Green mobility data models and services for smart ecosystems

D3.2 Green Mobility Services Development

Document Identification	
Contractual Delivery Date	28/02/2023
Actual Delivery Date	28/02/2023
Responsible Beneficiary	ATOS
Contributing Beneficiaries	IMEC, HOPU, IMREDD
Dissemination Level	PU
Version	1.0
Total Number of Pages	62

Keywords
Machine Learning, Green Mobility Services, Air Quality, Noise Annoyance, Bikes Availability, Traffic.

This document is issued within the frame and for the purpose of the GreenMov project. This project has received funding from the European Union's Innovation and Networks Executive Agency – Connecting Europe Facility (CEF) under Grant AGREEMENT No INEA/CEF/ICT/A2020/2373380 Action No: 2020-EU-IA-0281. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Document Information

Related Activity	Activity 3	Document Reference	D3.2
Related Deliverable(s)	D3.1	Dissemination Level (*)	PU

List of Contributors	
Name	**Partner**
Miguel Aguilar	ATOS
Ignacio Sevillano	ATOS
Carmen Perea	ATOS
Mehdi Nafkha	IMREDD
Benoit Couraud	IMREDD
Nuria Bernabé	HOPU

Document History			
Version	**Date**	**Change editors**	**Changes**
v0.0	15/01/2023	Miguel Aguilar	First draft and ToC
v0.1	28/01/2023	Ignacio Sevillano	Inputs from Atos
v0.2	01/02/2023	Mehdi Nafkha	Inputs from Nice
v0.3	06/02/2023	Miguel Aguilar	Inputs from ATOS
v0.4	10/02/2023	Nuria Bernabé	Inputs from Libelium (HOPU)
v0.5	11/02/2023	Benoit Couraud	Inputs from Nice
v0.7	13/02/2023	Miguel Aguilar	Formatting
v0.9	27/02/2023	María Guadalupe Rodriguez (ATOS)	Quality Review Form

Document History			
Version	**Date**	**Change editors**	**Changes**
v1.0	28/02/2023	Carmen Perea (ATOS)	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Reviewer	Mehdi Nafkha & Benoit Couraud (IMRED)	24/02/2023
Quality manager	Maria Guadalupe Rodríguez (ATOS)	27/02/2023
Project Coordinator	Carmen Perea (ATOS)	28/02/2023

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation/ acronym	Description
ACF	Autocorrelation Function
AEMT	Agencia Estatal de Meteorología
AI	Artificial Intelligence
AQC	Air Quality Calculation
AQF	Air Quality Forecasting Service
AQIC	Air Quality Index Calculation Service
ARIMA	Autoregressive Integrated Moving Average
BAF	Bikes Availability Forecasting Service
BSS	Bicycle Sharing System
CO	Carbon Monoxide
CSV	Comma Separated Values
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
EEA	European Environment Agency
kNN	K-Nearest-Neighbour
LAeq	average sound level measured
LAeq2	average sound level measured during 2h
LAmax	maximum sound level measured
LAmax2	maximum sound level measured in 2h
miMASK	percentage of time that noise exceeds a given threshold 65dB
MSRE	Mean Square Relative Error
NAC	Noise Annoyance Calculation Service

Abbreviation/ acronym	Description
NAF	Noise Annoyance Forecasting Service
NaN	Not a Number
NGSI-LD	Next Generation Service Interfaces- Linked Data
NO	Nitric Oxide
NO_2	Nitrogen Dioxide
NOX	Nitrogen Oxides
O3	Ozone
PACF	Partial Autocorrelation Function
PM	Particulate Matter
SO_2	Sulfur Dioxide
SARIMA	Seasonal autoregressive integrated moving average
SARIMAX	SARIMA modelling with exogenous factor
SDM	Smart Data Model
R2	Metric Coefficient of determination
TEIC	Traffic Forecasting Impact Calculation Service
TF	Traffic Forecasting Service
TR	Traffic Recommendation Service
VARMA	Vector autoregressive moving-average

Executive Summary

Deliverable “D3.2 Green Mobility services” collects the Activity 3’ “Green Mobility Services” main results.

The main objectives of the task are:

Define and implement the cross smart services for green mobility proposed by GreenMov.

This deliverable detailed the results of task “T3.2 Green mobility services development” and it is based on the results provided by the previous task “T3.1 Green mobility services definition”.

Two types of services are discussing in the document: Calculation and forecasting services.

Furthermore, the services are detailed and grouped into the following categories:

- Air Quality: Air Quality Forecasting, Air Quality Index Calculation.
- Traffic Environmental Impact: Traffic Forecasting, Traffic Environmental Impact Calculation, Traffic Recommendations.
- Bikes Availability: Bike’s availability forecasting.
- Noise Annoyance: Noise forecasting, Noise Annoyance calculation.

For each service the following information has been provided:

- Description of the service: An overview of each service is provided as well as an analysis of the data involved in the service (Historical data, pre-processing) used by the service (only for forecasting services)
- Model Development: This subsection explains the algorithms selected for each service, the training carried out and the results obtained.
- Model Deployment: this section provides the information about the service deployment provided.
- Conclusions where the main conclusion reached are explained.

On one hand, Activity 3 received inputs from Activity 2 “Smart Data Models for green mobility” and Activity 4 “Architecture for Context Broker enhancement in concurrent data intensive scenarios as mobility”. On the other hand, Activity 3 provided outputs to Activity 5 “Pilots deployment”. Therefore, the results achieved in this point are crucial for the successful conclusion of the project GreenMov.

1 Introduction

This deliverable represents the second report regarding Activity 3 “Smart services for green mobility” and it is the output of the task 3.2 “Green mobility services development”. This document thoroughly outlines both the development of the models for each service and their deployment. It is the follow-up to first task 3.1 “Green mobility services definition”, and its corresponding D3.1 [1], where all aspect regarding the services where defined and explained in detail.

Due to rapid urban growth that Europe and the world is suffering [2], smart cities concept is getting even more important. Mobility and Environmental issues are increasing since cities are becoming crowded. GreenMov project aims to address these problems proposing the usage of services that allow cities to improve the management and efficiency of the urban environment.

The developed services are classified into: Air Quality, Noise Annoyance, Traffic and Bikes Availability. In order to give user-friendly information, two service types are defined: forecasting services (this type calculates the level of certain magnitudes given a time-location tuple, such as pollutants or traffic intensity) and calculation services (the aim of this type is to process the outcome of forecasting services to provide useful and simple information to final users). The idea is to deploy the forecasting service followed by the calculation service. Air Quality, Noise Annoyance and Traffic services applies this architecture. Two additional services are defined: Bikes Availability and Traffic Recommendation. The first one is a forecasting service. It does not require a calculation service since its outcome it is already user-friendly. In case of Traffic Recommendation service, it is a calculation service that considers all the rest of predicted information from forecasting services to provide useful recommendations.

All stages from data analysis to model deployment are explained in detail in this document. The structure of service explanations depends on the specific type of service being discussed. In the case of forecasting services, the first section is devoted to statistical analysis of the data. Then, emphasis is placed on the development of the machine learning model, and the explanation concludes with the machine learning operations platform used for deployment. For calculation services, the section on statistical analysis of historical data is removed, and both the model development and the platform used for deployment are simplified.

In the list below, all the set of services that are developed are shown.

- Air quality forecasting: hourly based estimation of pollutants in the air.
- Air quality index calculation: binary categorization of air quality based on pollutants estimation.
- Traffic forecasting: estimation of traffic flow indicators.
- Traffic environmental impact calculation: binary categorization based on traffic flow indicators.
- Traffic recommendation: recommendations for reducing the environmental impact of traffic.
- Bikes availability forecasting: estimation of the availability of bike docks.
- Noise annoyance forecasting: estimation of noise pollutants.
- Noise annoyance calculation: binary categorization of noise annoyance quality based on noise pollutants estimations.

The interconnections between Activity 3 and the other Activities in this project are depicted in Figure 1: Activity Relationships. The data model applied in Activity 3 is specified in Activity 2, "Intelligent Data Models for

Sustainable Transportation". Activity 4, "Improved Context Broker Framework for Handling Concurrent Data-Intensive Situations in Transportation," supplies the necessary infrastructure for Activity 3. Activity 5, "Pilot Deployments," outlines the pilot programs and furnishes past data sets, utilizing the services established in Activity 3. Activity 1, "Project Management," and Task 6, "Creating Impact and Business Opportunities," are cross-functional tasks aimed at project administration and impact creation, respectively.

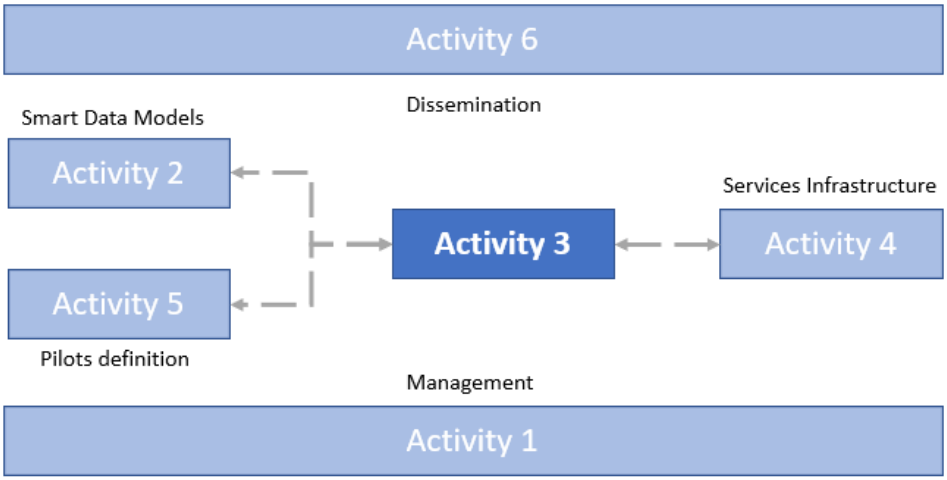

Figure 1: Activity relationship

1.1 Structure of the document

This document is organized into four key sections. We have grouped the sections based on the purpose of each service. The sections include: Air Quality, Traffic Environmental Impact, Bikes Availability, and Noise Annoyance, as well as introductory and concluding sections.

Chapter 2 presents services implementation regarding air quality.

Chapter 3 presents services implementation regarding traffic environmental impact.

Chapter 4 presents the service implementation regarding bikes availability.

Chapter 5 presents services implementation regarding noise annoyance.

2 Air Quality Services

Air quality services are presented in this section. As it is explained in D3.1 [1], basic schema is applied, that is, forecasting service followed by calculation service in order to provide easy-readable information to users. A certain prediction of a given set of pollutants will be output by AQF service. Then, AQC service will process this prediction to get the corresponding air quality index. The whole process is related to a tuple time-location. Both AQF and AQC services works in an hourly based manner.

This service will be deployed in Murcia/Molina del Segura and Nice use cases. The list of pollutants depends on the information handled on each case. In section 2.1.2, the relationship between use case and selected pollutants will be explained. In section 2.1.4.1, the roll of each service when a request from the user is received is further discussed.

2.1 Air Quality Forecasting

2.1.1 Description

AQF service provides an hourly based prediction of a given set of pollutants. Having received a request containing a timestamp and a localization, it will output the corresponding pollutant levels.

In this section we explain the process, from historical data analysis to model deployment. The order of each subsection is the same we followed all along the project. For those steps where the process is different from one use case to another, subsections are added.

2.1.2 Data analysis

2.1.2.1 Historical data

This report will provide a brief introduction to the analysis of air quality data in Murcia and Nice. The two cases will be treated separately, with a detailed examination of each location's air quality data. The goal of this analysis is to provide a comprehensive overview of the data quality situation in both cities.

In Murcia, there are a total of six stations that collect air quality data. However, when analysing the data from these stations, three main problems were identified. Firstly, there is a significant amount of missing data, as shown in the table (see table below). Secondly, there is a noticeable character of noise in the signals, which can make it difficult to accurately interpret the data. Lastly, there is a clear bias in the scale of the data from month to month, which can further complicate the analysis of the air quality situation in Murcia.

Table 1: Missing data for HOPU (Murcia) stations

Pollutant	NO	NO_2	O_3	SO_2	CO	PM_{25}	PM_{10}
Missing values (%)	20 %	26 %	26 %	26 %	25 %	20 %	20 %

As evidence of the two last problems, we present several graphs that we believe best demonstrate the character of noise and bias in scale. These graphs are taken from the station with the identifier 'HOPac67b2cd2caa', which we consider to be the greatest example of these issues.

Firstly, we would like to introduce a time series of PM10 data from the station 'HOPac67b2cd2caa'. This series highlights the biased scale effect, which can be seen in the fluctuations of the data over time.

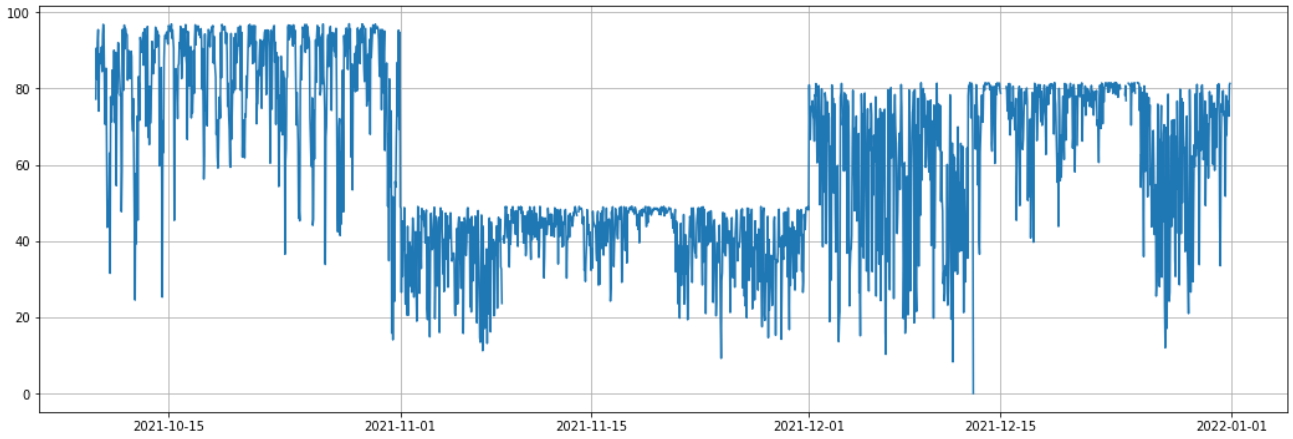

Figure 2: PM10 time series for a station in Murcia.

