amount of vehicles that pass a certain point on the road in a time slot. Traffic flows are commonly visualised using line-based map layers, where the traffic flow is represented either by line thickness or colour. To specifically show the modal split of the traffic flows we can show them separately or at the same time. In the latter case, it is best to use colour codes to represent different types of vehicles and line thickness to represent the traffic flow.

The category of traffic data contains multiple different data types that have to be visually represented using different methods.

Some of the data that is shown using the methods for geo-spatial data visualisation are:

- Traffic counts, shown either spatially by aggregating the counts per day or temporally by aggregation of the counts per specific time slot, commonly hours. Traffic counts at a specific location depend on time.
- Traffic flows measured in vehicles per hour passing a road. The traffic flow at a specific location depends on time. The specific flows for different modes, such as public transport, heavy duty vehicles, bicycles and pedestrians are visualised on separate map layers. An example from the simulation of Bilbao is shown in Figure 28.
- Congested roads. Simplest way of identifying problematic roads or junctions is to show the locations of congested traffic. We can detect congested traffic and traffic jams by searching for road segments with high traffic density and travel speed below the free-flow speed.

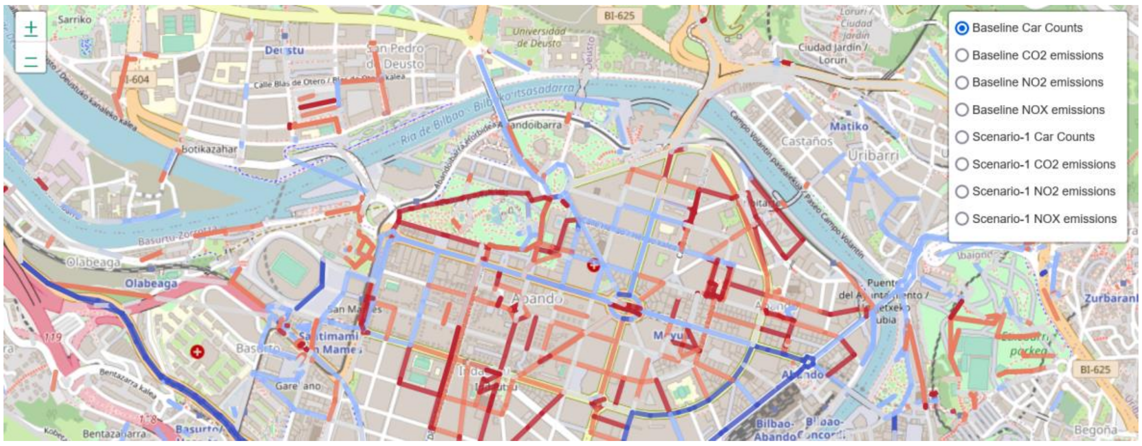


Figure 25 Coloured lines show the number of vehicles on specific roads, where red is higher, and blue is lower.

4.4 Air pollution, noise pollution

Generally, the most common method for visualising air pollution is an air quality index heat map. Some of the advantages of visualising air pollution as a heat map are high understandability and very low visual cluttering. A negative aspect of heat maps in this case is that air pollution often does not spread equally in all directions due to air movements and buildings blocking the pollutants' paths. Besides, heat maps may oversimplify the depicted phenomenon, especially when zoomed out. This is however not very important as the users are mostly interested in general pollution levels and in the case of the URBANITE project the levels of specific pollutant emissions. The visualisation is shown in Figure 29.

DRAFT VERSION

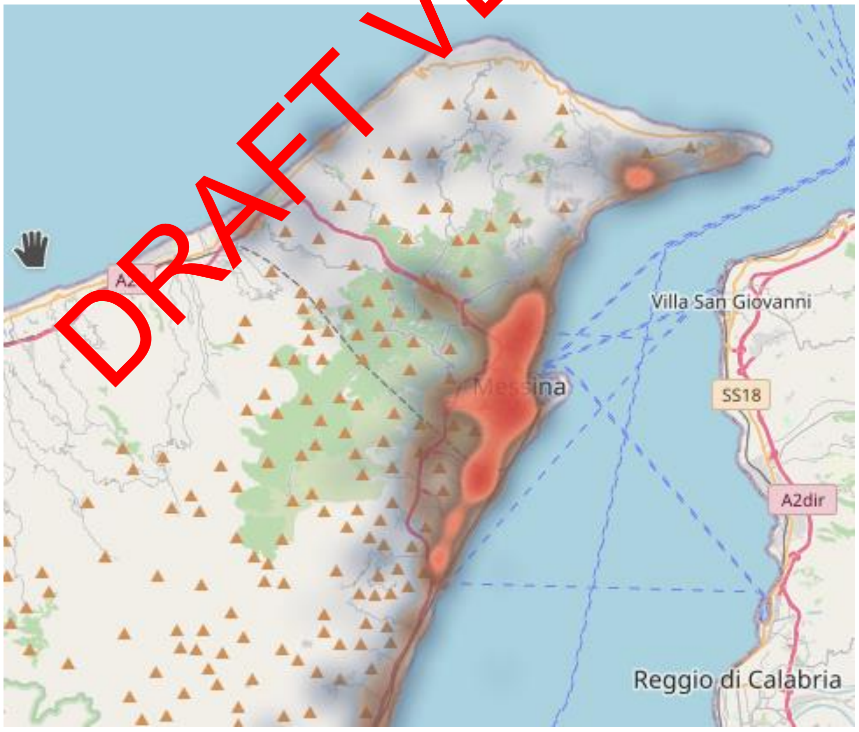


Figure 26 A heatmap showing air quality in Messina, based on sample data.

4.5 Points of interest

Points of interest include congestion hotspots, locations of public transport delay, accident prone junctions as well as some of the simulation input data, such as facilities.

The category of traffic data contains multiple different data types that have to be visually represented using different methods.

Some of the data that are shown using the GIS-methods are:

- Traffic counts, shown either by aggregating the counts per day or spatio-temporally by aggregation of the counts per specific time slot, commonly hours.
- Traffic flows, measures in vehicles per hour passing a road. The traffic flow at a specific location depends on time. The specific flows for different modes, such as public transport, heavy duty vehicles, bicycles and pedestrians are currently visualised as separated layers.
- Congested roads. Simplest way of identifying problematic roads or junctions is to locate them on a map. We can detect congested traffic and traffic jams by searching for road segments with high traffic density and travel speed below the free-flow speed.
- Locations of facilities, used in preprocessing of the data to create simulations, shown on Figure 30.

Some of the more general traffic simulation data are better visualised using simpler line and bar charts, detailing the traffic volumes and modal splits. Traffic flows at specific locations over time are shown using a line chart. Traffic flow predictions at specific locations are shown using a line chart with the confidence interval included to inform the user that these are not exact. Modal splits of traffic at specific locations as well as city-wide aggregations of modal splits are visualised using area charts or stacked area charts.

DRAFT VERSION

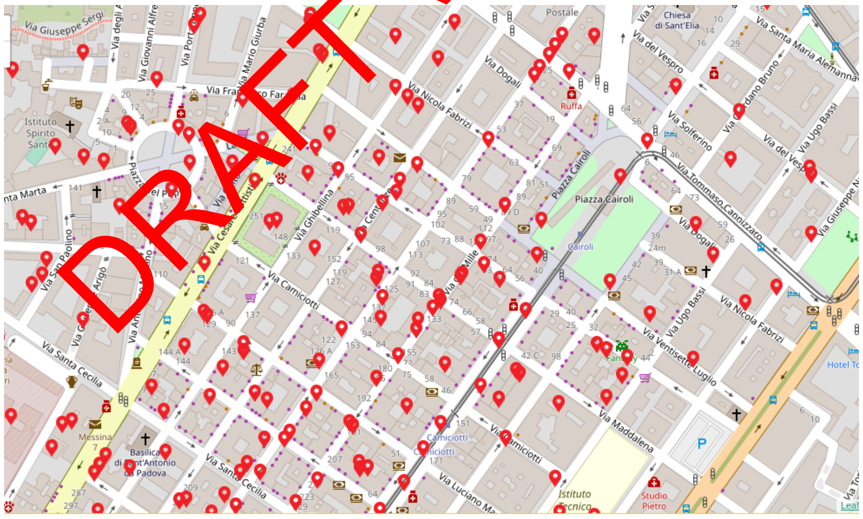


Figure 27 POIs in Messina marked with an icon, showing various facilities in the city centre.