Secondly, we would like to introduce a differentiated time series of PM10 data from the station 'HOPac67b2cd2caa'. Despite the first autoregressive term being non-zero as a result of the differentiation, the noisy character of the data remains evident. Both time series highlight the fluctuations and irregular patterns in the PM10 data, making it difficult to identify clear trends and patterns in the air quality over time.

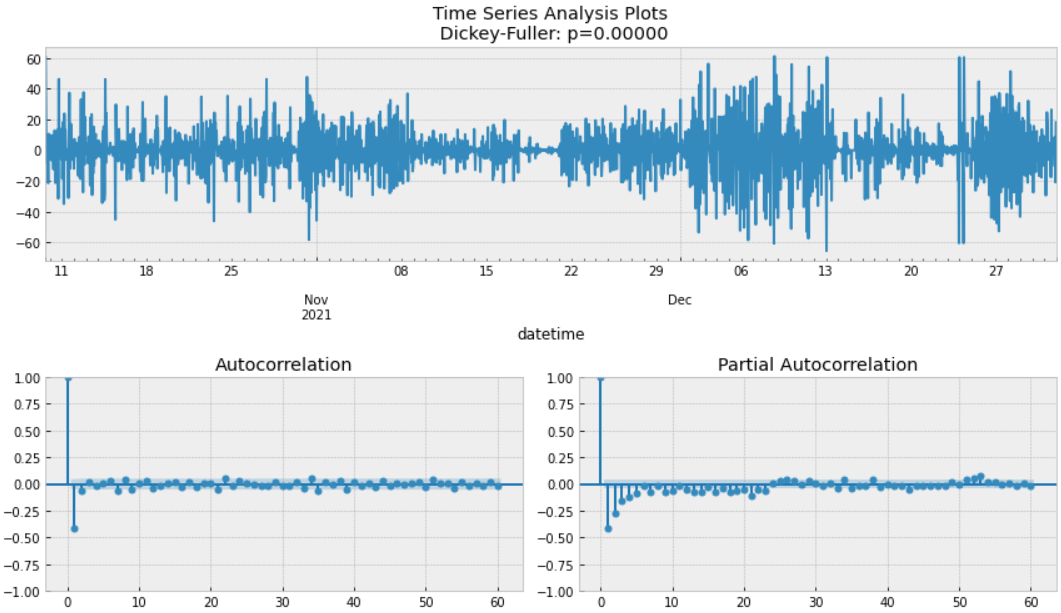

Figure 3: PM10 differentiated time series for a station in Murcia.

Given these limitations and challenges presented in this analysis of air quality data in Murcia, HOPU is developing a microservice to mitigate the impact of this anomalies on the algorithm. Additionally, the model design will be based on the data from Nice, which we believe will provide a more reliable and accurate picture of the air quality situation.

In the case of Nice, we have access to data that dates to the year 2020 for the following air pollutants: NO, NO2, NOX, PM10, and PM25. We can note that the forecasting model will generically consider other remaining pollutants, to a flexible solution.

In the Nice air quality dataset, there was no missing data initially, however, some data points may have been discarded during the pre-processing stage. The distribution of each pollutant is presented in the following figures.

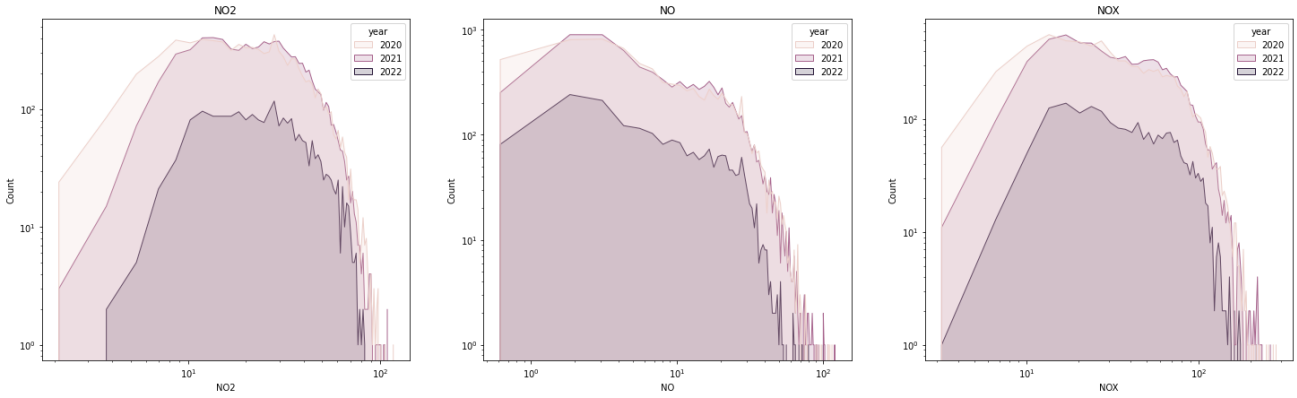

Figure 4: NO2, NO, NOX empirical distributions.

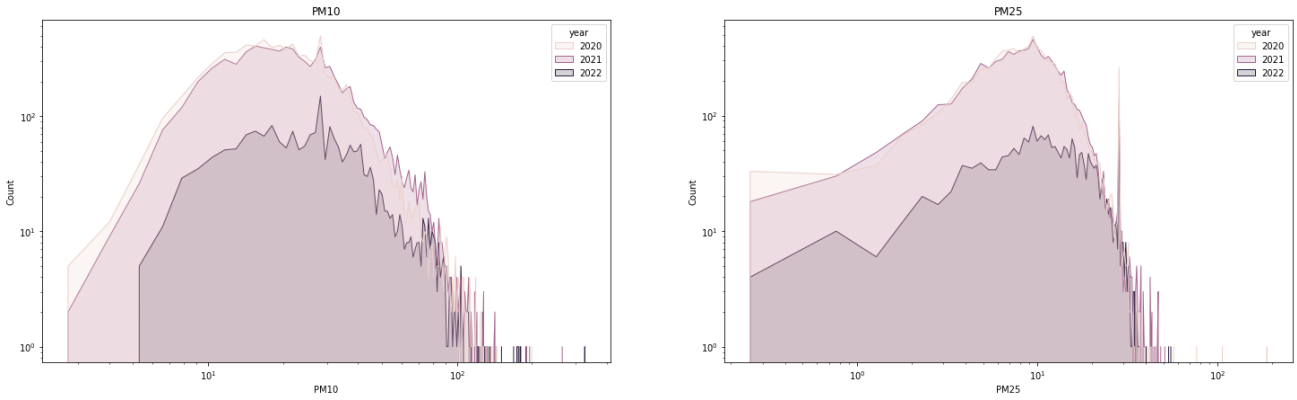

Figure 5: PM10, PM25 empirical distributions.

The distributions of the pollutants are within expected ranges and display a pseudo gaussian character. Additionally, note the anomalous accumulation observed for PM25 around the value of '28.3'.

2.1.2.2 Pre-processing

The purpose of this stage is to ultimately create a reliable and organized table of data, ordered chronologically and evenly spaced in time. To ensure the reliability of the data, we will check that the data is in the correct format and eliminate anomalous accumulations. This pre-processing step is crucial for the success of the analysis, as it will provide the foundation for building accurate and trustworthy models. It will be done automatically all the time new data is received.

In addition to this, we incorporate outlier detection and elimination, based on the interquartile range [6]. The following table shows the percentage of outliers in the dataset of Nice. This step helps to reduce the noise and improve the accuracy of the data.

Table 2: Outliers percentage in the Nice pollutants dataset.

Pollutant	NO	NO_2	NO_X	PM_{25}	PM_{10}
Missing values (%)	0 %	0 %	0%	0.02 %	0.1 %

As a conclusion of this pre-processing stage, the following table shows the amount of information eliminated during this step on the dataset.

Table 3: Discarded data in the Nice pollutants dataset.

Pollutant	NO	NO_2	NO_X	PM_{25}	PM_{10}
Missing values (%)	0 %	0 %	0%	2 %	0.1 %

2.1.3 Model Development

In this section, we introduce the prediction algorithms and discuss their parameters and the method used to obtain them. To ensure accurate and reliable predictions, various algorithms have been considered. The selection of the best model is made based on the evaluation of their performance through statistical metrics. The parameters of each algorithm are fine-tuned using techniques such as Grid Search and Cross-Validation.

2.1.3.1 Model description

Among the models tested, we found Autoregressive Integrated Moving Average (ARIMA) , SARIMA (seasonal autoregressive integrated moving average) modeling with exogenous factor (SARIMAX) , Vector autoregressive moving-average (VARMA), Random Forest Decision Trees, Holt-Winters, and Exponential Smoothing. Finally, we settled on Gradient Boosted Decision Trees. Below, we briefly justify this decision.

The first criterion is strictly related to the quality of prediction. In this sense, both ARIMA and gradient boost decision trees outperformed over the rest of the models.

Additionally, the design of the models should be straightforward in terms of complexity and resource management. The XGBoost library provides us with support for these requirements. By using XGBoost, we can simplify the implementation and management of our models, while still maintaining high prediction accuracy.

With all of that and given that our event space for the pollutants consists of positive real numbers, we have chosen to utilize a boosted decision tree regression model specifically.

2.1.3.2 Training

The model employed is a multi-variable decision tree. As input, the model considers samples with lag from the temporal series we aim to predict. We leave open the possibility for the future incorporation of other meteorological variables.

Upon studying the PACF (Partial Autocorrelation Function) and ACF (Autocorrelation Function), we obtain the most relevant lags for each contaminant. Intuitively, the PACF and ACF are statistical measures that assess

the correlation between an observed time series and its own lagged values. The following table summarizes the results.

Table 4: Considered lags in the AQF model.

	Lag 1	Lag 2	Lag 3	Lag 24	Lag 168
NO	X	X	X	X	X
NO_2	X	X		X	X
NO_X	X	X	X	X	X
PM_{25}	X	X	X	X	X
PM_{10}	X	X	X	X	X
Generic	X	X		X	X

After conducting a grid search (more information regarding to the test phase is included in the next section) for the model, we have determined that the optimal parameters for the model utilizing the available dataset are as follows:

Table 5: AQF model parameters.

Parameter	*n_estimators*	*max_depth*	*learning_rate*	*min_child_weight*	*gamma*
Description	The number of trees in the forest.	Maximum depth of a tree.	Step size shrinkage has used them in update to prevent overfitting.	Minimum sum of instance weight(hessian) needed in a child.	Minimum loss reduction required to make a further partition on a leaf node of the tree.
Value	83	5	0.09	3	1

2.1.3.3 Results

In a time series context, the traditional train-test split method is different from the standard approach because the data is divided chronologically. The objective is to ensure that the predictions are not biased by having information from the future in the training set. This means that the training set consists of data that precedes the data in the test set, so the model can only use information from the past to make predictions.

The chronological split allows for a more realistic evaluation of the model's performance, as it mimics the real-world scenario where only historical data is available for making predictions.

We will be using the R-squared metric to evaluate the performance of our model. R-squared, also known as the coefficient of determination, is a statistical measure that indicates the proportion of the variation in the dependent variable that is explained by the independent variables.

Table 6: R-squared metric in AQF.

Pollutant	NO	NO_2	NO_X	PM_{25}	PM_{10}
R^2	0.75	0.82	0.79	0.78	0.77

We also include, for complete, the results obtained when performing a forecast on the Murcia's stations dataset. Based on what was explained in the introduction, we should not consider the following obtained results as a validation criterion for the service.

Table 7: R-squared metric in AQF for Murcia.

Pollutant	CO	SO_2	NO	NO_2	PM_{25}	PM_{10}
R^2	0.16	0.17	0.46	0.49	0.67	0.30

2.1.4 Model Deployment

Once the model development step is ready, it is time to design and implement a Production platform that provides features to cover all phases that take place: Data Preparation, Model Creation and Rollout.

Regarding Data Preparation, the first step will be retrieving data from the context broker and processing it to be ingested by the system. Then, the analysis and preprocess defined in 2.1.2.2 is applied. At this point, data is ready and stored to be used by the system.

Model Creation phase covers all process from training to registering a new model. After loading data, the model is built and trained, as well as validated before the registering step. All the processes are monitored, so developer can debug them.

Now that the model is created, it is time for deployment. The goal is to give the stakeholder (corresponding Calculation Service 2.2) a way to request a prediction given a time-location tuple.

2.1.4.1 Workflow

In Figure 6, the schema followed is clearly defined. As stated above and shown in the schema, workflow is composed of three main phases: Data Preparation, Model Creation and Rollout. The rest of the section addresses each phase in detail.