4.6 Other traffic-related visualisation methods

We use one colour map for all the visualisations that are layered over the city map. The colour map selected must be diverging in order to highlight best and worst values according to their desirability. The chosen map is a diverging colour map using red colours for undesired values and blue for desired values. The colour map should also be appropriate for colour blind users to avoid potential misunderstandings. With the requirements of the colour map defined, we selected a colour map named cold-warm that fits our needs. The colour map is shown in Figure 31. We have opted to use a five-step colour map instead of the full gradient to make the extreme values stand out more.

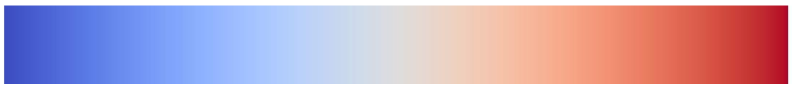
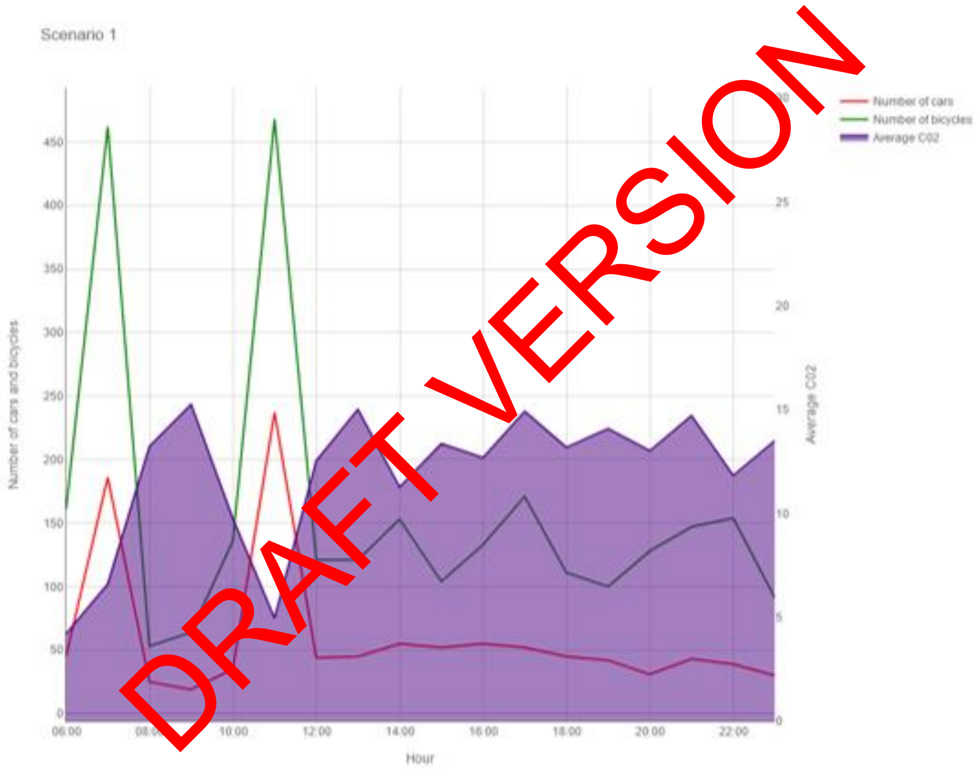


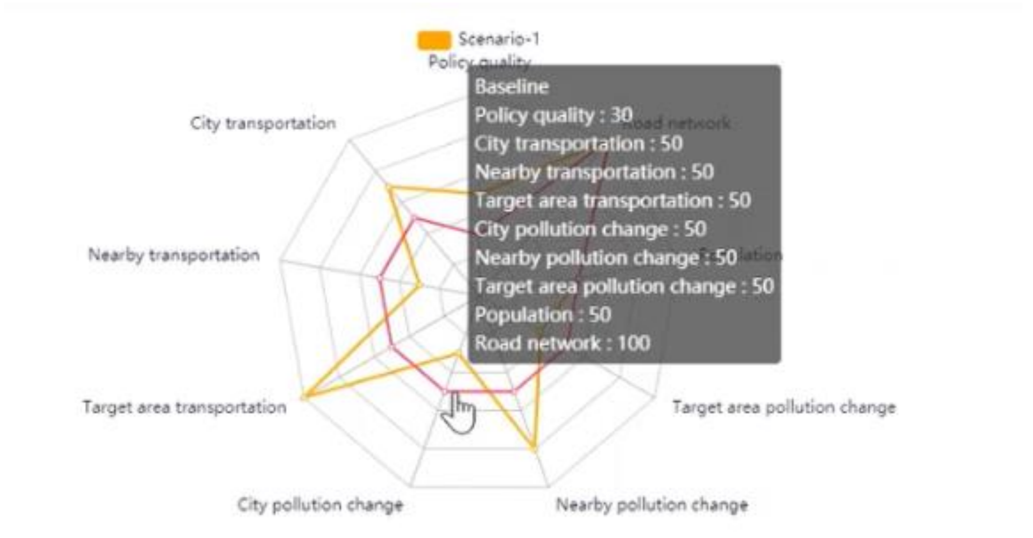
Figure 28 The colour map used to represent different values.

We use interactive charts implemented using the echarts library to create line charts, histograms, and spider charts. Line charts are used to analyse the modal splits on streets and the level of selected emitted pollutant CO2 using an overlay area chart as shown in Figure 32. Figure 33 shows a spider chart comparing two different policies by visualising the values of KPIs calculated from the simulations of the policies. These were made interactive to allow the user to zoom in and move the viewport around. Hovering over any of the lines shows the number of trips of the appropriate modes as well as the mode the line represents.



DRAFT VERSION

Figure 29 Line charts show the number of cars and bicycles, and the area chart shows the amount of CO2.



Baseline
Policy quality : 30
City transportation : 50
Nearby transportation : 50
Target area transportation : 50
City pollution change : 50
Nearby pollution change : 50
Target area pollution change : 50
Population : 50
Road network : 100

Figure 30 A spider chart shows the values of different KPIs for two different simulations.

DRAFT VERSION

5 Delivery and usage

5.1 Data modelling methods

5.1.1 Installation instructions

As previously mentioned, Orange2 is a free, open-source software for data visualisation, machine learning, data mining, and data analysis. It has been developed by the University of Ljubljana from Slovenia [2]. You can download the program from the web page: https://orangedatamining.com/download/#windows as standalone or portable solution.

If you are using python provided by Anaconda distribution, you are almost ready to go. Add conda-forge to the list of channels you can install packages from: conda config --add channels conda-forge and run conda install orange3. Orange can also be installed from the Python Package Index: pip install orange3.

5.1.2 User Manual

You can find documentation around the visual programming, development and the python library, and different tutorials along the lifecycle of data processing (https://orangedatamining.com/docs/).

5.1.3 Licensing information

The license terms for the software are under discussion among the consortium. AGPLv3[3] are being considered.

[2] More information about Orange is available on the official website: https://orangedatamining.com.
[3] https://www.gnu.org/licenses/agpl-3.0.en.html

5.1.4 Download

You can download the program from the web page: https://orangedatamining.com/download/#windows as standalone or portable solution.

5.1.5 Application in the context if URBANITE use cases

Data and configuration files for the exercises and experiments explained in D4.2, can be found on https://git.code.tecnalia.com/urbanite/private/wp4-algorithms-and-simulation/models_and_working_files/-/blob/main/Strategies%20and%20algorithms%20for%20data%20modelling%20and%20visualisations/Strategies_and_algorithms_for_data_modelling_and_visualisations.7z, on the GitLab maintained by Tecnalia[4].