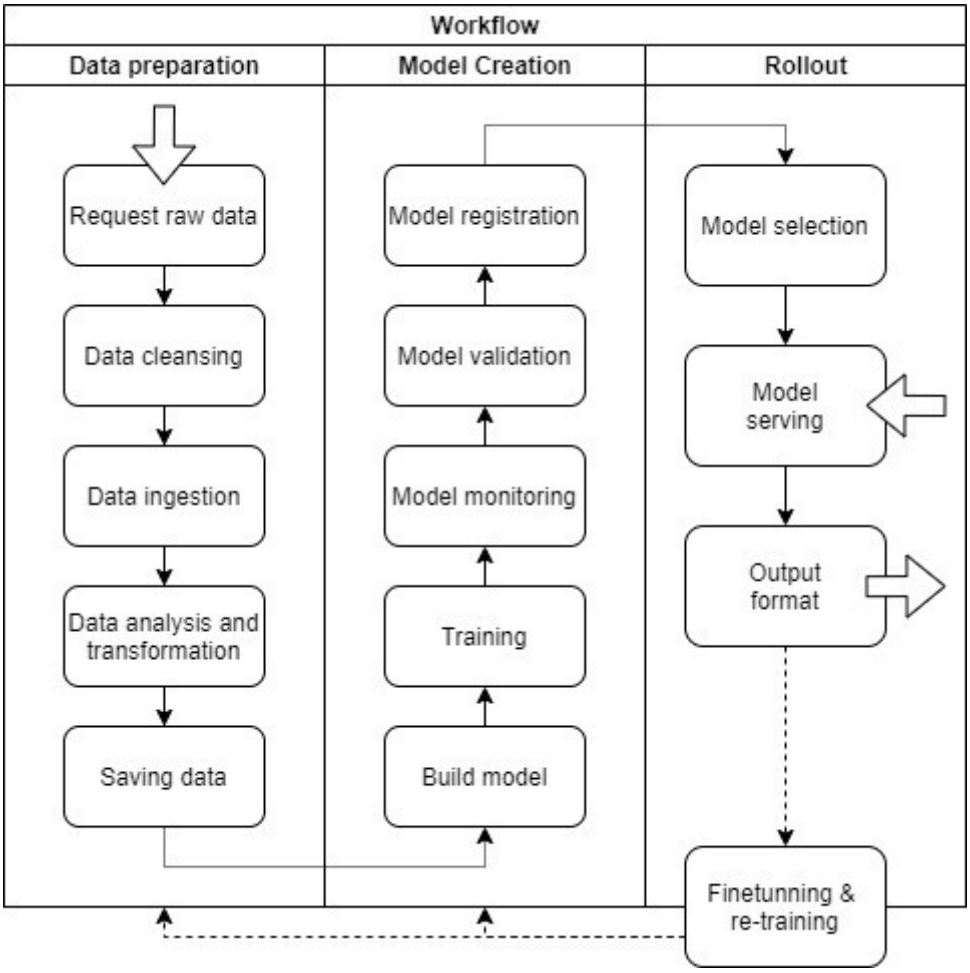

Figure 6: Workflow AQF service

Data Preparation

Data preparation is the process of cleaning, transforming, and organizing data into a format that is suitable for training a machine learning model. This stage is critical because the quality and structure of the data used to train the model has a direct impact on the performance of the model. Below, the steps shown in the Figure 6 are described, respecting the chronological order in which they occur.

1. Request raw data. Execute the corresponding query against the context broker to obtain new data.
2. Data cleansing. Transform the payload of the request to remove any useless information for the system.
3. Data ingestion. Add new records to the data table. Duplicates are deleted.
4. Data analysis and transformation. Apply the preprocess defined in 2.1.2.2.
5. Saving data. Store a new version of the table.

At this point, the historical data is up to date, meaning that all available data will be considered in the next phase, Model Creation.

Model Creation

Model Creation is the process of building, training and validating a machine learning model. In this specific case, because of model development phase 2.1.3, the XGBoost machine learning model will be used.

1. Build model. Initialize XGBoost model.
2. Training. In this step a gridsearch approach is applied.
3. Model monitoring. Supervise training step in real time.
4. Model validation. Verify model performance.
5. Model Registration. Register and store the latest version of the model. 48-hours prediction are calculated and stored along with the model.

Once this phase is finished, it is time to put the machine learning model in production.

Rollout

This refers to the process of deploying a trained model into a production environment and making it available for actual use by end users.

1. Model selection. It is the process of selecting machine learning model that will be served through an API. In this case, the criteria is the most recently trained model.
2. Model serving. Deploy a REST API to allow requests from users.
3. Output format. Format predicted values to be compliance with SDM.

Once this phase is completed, the deployment of the model is considered finished. Nevertheless, one of the objectives is to use the data's potential in real-time as efficiently as possible through the infrastructure provided by the use cases. Thus, both daily retraining and periodic data updates have been designed.

Daily retraining is triggered at night, when the service has low demand, and the Data Preparation, Model Creation, and Rollout phases are executed sequentially. While the most optimistic case would be to retrain the model every hour, the infrastructure doesn't have unlimited resources, and the impact on forecasting accuracy would not be significant. Therefore, periodic data updates have been introduced.

In this process, Data Preparation, a few steps from Model Creation, and the Rollout phase are executed. The purpose is to use real-time data without consuming too many resources. Instead of creating a new model, a new set of predictions are made and stored along with the existing model. For clarity, a new version of the model is registered.

The aim is to deliver a Docker image for each service that can be used as a module in a larger system. However, in order to use the architecture where the Forecasting service is followed by the Calculation service, it is required to orchestrate the entire process. In this case, the Forecasting service will act as the master and send the predictions to the Calculation service, which will then merge and deliver all information to the user. An environmental variable is configured to enable or disable this step so that both services are independent.

In the next section all the technical aspects and selected technologies to accomplish requirements are further discussed.

2.1.4.2 Architecture

The technologies selected to orchestrate all the processes that take part in this service are:

- **MLFlow [9]** . This tool allows us to implement Model Tracking and Model Monitoring steps. It offers a useful user interface to monitor all the aspects that take part in model creation phase, such us train parameters, train metrics and model registration.
- **KServe [12]**. It is the selected Model Inference Platform. It provides tools to accomplish Model Serving step, as it offers several linked APIs to preprocess data, make the inference and postprocess it.
- **Crontab [13].** The scheduler is required since it is necessary to trigger actions like Model Creation and Data Preparation in each time with specific frequency.

All of the above is encapsulated in a single Docker Image as it is shown in Figure 7. There are three different processes: Preprocess (yellow), Training (purple) and Inference (blue).

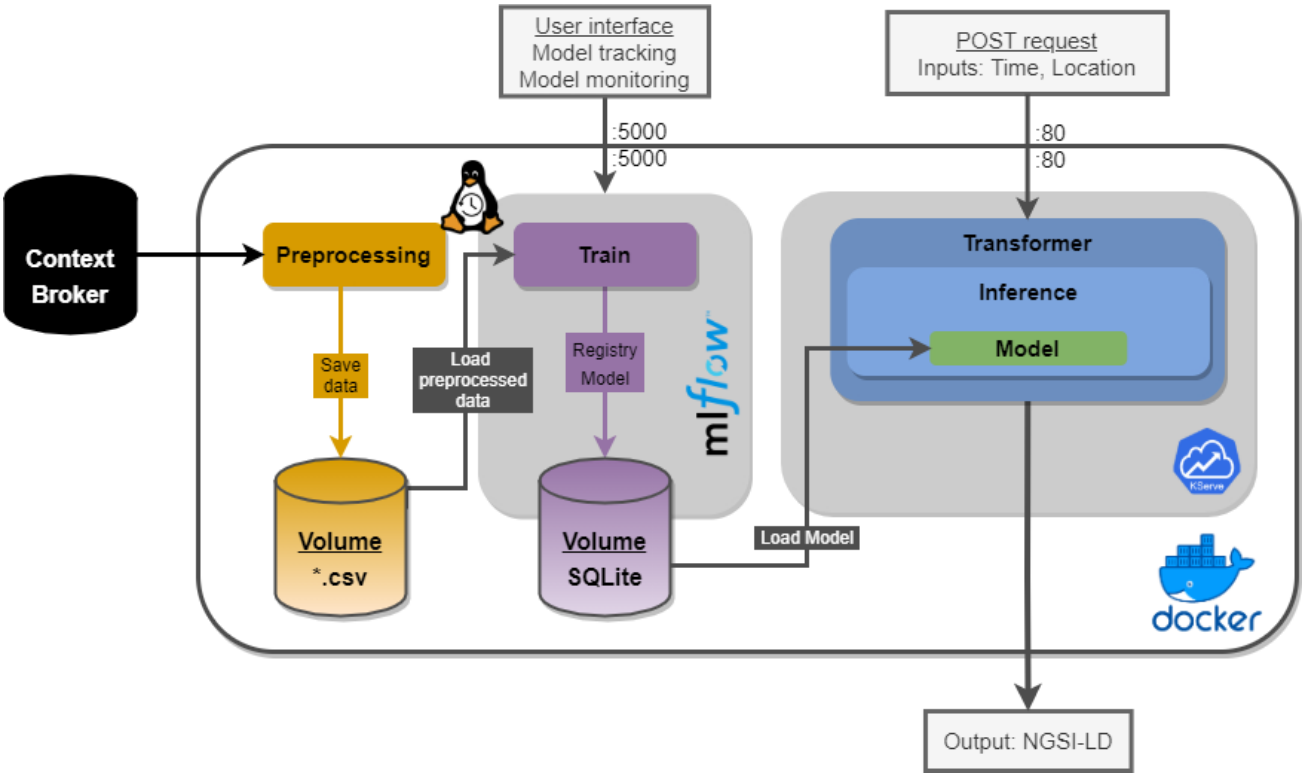

Figure 7: Architecture AQF service

The Preprocess phase refers to the Data Preparation stage described in the last section. In this stage, new data is requested from the corresponding context broker and then undergoes cleaning and preprocessing before being appended to the historical data stored. The result is then stored, creating a new version of the table. This new version is loaded for the training step, which executes all the steps outlined in the Model Creation phase to produce a new model registered in the SQLite database. This new model is selected for inference because it meets the criteria of being the last model trained. The system is now able to respond to requests through the inference platform.

The relationship between the training step (using MLFlow) and the inference step (using KServe) is maintained through the database, where all models are registered. Upon receiving a request, the last trained model is loaded into the inference platform from the SQLite database, making the two services independent of each other. While

the training step is executed, users can still make requests, but, in any case, it is scheduled to run at late-nights on a daily basis.

To avoid multiple training sessions per day and minimize computational costs, the update process defined in the last section considers real-time data by updating the data inside the last trained model every hour. This updated model is registered in the same manner as a new model.

Two routines are defined in the schedule, controlled by Crontab. The first routine is the training routine, which executes the Preprocess and Train phases once a day at late-night. The second routine is the update routine, which has hourly frequency and executes the Preprocess and few steps of the Model Creation phase. MLFlow, KServe, and Crontab are executed at startup.

To ensure the persistence of data and trained models, Docker volumes are used. The .csv file containing the historical data is stored in this path, ensuring it is not deleted if the service goes down. The SQLite database configured in the volume path stores all the information required to register the last trained model upon startup.

2.1.5 Conclusions

At this stage, the entire process from analyzing the historical dataset to deploying a fully functional machine learning model in production has been outlined. Firstly, data analysis provides us with crucial insights into the quality of the data. Once this is confirmed, we move on to selecting the appropriate models and data preprocessing techniques to meet our performance indicators. In the case of the Murcia/Molina use case, we encountered several data issues, so we shifted our focus to the Nice use case, which utilizes only one station. However, it's possible to deploy multiple services to accommodate multiple stations.

Concurrently, the use cases communicated their expectations, prompting us to identify technologies and processes that would aid in this endeavor. MLFlow and KServe are noteworthy examples of these cutting-edge technologies designed to streamline machine learning operations. Thus, we chose to implement them. To maintain consistency and ease of deployment, all components have been packaged into a single Docker image, in accordance with the agreement made at the start of the project that task 3.2 would produce a single Docker image. In the future, a microservice architecture may be incorporated.

2.2 Air Quality Index Calculation

2.2.1 Description

This service calculates Air Quality Index and Air Quality Level, based on the level of pollutants, as dictated by European normative. It receives the predictions from the Air Quality Forecasting service and outputs an AirQualityForecast instance with the corresponding Air Quality Index and Air Quality Level.

2.2.2 Model Development

2.2.2.1 Preprocessing

When a request is made, this service checks that the AirQualityForecast entity is correct and has all the required fields. This should be a JSON-LD instance which the fields corresponding to the AirQualityObserved datamodel.

2.2.2.2 Rules

This service calculates Air Quality Index and Air Quality Level using the rules defined by the European Environment Agency [3]. The rules can be seen in the following table:

Table 8: Rules for Air Quality Level calculation

	INDEX LEVEL (ug/m3)					
	Good	Fair	Moderate	Poor	Very poor	Extremely poor
PM2.5	0-10	10-20	20-25	25-50	50-75	75-800
PM10	0-20	20-40	40-50	50-100	100-150	150-1200
NO2	0-40	40-90	90-120	120-230	230-340	340-1000
O3	0-50	50-100	100-130	130-240	240-380	380-800
SO2	0-100	100-200	200-350	350-500	500-750	750-1250

The Air Quality Index and Air Quality Level of each pollutant combines in one only indicator by taking the less favourable of all of them.

2.2.3 Model Deployment

2.2.3.1 Architecture

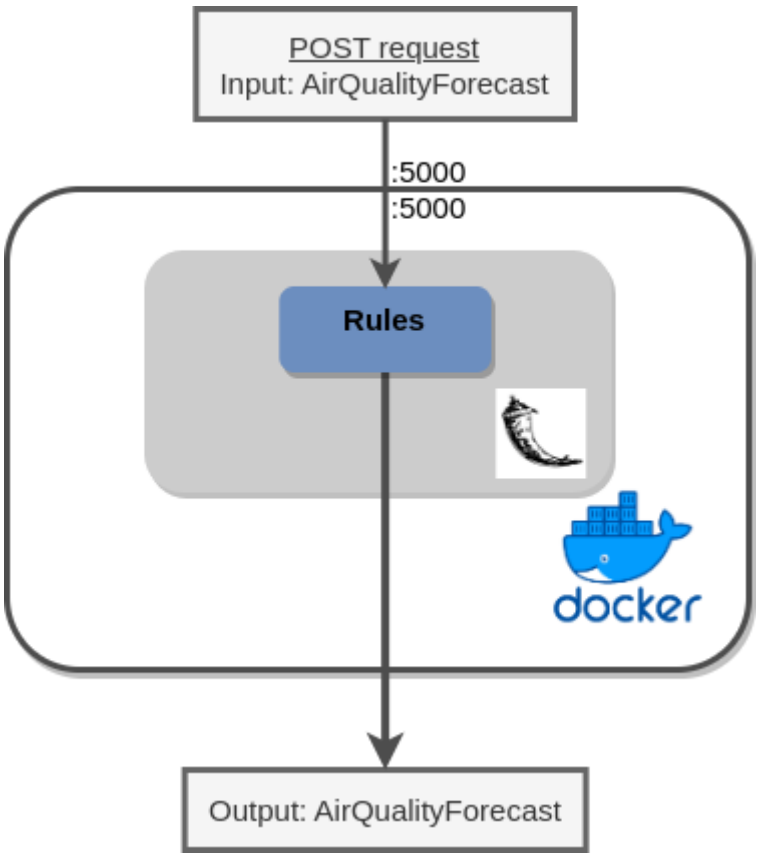

Figure 8: Air Quality Index Calculation service architecture

We use Flask [10] to define a Rest API. This is encapsulated in a docker container.

2.2.3.2 Workflow

1. The service receives a request.
2. The service checks the body of the request to see if it contains a correct AirQualityForecast entity.
3. The service applies the calculation rules and fills the corresponding fields of the AirQualityForecast entity with the results.
4. The service returns the completed AirQualityForecast entity.

2.2.4 Conclusions

The service developed calculates Air Quality Index and Air Quality Level from an AirQualityForecast entity following European rules. The service can be easily deployed thanks to Docker.

3 Traffic Environmental Impact Services

Traffic environmental impact services consist in an assessment of the impact of traffic on the environment, and especially air quality, at a given location and time in the future. This service is used within GreenMov by Nice and Murcia Molina use cases. An overall architecture of the service is proposed in Figure 9: Main architecture for Traffic environmental impact services .

The traffic environmental impact services are split in two main services. The first one is the traffic forecasting, whereas as the second one corresponds to the environmental impact calculation. The process of these services are the following, as shown in Figure 9: Main architecture for Traffic environmental impact services :

1. Data is collected from sensors and cameras to provide information on the number of cars that constitute the traffic at each time of the day. This includes the classification in type of vehicles, that are ranged according to their level of particles and CO2 emissions. The data also includes the noise associated with each vehicle type category
2. Then, a forecast of future traffic density is done by the traffic forecasting service. This forecast mostly provides the number of vehicles per category at a given time in the future.
3. Finally, the environmental impact calculation uses this forecast and the data associated with the environmental impact of each vehicle category to compute the environmental impact, including the noise impact and the particles emissions.

This traffic environmental impact is triggered by an API request that must specify the time and location of the forecast.

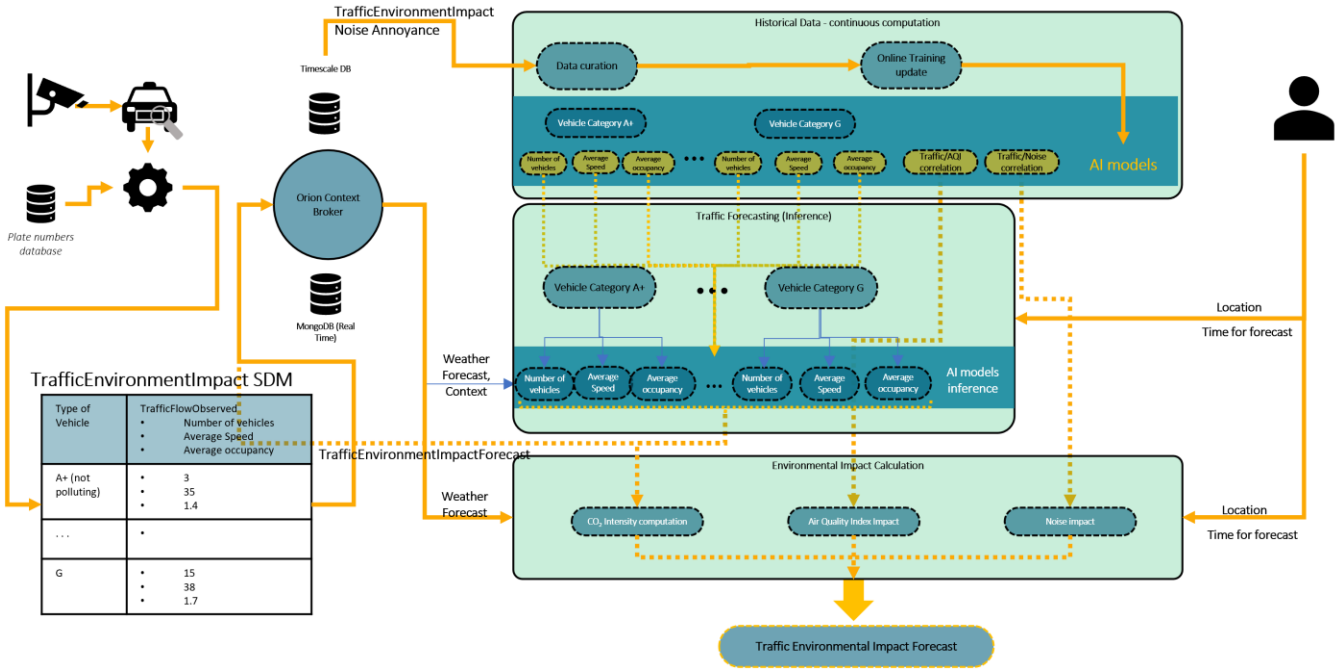

Figure 9: Main architecture for Traffic environmental impact services

3.1 Traffic Forecasting

3.1.1 Description

The service offers a traffic flow prediction for a designated location, at a specific time T requested by the user of the service. This prediction is based on the analysis of previous historic traffic data, providing valuable information for travel planning and recommendations generation. By setting T equal to 0, the prediction returns the current traffic situation, allowing real-time monitoring of the traffic flow.

3.1.2 Data analysis

3.1.2.1 Historical data

The AI model that powers the service is trained on a range of historical data, that has been augmented with various other characteristics than traffic flow. This augmented data set considered for forecasting the traffic consist of the following data:

- The intensity of traffic per type of vehicle;
- The average speed per type of vehicle;
- The car occupancy per type of vehicle;
- The emissions range per type of vehicle;
- The temperature;
- The humidity levels.

The intensity of the traffic depends on several factors, but mostly depends on the time of the day. An overview of gathered data in Nice is proposed in Figure 10. Except from the obvious sensor issues that led to many data close to 0, we can see that there is a high variability of data.

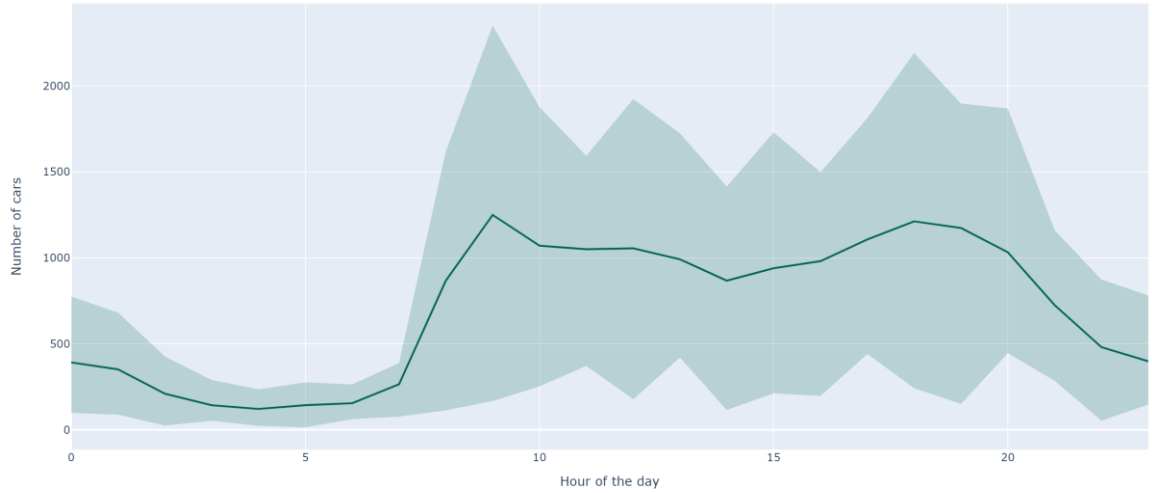

Figure 10: Traffic intensity data overview

Furthermore, the type of day has also a considerable impact on the traffic intensity, as we can see in Figure 11

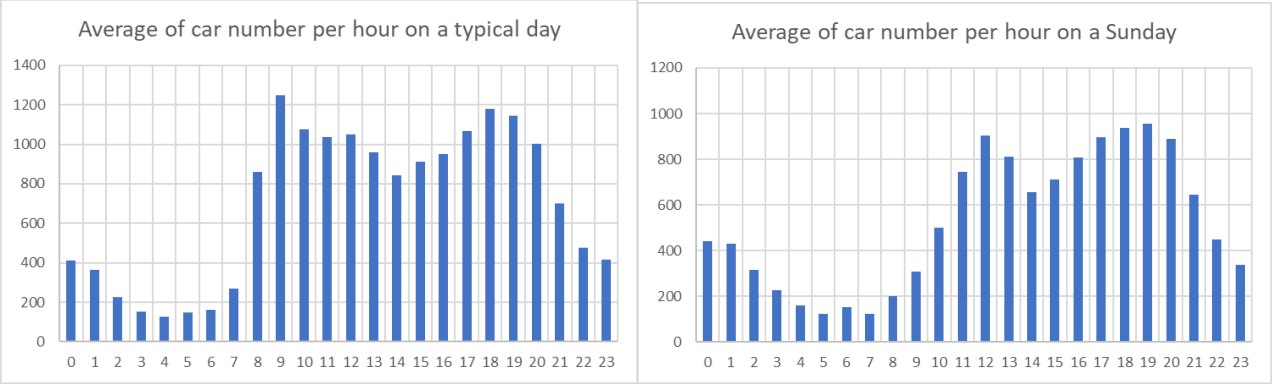

Figure 11: Impact of the type of day on the traffic intensity

From this data, we can then infer new data to further augment the dataset, as for example, we can include the previous traffic intensity (1hour ago, 2 hours ago, ... 168 hours ago). An example of correlation between the main traffic intensity data and side parameters is shown in Figure 12, which highlights the fact that the traffic intensity is mostly dependent on previous traffic intensity, and on the time of day.

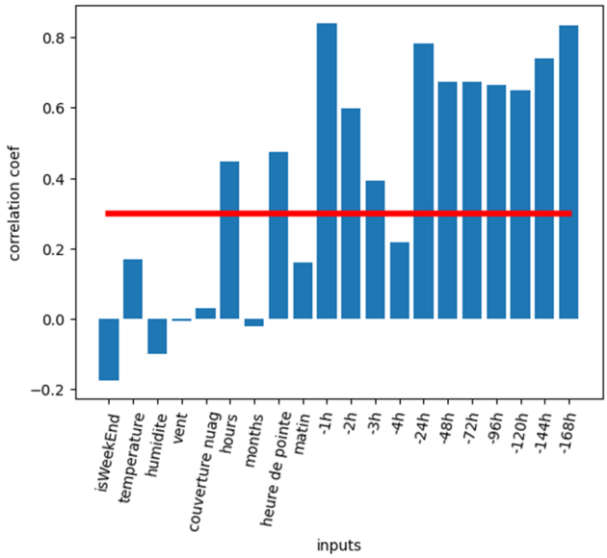

Figure 12: Correlation between outputs (traffic intensity) and potential features (input parameters)

By taking into account these multiple factors, the AI model is able to provide an accurate prediction of the traffic flow at a specific location and time.

3.1.2.2 Preprocessing

In the context of this traffic flow prediction service, the data sets are very diverse and complex. As a result, the data cleansing and pre-processing steps play a critical role in ensuring that the AI model is trained on high-quality data.

One of the key challenges in pre-processing the data is to ensure that it is properly balanced and representative of the diverse conditions that can impact traffic flow. This includes considering factors such as seasonality, weather patterns, and special events that may have a significant impact on traffic. By carefully pre-processing

the data, the service can ensure that the AI model is able to account for these various factors and produce accurate predictions. Pre-processing of the data included:

- Alignment of time for each of the considered dataset, in order to ensure synchronicity between the datasets
- Cleaning of the time array for all dataset, which consists in filling small gaps in the dataset using extrapolation methods
- Outliers removal, which consists in removing data that are out of range based on the difference with the average value at the same time.

This data pre-processing steps led to the removal of 3.9 % of the data in the dataset of Nice for example.

Then, the service needs to perform regular updates and retraining of the AI model to account for any changes in the data, traffic patterns, and other factors that may impact traffic flow. By continuously improving and refining the data sets, the service can maintain its accuracy and reliability over time.