This directory includes information includes: icons, images, and a set of supporting python scripts.

5.2 Map-based visualizations

This project aims to implement prototype of a map visualization component and traffic flow visualization component. This repository contains Urbanite UI Template. This template is built starting from NGX-Admin[5], an open source dashboard based on Angular[6], Nebular[7] with Eva Design System[8].

5.2.1 Installation instructions

To properly install the template, follow these steps:

Pre-installation: Git, Npm, Angular CLI
git clone https://git.code.tecnalia.com/urbanite/wp5-integration-and-devops/urbanite-ui-template.git
cd urbanite-ui-template
npm install

5.2.2 User Manual

Run. To start the application, follow these steps:

- cd urbanite-ui-template
- ng serve

The application will be available at http://localhost:4200

DRAFT VERSION

[4] https://git.code.tecnalia.comNgx-admin - most popular admin dashboard on Angular 9+ and Nebular. (akveo.github.io)
[4] Angular
/urbanite/private/wp3-data-management/storage/dataStorage
[5] Ngx-admin - most popular admin dashboard on Angular 9+ and Nebular. (akveo.github.io)
[6] Angular
[7] Nebular - Customizable Angular UI Library, Auth and Security (akveo.github.io)
[8] Eva Design System

5.2.3 Licensing information

The license terms for the software are under discussion among the consortium. AGPLv3[9] are being considered.

5.2.4 Download

Data and configuration files for the exercises and experiments explained in D4.2, can be found on the Data Modelling with Orange Widgets the GitLab maintained by Tecnalia[10].

5.3 Bike OD-Matrix Module

The aim of this module is to obtain OD-Matrixes for the bike city service. This component processes and analyses the historical data obtained from the bike city service. Specifically, it processes the origin and destination for every trip.

5.3.1 Installation instructions

To install this software as stand-alone (independently of the Urbanite UI):
git clone https://git.code.tecnalia.com/urbanite/private/wp4-algorithms-and-simulation/bikeanalysis.git

The last version of the component is located in the v2.0 directory. To run it: bikeanalysis$ cd v2.0 bikeanalysis/v2.0$ docker-compose up -d

5.3.2 User Manual

The user manual is available on the webpage: https://git.code.tecnalia.com/urbanite/private/wp4-algorithms-and-simulation/bikeanalysis.git

5.3.3 Licensing information

The license terms for the software are under discussion among the consortium. AGPLv3[11] are being considered.

5.3.4 Download

All the source code is available the GitLab maintained by Tecnalia[12]: https://git.code.tecnalia.com/urbanite/private/wp4-algorithms-ad-simulation/bikeanalysis

DRAFT VERSION

5.4 Traffic Prediction Module

The purpose of this module is to perform prediction of the traffic flow at the locations of the city sensors. This component processes and analyses the historical data measured by the loop sensors. The loop sensors are sensors that measure the number of vehicles that go through a specific point in a given road. Typically, they are recalled as "loops" because the most common type of these sensors are made of magnetic loops that measure the inducted current produce by the metal of the moving vehicles. These sensors can be of different types able to measure speed, individual vehicle passing and other characteristics at the position of the road.

[9] https://www.gnu.org/licenses/agpl-3.0.en.html
[10] https://git.code.tecnalia.com/urbanite/private/wp3-data-management/storage/dataStorage
[11] https://www.gnu.org/licenses/agpl-3.0.en.html
[12] https://git.code.tecnalia.com/urbanite/private/wp3-data-management/storage/dataStorage

5.4.1 Installation instructions

To install this software as stand-alone (independently of the Urbanite UI):

```
$ git clone https://git.code.tecnalia.com/urbanite/private/wp4-algorithms-and-simulation/trafficprediction.git
```

The last version of the component is located in the v2.0 directory. To run it:

```
trafficprediction$ docker-compose up -d
```

5.4.2 User Manual

The user manual is available on the webpage: https://git.code.tecnalia.com/urbanite/private/wp4-algorithms-and-simulation/trafficprediction

5.4.3 Licensing information

The license terms for the software are under discussion among the consortium. AGPLv3[13] are being considered.