3.1.3 Model Development

3.1.3.1 Model description

The development of the traffic flow prediction service is a crucial aspect of ensuring its accuracy and effectiveness. The AI model must be able to effectively analyse the historical traffic data sets, considering the various factors that impact traffic flow, and make predictions about future traffic conditions. To ensure this, the service must perform several key steps, including data preparation, model selection, and model training, the service must choose an appropriate AI model that is capable of handling the complexity of the data and the variety of factors that impact traffic flow. This may involve evaluating several different AI models and selecting the one that provides the most accurate predictions. The models that were benchmark in the case of traffic forecasting include K-Nearest Neighbours (KNN), Decision Tree, Random Forest and XGBoost. By comparing the performance of these models using different historical data sets, it was determined that KNN outperforms the others with an accuracy above 82%. While XGBoost also produced good results, it required more time for parameter optimization. Therefore, KNN is deemed the best choice for this application. In terms of features , the model takes the following inputs to provide a forecast:

- Type of day (week or weekend).
- Hour of the day.
- Traffic at the following lags (hours before):
    - 1h, 2h, 3h, 24h, 72h, 96h, 120h, 144h and 168h.

The outputs of the model consist in all the traffic intensity values, from the time at which the request was sent to the traffic intensity at the requested time and location. Finally, each model corresponds to a unique location. Therefore, in the case of GreenMov, there is one model for each of the locations of the use cases that require this service.

3.1.3.2 Training

To optimize the model, the training was based on a train/test split of 70% and 30% for cross-validation. The model was implemented using scikitlearn [15] library, and the training was realized through the fit function. Then, a benchmark on the training data requirement was done in order to assess the required amount of data we

need to use to train our models. For kNN model, it was shown that the best accuracy was obtained when training with 185 days datasets. This allows the services to only store 185 days of data.

3.1.3.3 Results

After evaluating various models for forecasting the traffic flow on the Promenade des Anglais, KNN was chosen as the best option with an accuracy above 82%, computed using R^2 value, as shown in Figure 13 and Figure 14. The results showed that KNN outperformed other models and was able to accurately predict the number of cars for the coming day, at a very low computational cost.

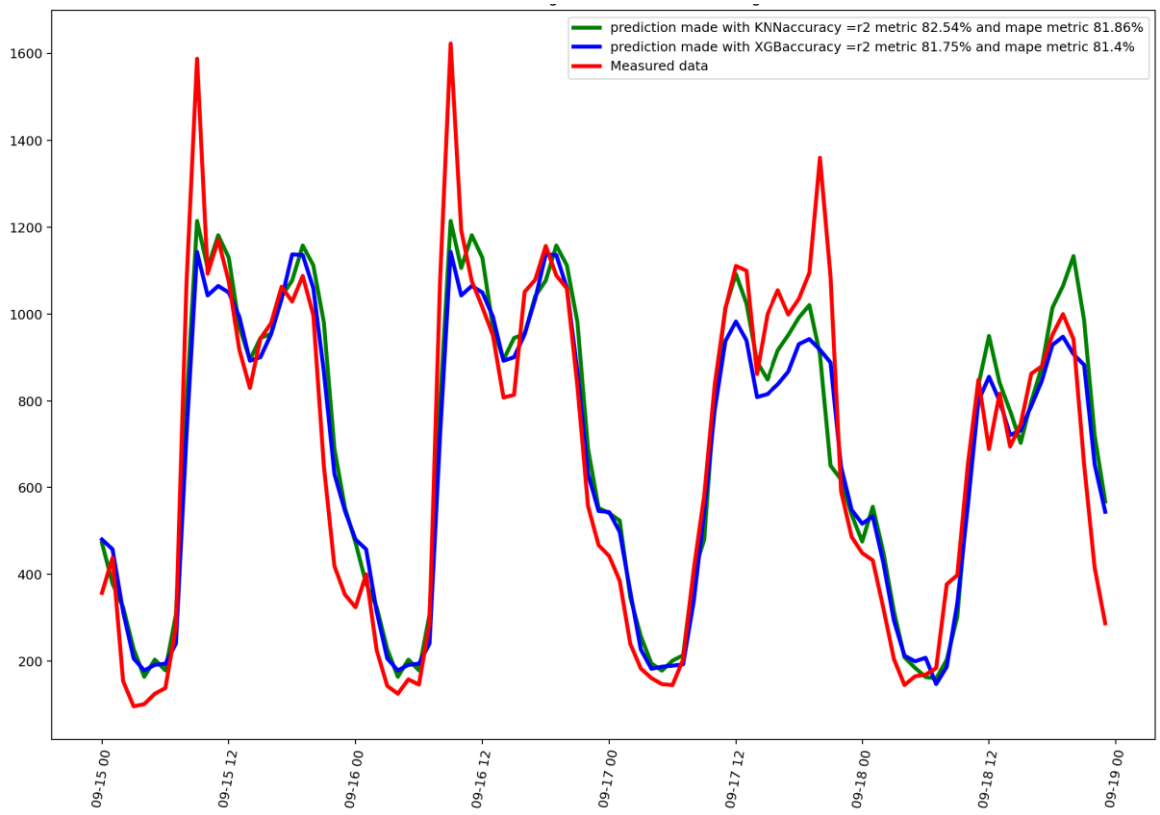

Figure 13: Comparison of accuracy of two different algorithms (kNN and xgboost)

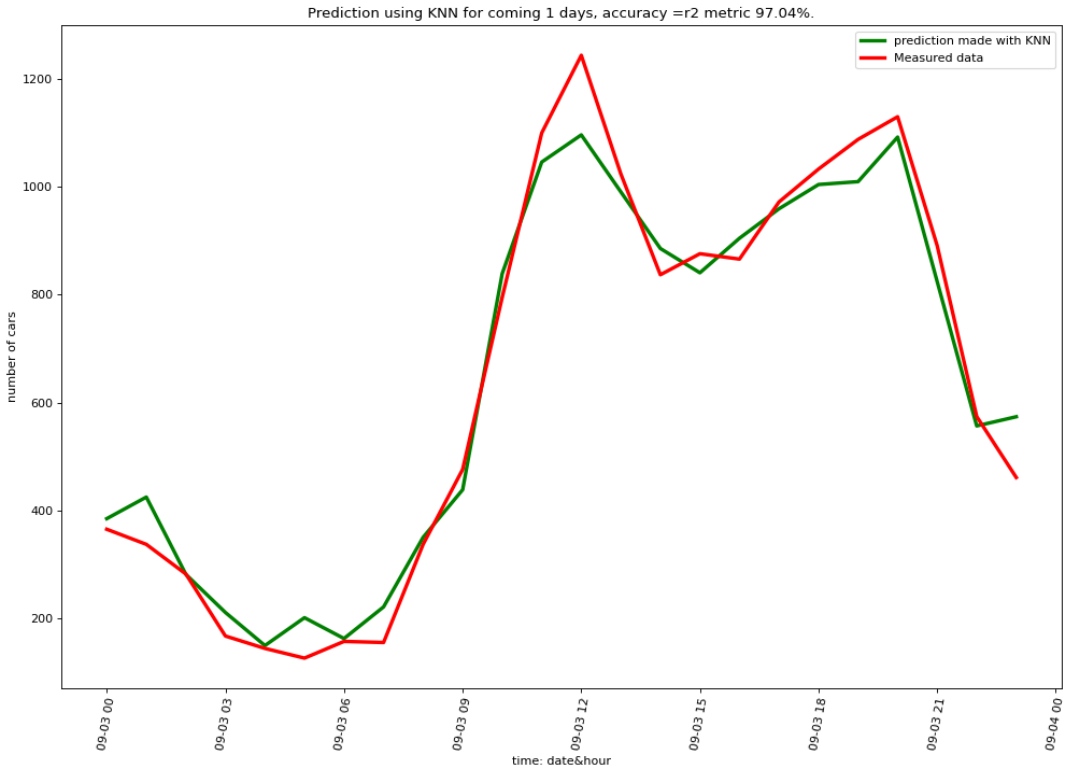

Figure 14: Predicted VS Measured Traffic data for KNN model In Nice.

Figure 14 shows the number of vehicles in that will constitute a predicted traffic vs the real number of cars for the coming 24h (obtained on the test dataset).

3.1.4 Model Deployment

Once the model was designed, the use cases have been able to implement it in their infrastructure. This subsection presents the process followed for service deployment.

3.1.4.1 Workflow

The workflow that was followed to achieve the final model is presented below, in Figure 15. First, data was gathered in order to be able to analyse the data and determine the traffic forecasting model. Therefore, data was gathered from the use cases. It was then cleaned (as explained in the pre-processing section), augmented with complementary data or information (such as last hours data, type of day, ...). These steps are gathered in the data preparation section.

Then, in the model creation phase, several models, features and training requirements were assessed in order to find the optimal combination. One model is designed for each location. Models that were tested include K-Nearest Neighbours (KNN), Decision Tree, Random Forest and XGBoost. Then, the features (inputs) that were selected consist in the type of day, and previous traffic intensity (of the last hour, the 2 last hour, until 168 hours ago). The training can use years of data, or can be restricted to a few weeks or months, in order to not capture specificities from summer when requested to provide a forecast for winter. Therefore, a benchmark of training dataset's size was done to determine the best size of training dataset to use in the selected forecasting solution.

These different benchmarks led us to a specific combination of model, training size and features that will be used in the traffic forecasting service.

Finally, in the roll out phase, we leverage the architecture, the inputs and the training requirements to provide a real time solution that determines the predicted traffic intensity at the required time. The working steps of a forecast computation are as follows:

- First, from the requested location of the forecast, the model that corresponds to the location is selected.
- Then, the last N data are requested to the context broker in order to retrieve the traffic intensity at the previous times (last hour, last 2 hours, last 24 hours,...).
- The traffic intensity of the next hour is predicted.
- Using this forecast as an input of the next model's inference, the model is used to compute the traffic intensity forecast in the next 2 hours.
- This new forecast is used as an input to forecast the traffic intensity in 3 hours.
- And this process is repeated until the time of the requested forecast is reached.

The result of the final forecast (for the requested time) is then formatted as NGSI-LD [7] and fed back to the user.

Finally, at the end of each day, the daily data is retrieved from the context broker and the model is trained again using the new training dataset that consists in the previous training dataset minus one day, and plus the last data from the last day. The resulting updated parameters of the model are then stored using joblib [14] to be used for next day's forecasts.

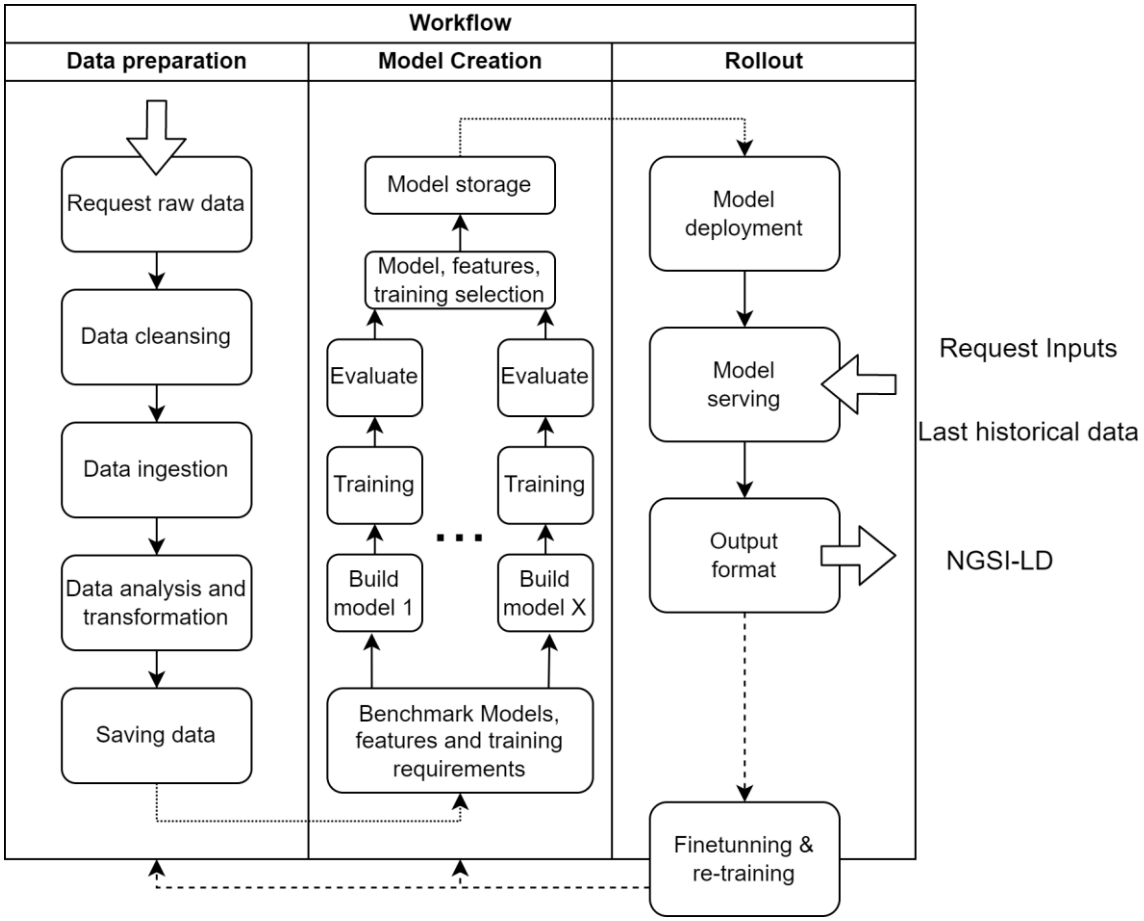

Figure 15: Workflow for Traffic forecasting service for each location

In the next section all the technical aspects and selected technologies to accomplish the functions described are further discussed.

3.1.4.2 Architecture

Figure 16 shows the overall architecture of the traffic forecasting service. The whole service is dockerized using docker-compose for fast and easy replication, monitoring and maintenance. FastAPI [11]is used to provide an API to the users of the service. When FastAPI service receives a request for a forecast, it launches a set of functions that will retrieve the models parameters corresponding to the location requested using joblib library. If no model exist for this location, then a kNN model is built in real-time with a choice of *k* equal to the average of the kNN models used for forecasting at close locations and trained with training data (historic values) requested to the context broker. The efficiency of kNN models make it possible to run such a process in real-time. If the model already exists, the traffic forecasting service will request the last values from the context broker to get the current values of the traffic, that will be used for the inference. Then, the inference phase consists in a series of inferences, from an inference of the next hour's traffic intensity to the traffic intensity at the requested forecasting time. Therefore, the traffic intensity of all the intermediary times (hours) between the time at which the request was sent and the time for which the forecast was asked for are predicted. All these intermediary predicted values are sent to the context broker for storage in case it can be used by other users.

Once the desired forecast time is reached , the data is converted in the right NGSI-LD [7] format and sent back to the end-user.

Then, at the end of the day, a cron job ensures that the model of each location is retrained to capture the trend changes. The updated parameters of the model used are stored using joblib library.

Therefore, inputs of the service are the time, location of the forecast, but also the last data from traffic intensity. Then, the output consist in all the traffic intensity values, from the time at which the request was sent to the traffic intensity at the requested time and location.

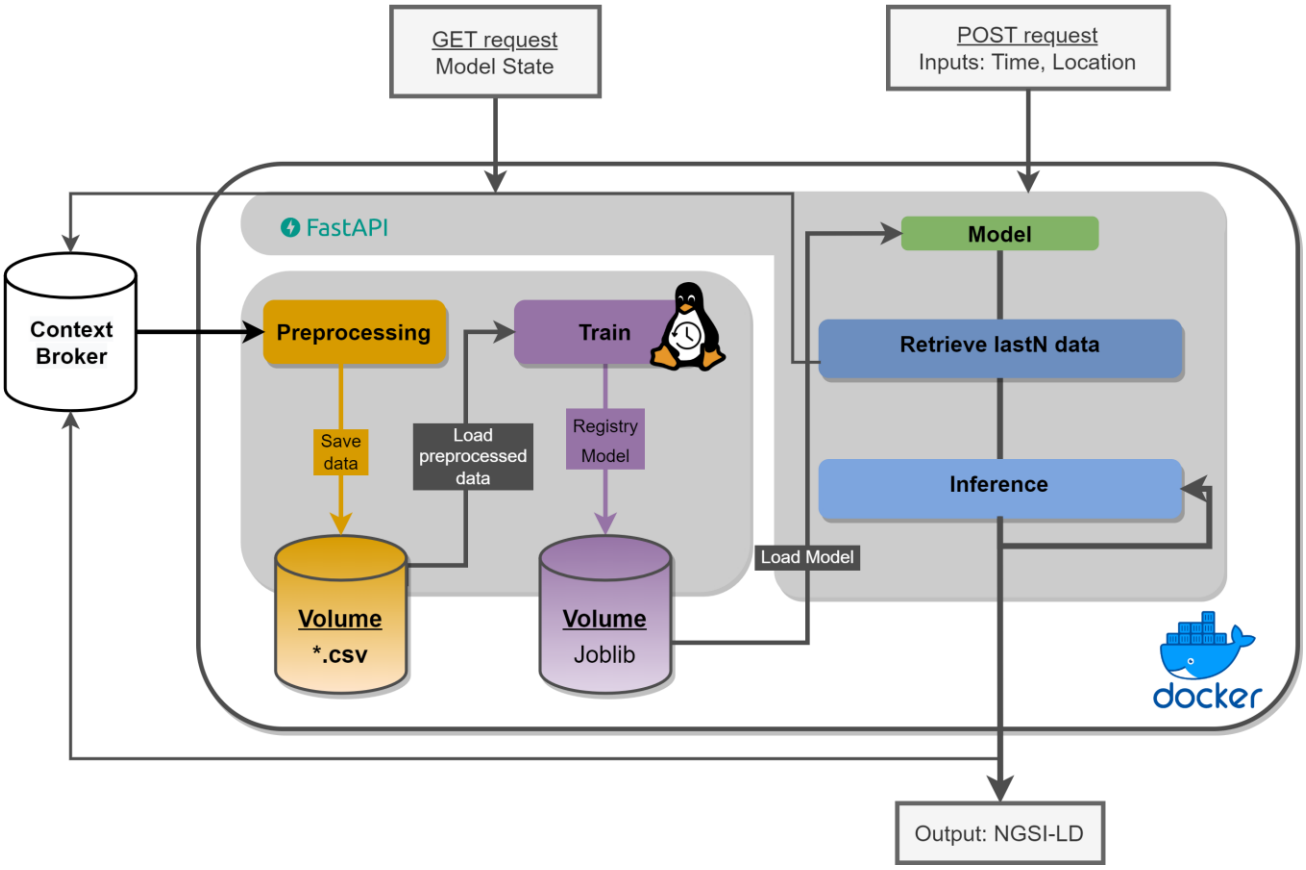

Figure 16: Architecture of the Traffic forecasting service

3.1.5 Conclusions

Traffic intensity follows a periodic pattern that is predictable. The process to provide a forecast consist in data cleaning first, to clean historical data. Then, several models and configurations were benchmarked in order to identify what forecast solutions fits best the shapes of traffic intensity evolution. It was identified that kNN is an accurate technology that does not require too many training data. This makes it a convenient technology to be used in an API configuration. FastAPI allows an easy and fully configurable solution for requests URLs, along with an automatically generated swagger documentation that helps users' guidance in their requests.

The proposed service was dockerized in order to enable quick replication. Each sub service (fastAPI, python functions, ...) has its own docker-image that is managed by a single docker compose file.

3.2 Traffic Environmental Impact Calculation

3.2.1 Description

This service calculates the emissions the cars predicted by the Traffic Forecasting service would produce, based on European studies [3]. Results are in g/h (grams produced by hour the cars are running).

3.2.2 Model Development

3.2.2.1 Preprocessing

When a request is made, this service checks that the TrafficEnvironmentImpactForecast entity is correct and has all the required fields.

3.2.2.2 Rules

The service calculates the emissions with the following formula [4, 3].

$$E_{CO2} = 44011 \cdot \frac{FC \cdot velocity \cdot intensity}{12011 + 1008 \cdot 1.96}$$

Where FC is the fuel consumption corresponding to each vehicle type, and the intensity is the number of cars of the said type.

3.2.3 Model Deployment

3.2.3.1 Architecture

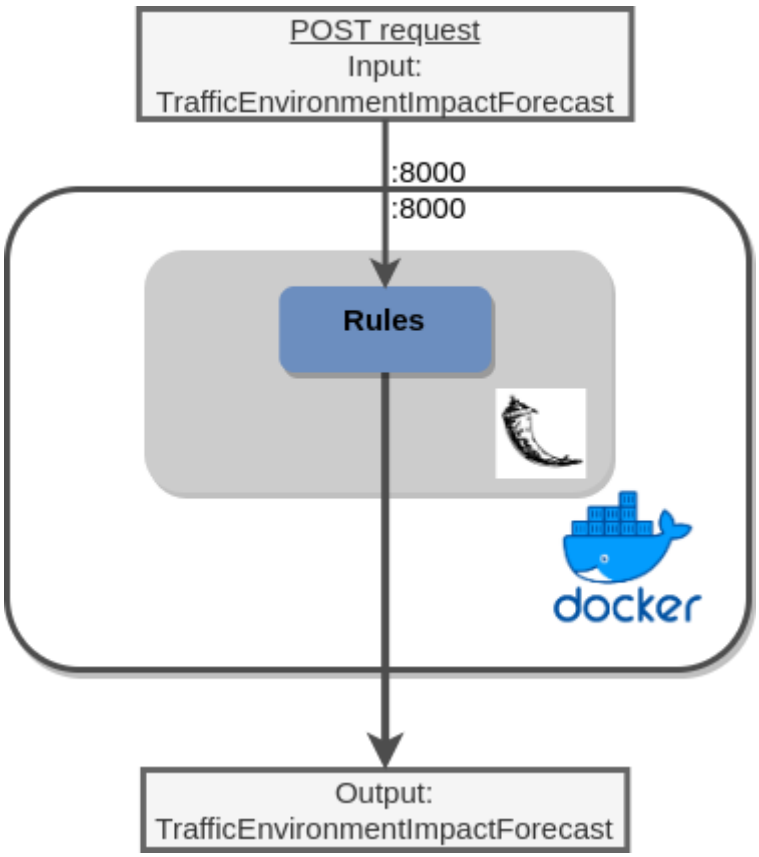

Figure 17: Traffic Environmental Impact Calculation service architecture

We use Flask to define a Rest API. This is encapsulated in a docker container.

3.2.3.2 Workflow

1. The service receives a request, that should be in NGSI-LD format.
2. The service checks the body of the request to see if it contains a correct TrafficEnvironmentImpactForecast entity.
3. The service applies the calculation rules and fills the corresponding fields of the TrafficEnvironmentImpactForecast entity with the results.
4. The service returns the completed TrafficEnvironmentImpactForecast entity in JSON-LD format.

3.2.4 Conclusions

The service developed calculates the emissions from an TrafficEnvironmentImpactForecast entity following European formulas. The service can be easily deployed thanks to Docker.

3.3 Traffic Recommendation

3.3.1 Description

Once the traffic intensity for a given time in the future has been predicted, and that its environmental impact was determined, it is then possible to provide relevant recommendations for end-users or for the city traffic management team in order to reduce the traffic intensity by proposing acceptable alternatives, such as public transportation, the use of shared bikes, or to promote transport reduction solutions such as working from home.

Therefore, this service uses a forecast of the traffic intensity, the noise annoyance and the air quality to determine if there is a need for noise or pollution reduction. If such a need is identified, then the recommendation service will look for possible recommendations. These recommendations include the proposition to use alternative transport solutions, shared bikes, or transport reduction. To provide these recommendations, the service uses the output from a bike availability forecasting service, along with the schedule of public transportation services.

Therefore, the service provides a recommendation to the user or the city transportation department, suggesting a reduction in the traffic intensity along with a proposition for the most efficient and sustainable mode of transportation for their needs. For example, if traffic is expected to be heavy and air quality is poor, the service may recommend using a bike or taking public transport instead of driving, depending on the weather forecast. On the other hand, if traffic is light and air quality is good, no specific recommendation will be provided apart from business as usual.

By considering all of these factors, the service is able to provide users with a comprehensive recommendation that takes into account not only the current traffic conditions, but also the impact of their transportation choices on the environment and their overall well-being.

3.3.2 Model Development

3.3.2.1 Pre-processing

This service's model will ingest data such as traffic predicted, Noise annoyance forecasting, Air Quality Index predicted, Bikes availability and public transportations in the area, these inputs need to be accurate, relevant, and ready for use which involves cleansing : removing any irrelevant, duplicate, or inconsistent data and Pre-processing : transforming the data into a usable format, such as normalizing, imputing missing values, and encoding categorical variables. By performing these steps, the service can help to ensure that it will not break (due to a missing time or to an outlier), and to ensure it will provide reliable recommendations to users.

3.3.2.2 Rules

The service that provides traffic recommendations for users in a given area can either work "on-demand"or be triggered on a daily basis to provide recommendations every day. It considers several key factors in order to make a recommendation. First, after pre-processing of the data, it compares the traffic intensity forecasts with a threshold level provided by an expert of the considered location (such a threshold can be an average of traffic intensity at similar times in the past). This allows to determine an indicator of the overall level of traffic in the area. In the meantime, the service also requests the air quality and noise annoyance at the considered time in order to compare these with a given threshold and determine if there is a need to act for a better environment. Secondly, the service computes the correlation between the traffic intensity and the air quality and noise annoyance. To do this, the service requests the last 7 days of air quality, traffic intensity and noise annoyance prior to the requested time of the recommendation. This allows to compute the correlation between the traffic

intensity and noise annoyance at the requested period of time on one side, and the correlation between the traffic intensity and air quality at the same period on the other side. Then, if the correlation between traffic and air quality has been established, the traffic recommendations service requests for alternative transportation solutions such as bikes availability or public transportation schedule to find out the best transportation alternative to the individual cars. This step also requires the knowledge of weather forecasts to assess the suitability to use bikes.

Therefore, the traffic recommendations can be multi fold: it can recommend to not change anything, or to reduce the use of individual cars by using shared bikes, or to use public transportation available locally at the considered period of time.

These recommendations are then sent to the subscribers or to the users who made the original request of a traffic recommendation.

3.3.3 Service Deployment

3.3.3.1 Architecture

The proposed service follows the architecture described in Figure 18.

A docker container includes all the necessary images to work. A FastAPI image is used to receive requests for subscription to daily recommendations or for real-time requests for traffic recommendations. A database stores the subscriptions, whereas a cron job is created to ensure periodical execution of the traffic recommendations. The recommendations require inputs from other services such as noise annoyance forecasting, air quality forecasting, traffic intensity forecasting and bikes availability forecasting. It also requires inputs such as public transportation mapping and schedules, but also weather forecast for the time of the recommendations. Recommendations are then provided depending on the impact of traffic on air quality reduction or on noise annoyance.

The recommendations are finally sent back to the user, either as a reply from the post request or as an email or a pushed notification.