5.4.4 Download

All the source code is available the GitLab maintained by Tecnalia[14]: https://git.code.tecnalia.com/urbanite/private/wp4-algorithms-and-simulation/trafficprediction

5.5 O/D from Trip Chaining Module

The purpose of this the calculation of the O/D matrices for buses, with information exclusively on climbs, which prevents an exact knowledge of coupcaicon, most requested stops. It is based on the transactional data of uploads, and we assume that the download stops coincide with those where the return trip is later resumed (trip-chain-based estimation). An analysis can be done at the line level (assuming it turns on the same line) or by thrips, including combinations of different lines.

5.5.1 Installation instructions

Directory Structure:

- data:
  - GTFS: gtfs_sept17/gtfs_sept17.zip
  - Barik Data: Bilbobus_09_21.csv
- python:
  - How to run scripts

```
$ ./run_all.sh
$ python fill_table.py
```

[13] https://www.gnu.org/licenses/agpl-3.0.en.html
[14] https://git.code.tecnalia.com/urbanite/private/wp3-data-management/storage/dataStorage

DRAFT VERSION

Content:
escription: It contains the following scripts:

- run_all.sh:
    - get_stops_by_linea_ordered.py
        - consumes: GTFS_FILE
        - produces: ordered_stops_by_line_dict.pkl
    - add_route_id_to_line.py
        - consumes: GTFS_FILE, Bilbobus_09_21.csv
        - produces:
            - odf_with_route_id.pkl: ordered dataframe with route id (intermediate file)
    - compute_stop_OD.py
        - consumes: odf_with_route_id.pkl: ordered dataframe with route id (intermediate file)
        - produces: stop_od_by_line_dict.pkl: OD Matrix stop to stop by bus line.
- fill_table.py:
    - consumes: stop_od_by_line_dict.pkl, ordered_stops_by_line_dict.pkl
    - produces: <>
- codigos de error:
    - -4: No next departure stop found in the file.
    - -3: Distance from End to Ini larger that 500 m.
    - -2: There are no stops in the line after the departure stop.
    - -1: Number of lines for user more than a 1000, route_id = 'NN'

5.5.2 User Manual

In progress.

5.5.3 Licensing information

The license terms for the software are under discussion among the consortium. AGPLv3[15] are being considered.

5.5.4 Download

All the source code is available the GitLab maintained by Tecnalia[16]: https://git.code.tecnalia.com/urbanite/private/wp4-algorithms-and-simulation/odfromtripchainning

5.6 Data Visualization Module

The purpose of this module is to provide different map-based visualization methods by using different diagrams (heatmap, matrix, traffic visualization, etc).

5.6.1 Installation instructions

Deployed with the general URBANITE UI.

DRAFT VERSION

[15] https://www.gnu.org/licenses/agpl-3.0.en.html
[16] https://git.code.tecnalia.com/urbanite/private/wp3-data-management/storage/dataStorage

5.6.2 User Manual

In progress.

5.6.3 Licensing information

The license terms for the software are under discussion among the consortium. AGPLv3[17] are being considered.

5.6.4 Download

All the source code is available the GitLab maintained by Tecnalia[18]: https://git.code.tecnalia.com/urbanite/private/wp5-integration-and-devops/urbanite-ui-template/-/tree/master/src/app/pages/messina.

6 Conclusions

This deliverable presents the developed data modelling and visualisation methods and their use within the URBANITE solution. Methods developed allow the users to view, analyse, model, and interact with the data. Since the data is heterogenous and covers a wide array of different types and formats, the methods to handle the data need to be different and cover many possible cases. The URBANITE project has taken this as a challenge and provides several solutions that manage different types of data.

The exploratory data analysis is facilitated using Orange, a data analysis, visualisation, and machine learning platform enabling simple visual programming of data analysis pipelines. We have developed several widgets for Orange to allow the users easy access to the harvested data. This data can be analysed using presented methods and the results presented using a variety of different visualisations. These methods have been used to develop some of the prebuilt models and allow users to use the same processes to build similar models.