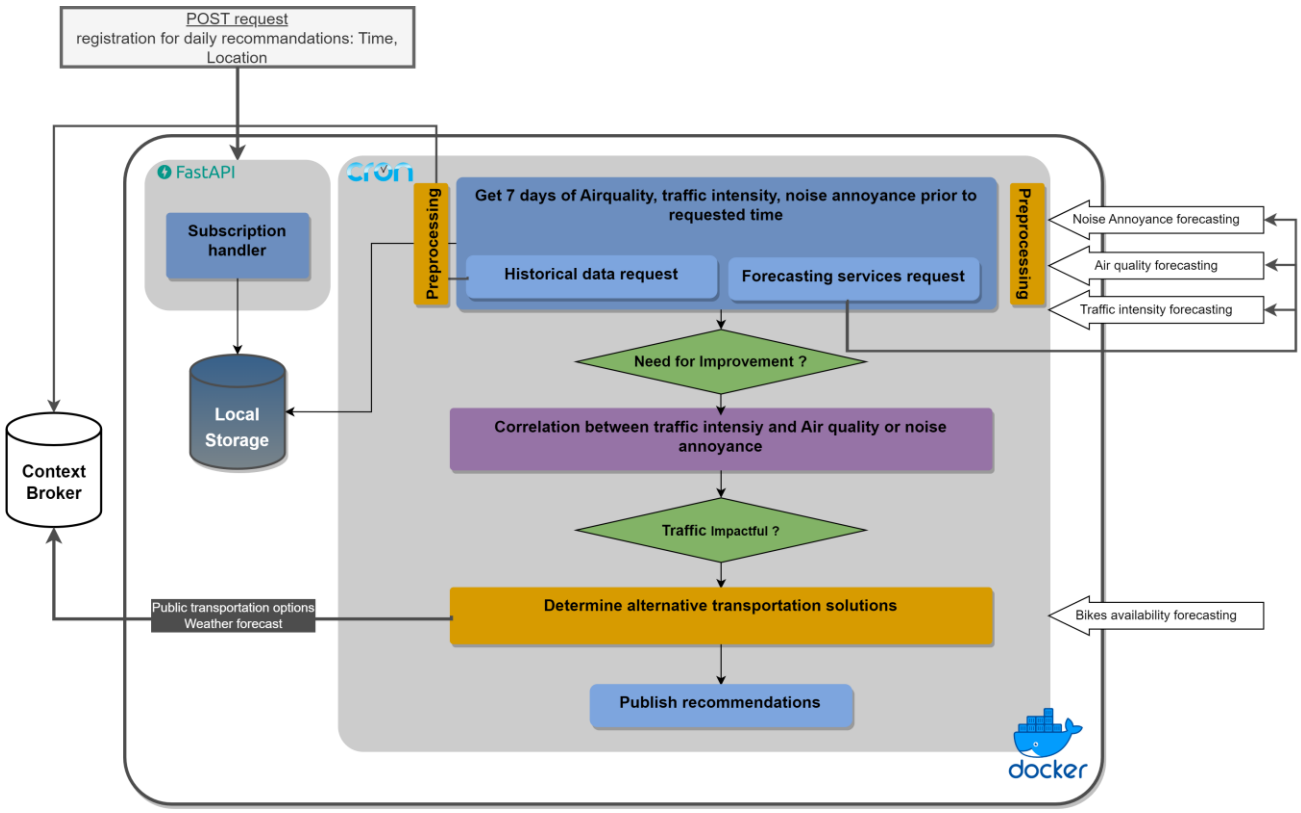

Figure 18: Traffic recommendations service architecture

3.3.3.2 Workflow

Following the architecture described in Figure 18, the first component that is started when a traffic recommendation is to be generated consists in the retrieval of noise annoyance, air quality and traffic intensity forecasts for the time of the requested recommendation, but as well for the 7 days prior to this time, as it will allow the service to determine if the traffic intensity is linked with bad air quality for example.

The forecast and historical data is then processed to remove any outlier or timing issues. Once the service received the 3 requested time series (air quality, noise annoyance and traffic intensity), it first checks if there is a need for improvement of air quality or noise impact at the requested time. If yes, it will proceed further, otherwise, a recommendation to not change anything is sent out. If there is a need to improve air quality or noise annoyance, the service will start to identify if there is an impact of the traffic on air quality and noise annoyance. This is done through a correlation between air quality and traffic, and between noise annoyance and traffic. To achieve this correlation, time-series of 7 days were considered as necessary. If the traffic is assessed as impactful, i.e. there is a strong correlation between traffic intensity and the air quality or noise annoyance, then the traffic recommendations service will look for recommendations that will aim to decrease the traffic intensity at the considered time. This is done through an assessment of potential options for alternative transportation, such as shared bikes or public transport (tramways, bus, subway). A weather forecast is also used to assess the suitability of proposing the use of bikes. Given these inputs, the traffic recommendation service will either propose to use the bikes if the weather is nice and there is no public transport available at the considered time, or to use public transport if the weather is not good or if there is no bike available, or to leave the choice to the user to use one or another. If the traffic is not correlated with air quality or noise, then, recommendations sent

to the user are to not change anything on the traffic, but to investigate further the sources of noise. An example of correlation analysis is shown in Figure 19 for noise and traffic intensity correlation, where cross correlation is used to capture the timing lags between the two variables evolutions.

Finally, the recommendations are then sent to the user, either as a reply to a real-time request for traffic recommendations, or by pushing or sending email on a daily basis to the subscribed users .

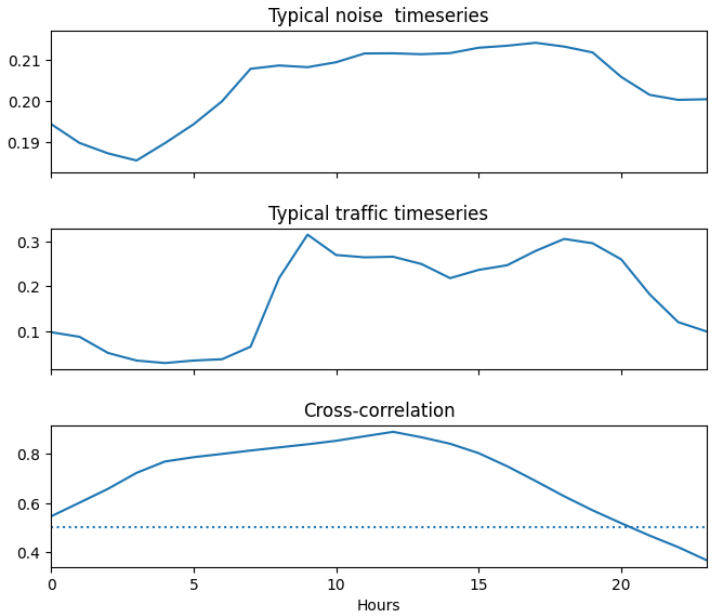

Figure 19: Example of Cross-correlation studies between traffic and noise

3.3.4 Conclusions

The proposed traffic recommendations service coordinates other services in order to assess the need for traffic reduction at given locations. It requires traffic forecasting, noise forecasting, air quality forecasting but also bikes availability forecasting and information about public transport and weather. Therefore, it is a comprehensive service that provides users with recommendations on a daily basis.

4 Bikes Availability Service

A model for predicting bike availability aims to estimate the number of available bikes at a given location and time based on past data. This model can be useful for bike sharing systems to optimize fleet management and improve the user experience. The model may consider factors such as holidays, day of the week, and historical bike usage patterns to make accurate predictions.

The first part of this section outlines the key components of our bike availability forecast model. We will discuss the algorithms used to collect and process data, and how the model calculates the availability of bikes. This section will also provide details on the model's limitations and potential for improvement.

The second section of this document outlines the deployment of our bike availability forecast model. We will discuss how it is maintained and updated. Additionally, we will explore several Machine Learning tools involved in this service.

Finally, the document concludes with a discussion of future directions and opportunities for improvement. We will outline potential areas for research and development, as well as ways the model can be enhanced to better meet the needs of bike-sharing systems and their stakeholders.

4.1 Bike's availability forecasting

4.1.1 Description

In this section, we will provide an overview of a bike availability prediction model. The model will be discussed in greater detail by referencing two real-world cases of bike sharing systems: VeloBleu in Nice, France and Bluebike in Flanders, Belgium. This model predicts the number of bikes available for rent at a given location and time. The bicycle availability forecasting service boasts a remarkable level of accuracy, down to the minute, and operates in real-time.

Throughout this discussion, we will primarily be referring to the case of Blue-bike in Flanders, Belgium. However, the same principles and mathematical techniques used to model bike availability for Blue-bike can also be applied to other bike sharing systems, including VeloBleu in Nice, France.

4.1.2 Data analysis

The data received from each Bicycle Sharing System (BSS) is composed of as many time series as there are stations. In this section, we will describe the most important aspects about them.

4.1.2.1 Historical data

Availability data points are recorded asynchronously, meaning that they may not be recorded at regular intervals. Nevertheless, we have a complete historical record starting from the year 2022, providing information on the availability of bicycles in each station over time.

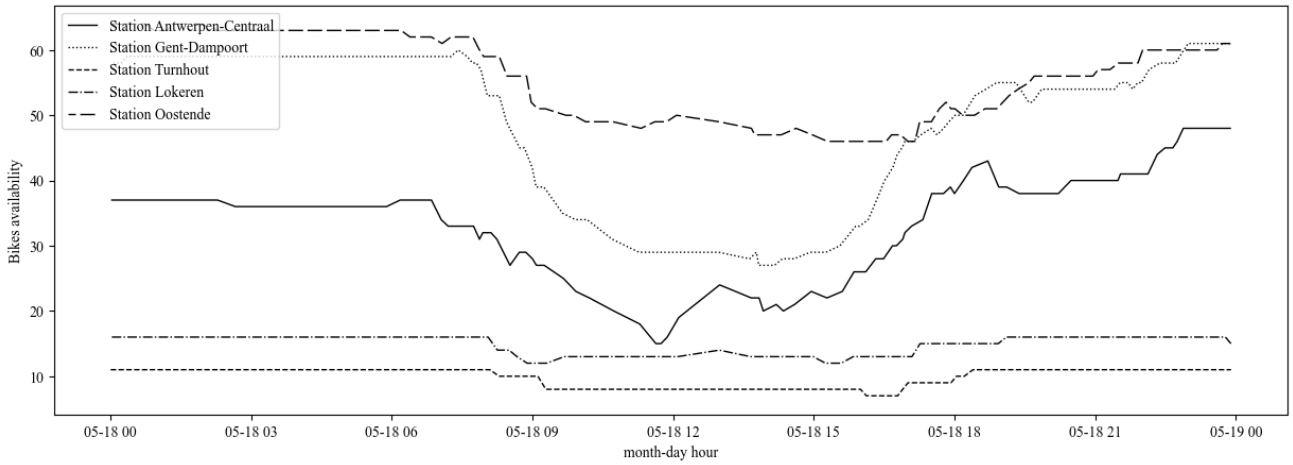

Figure 20: Bike availability for 5 stations in Flanders.

Instead of analyzing the availability data, we can transform the information to observe the number of bicycles that arrive at each station on an hourly basis. This transformation allows us to observe differences between weekdays and weekends. Figure 21 and Figure 22 illustrates these differences. Hence, regressors such as the day of the week, hour, and whether it is a workday will be useful in our modeling process.

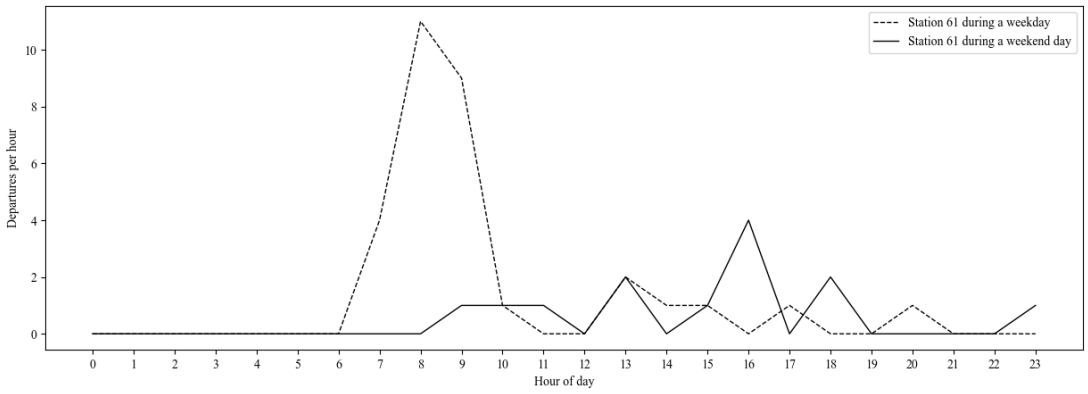

Figure 21: Departure rates of bikes: weekday vs weekend day

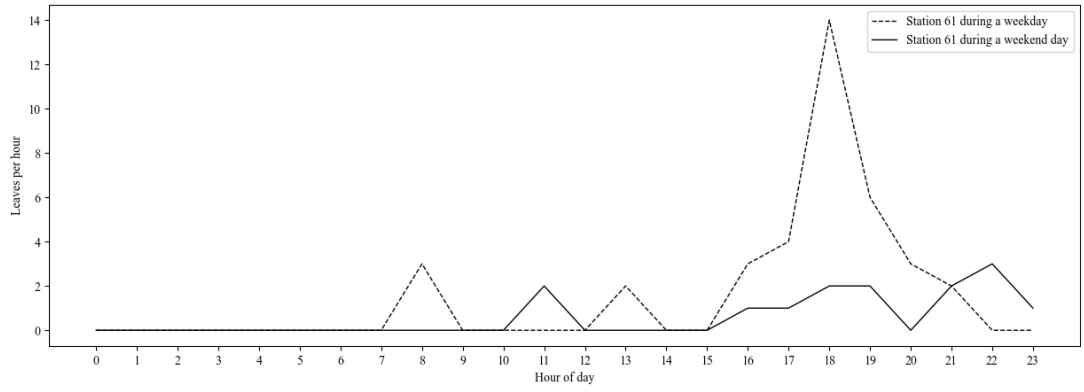

Figure 22: Arrival rates of bikes: weekday vs weekend day

On weekdays, there appears to be a correlation between the number of bicycles that depart in the morning and the number of bicycles that return in the afternoon. This can be further confirmed by analyzing the correlation matrix between these two variables (see Figure 23).