Several components, listed in Section 3, have been developed providing specific analysis, identified during the project as important to the use cases. These are included in the web application and allow the users to analyse and visualise expected future traffic flow, cycling traffic patterns and common cyclist paths, and estimation of origin-destination matrices for public transport. These components are the traffic flow prediction component, bike analysis component, bike trajectories component, and the bus OD matrix estimation component.

Supporting the traffic simulation module [15], described in deliverable D4.4, are different visualisations of the simulation results, including air pollutants (CO_2, PMx, NOx), animated vehicle movements, values of KPIs on specific streets (bikeability, bike safety), and visualisations of map-based data, such as points of interest.

The work presented is ongoing and some of the methods described are still in progress. The next steps include finalisation of the work and integration, as well as optimizations to some of the methods, and will be finished by M30 when the tasks T4.1 and T4.4 conclude.

DRAFT VERSION

[17] https://www.gnu.org/licenses/agpl-3.0.en.html
[18] https://git.code.tecnalia.com/urbanite/private/wp3-data-management/storage/dataStorage

DRAFT VERSION

7 References

[1] B. G. Peters, «What is so wicked about wicked problems? A conceptual analysis and a research program. Policy and Society, 36(3), 385-396.,» 2017.

[2] Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., et al., «Orange: data mining toolbox in Python. the Journal of machine Learning research, 14(1), 2349-2353.,» 2013.

[3] Shulajkovska M., Smerkol M., Dovgan E., and Gams M., «Machine Learning-Based Approach for Estimating the Quality of Mobility Policies.,» 2021.

[4] I. Laña, J. Del Ser, M. Velez, and E. I. Vlahogianni, «Road traffic fore-casting: Recent advances and new challenges, IEEE Intelligent Transportation Systems Magazine, vol. 10, no. 2, pp. 93–109.,» 2018.

[5] Cook, M. S. Ahmed and A. R., «Analysis of freeway traffic time-series data by using Box-Jenkins techniques, Transportation Research Record, no. 722, pp. 1-9.,» 1979.

[6] Tsao., M. Levin and Y.-D., «On forecasting freeway occupancies and volumes (abridgment), Transportation Research Record, no. 773.,» 1980.

[7] Ratcliffe, C. Moorthy and B., «Short term traffic forecasting using timeseries methods, Transportation planning and technology, vol. 12, no.1, pp. 45–56.,» 1988.

[8] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, «Spatio-temporal short-term urban traffic volume forecasting using genetically optimized modular networks, Computer-Aided Civil and Infrastructure Engineering, vol. 22, no. 5, pp. 317–325.,» 2007.

[9] I. Olabarrieta, I. Laña, «Effect of Soccer Games on Traffic, Study Case: Madrid. 1-5. 10.1109/ITSC45102.2020.9294749.,» 2020.

[10] L. Breiman, «Random Forests. Machine Learning. 45 (1): 5–32. doi:10.1023/A:1010933404324.,» 2001.

[11] I. Laña, I. Olabarrieta, J. Del Ser., «Output Actionability of Traffic Forecasting Models: Measuring and Understanding Uncertainty of Forecasts to provide Confidence Levels (in preparation).,» 2021.

[12] G. Voronoi, «Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites" (PDF). Journal für die Reine und Angewandte Mathematik. 1908 (133): 97–178. doi:10.1,» 1908.

[13] P. Newson, J. Krumm, «Hidden Markov Map Matching Through Noise and Sparseness. 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2009), November 4-6, Seattle, WA,» 2009.

DRAFT VERSION

[14] Smerkol, M., Shulajkovska, M., Dovgan, E., and Gams, M., «Visualizations for Mobility Policy Design.,» 2021.

[15] Smerkol, M., Shulajkovska, M., Dovgan, E., and Gams, M., «Traffic Simulation for Mobility Policy Analysis.,» 2021.

DRAFT VERSION