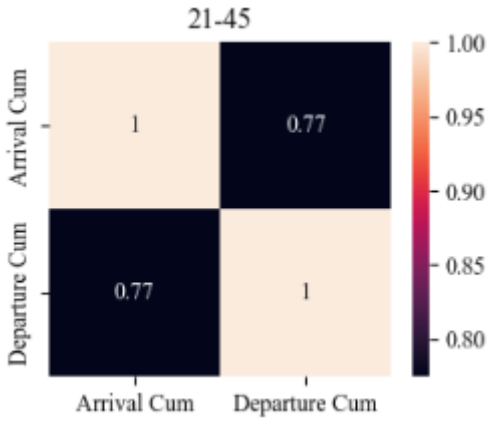

Figure 23: Correlation between Arrival and Departure Cumulative data.

4.1.2.2 Pre-processing

The aim of the data preprocessing is to create a transformer that inputs an asynchronous time series of bicycle availability and outputs a synchronized time series, sampled at 15-minute intervals. The output will include two columns, detailing the number of bicycles that arrived and the number of bicycles that departed in the last 15 minutes.

4.1.3 Model Development

At this point, it should be clear to the reader that the prediction model will focus on the arrival and departure ratios at a bicycle station, rather than the availability itself. In this section, we will provide a comprehensive overview of the model's development.

4.1.4 Model description

Our model will consist of two main stages:

1. Predicting arrival and departure ratios for a bike station.
2. Calculating bike availability based on the arrival and departure ratios.

Let's start with the simpler stage: calculating bike availability. By using the current bike availability data and the arrival and departure ratios, calculated in 15-minute intervals, we can calculate future bike availability in the same interval through a simple formula.

$$Availability\ at\ (t + \Delta t) = Availability\ at\ t + Arrivals\ in\ \Delta t - Departures\ in\ \Delta t$$

where Δt is the time interval (e.g. 15 minutes) and Availability at t represents the number of bicycles available at time t. Thus, we can provide an estimate of the availability at any given time.

4.1.4.1 Training

It is widely acknowledged in the field of statistics that events related to the arrival of individuals, such as in a queue or packages of information to a server, or in our case, bicycles to a station, tend to follow a Poisson

distribution. In this phase of model training, we will explore the applicability of this distribution to the arrival of bicycles at a station.

We will begin by validating that the arrival and departure ratios of the stations follow a Poisson distribution. In Figure 24, we compare distributions for various time intervals for several stations.

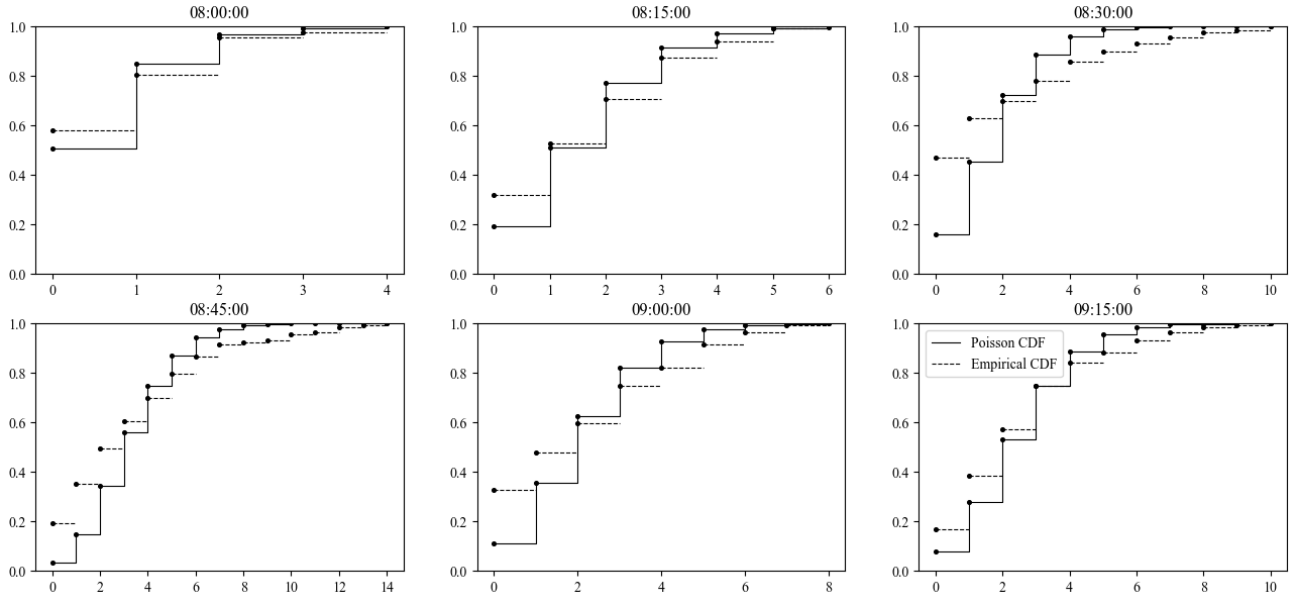

Figure 24: Empirical CDF vs Poisson CDF for 6 time intervals at Station Gent-Dampoort.

The Kolmogorov-Smirnov test is a widely used statistical test for assessing whether a sample of data follows a specific distribution. It compares the sample distribution to the hypothesized distribution by computing the maximum difference between their cumulative distribution functions. The test outputs a p-value that provides information on the strength of evidence against the null hypothesis (that the sample and hypothesized distributions are the same).

In our case, we will use the Kolmogorov-Smirnov test to validate our assumption that the arrival and departure rates of the bike stations follow a Poisson distribution. By conducting the test on each station and interval of the day, we found that over 91 % of the stations can be modeled with a confidence level of greater than 95%. The following table summarizes the results obtained.

This supports our hypothesis that our arrival and departure ratios follow a Poisson distribution.

Based on all the information presented in this subsection, we have decided, among other models, to use the XGBoost Regressor for Poisson models.

- Regarding the predictors that we have added, they include:
- Features related to the characteristics of the moment: minute of the day, day of the week, etc.

Lags of the ratios that we aim to predict: As is commonly done, we use the lag of our predictor of 3 to 4 previous samples for the arrival and departure ratio models. We also include lags of one day and one week in order to consider seasonality. As we highlighted in the presentation of the historical data, there is a relationship between the arrivals in the afternoon and the departures in the morning. Thus, we will use a lag of several hours that relates to these two moments.

It is worth mentioning that two types of training are carried out:

- Training models for a data set for each station, on a daily frequency.
- Training and hyperparameter tuning for a single station, on a daily frequency.

4.1.4.2 Results

In a time series context, the traditional train-test split method is different from the standard approach because the data is divided chronologically. The objective is to ensure that the predictions are not biased by having information from the future in the training set. This means that the training set consists of data that precedes the data in the test set, so the model can only use information from the past to make predictions.

The chronological split allows for a more realistic evaluation of the model's performance, as it mimics the real-world scenario where only historical data is available for making predictions.

We will be using the R-squared metric to evaluate the performance of our model. R-squared, also known as the coefficient of determination, is a statistical measure that indicates the proportion of the variation in the dependent variable that is explained by the independent variables.

For the case of Flanders, we have trained and calculated this statistic. Obtaining, for all stations, an average R2 score of 0.97. The following figure shows, using points, the score obtained for each model.

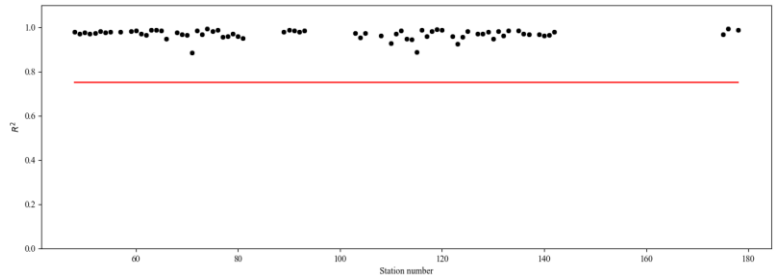

Figure 25: R2 for Flanders use case.

For the case of Nice, we have obtained the following results.

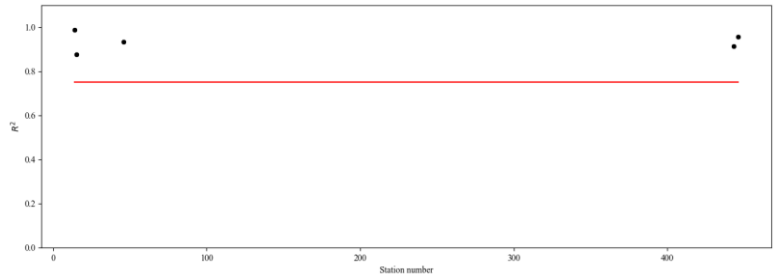

Figure 26: R2 for Nice use case.

4.1.5 Model Deployment

The guidelines followed in this section correspond to those explained in detail in 2.1.4 regarding Air Quality Forecasting model deployment. In this sense, three different stages are identified: Data Preparation, Model Creation and Model Deployment.

In the following sections, this stage will be explained in detail, emphasizing differences with AQF service.

4.1.5.1 Workflow

In Figure 27, schema of the workflow is shown. As it is stated above, three stages are defined: Pre-process, Model Creation and Model Deployment. These stages ensure that the model is properly developed, trained, and evaluated, and that the results are meaningful and can be applied to real-world problem.

Differences between 2.1.4.1 and this case are pointed out within the rest of the section.

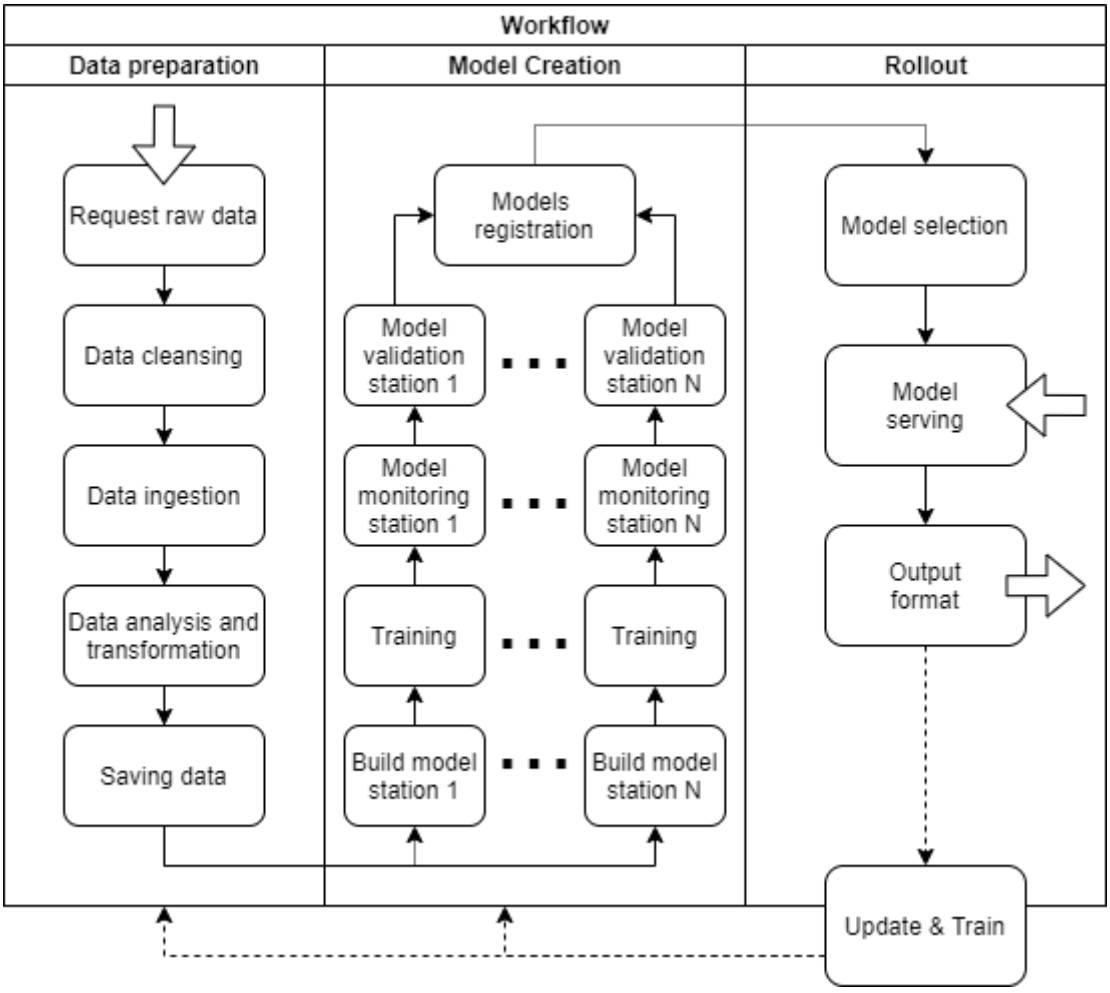

Figure 27: Workflow AQF service

Data preparation

In this stage, the sequential execution of request for new data, data cleansing, data ingestion, data analysis and transformation, and data saving takes place. The analysis and pre-processing applied in each step are defined in 4.1.2.2, and the result of this phase is a new version of the table with updated and pre-processed information.

Model Creation

In this phase, there is a key difference. As a result of Model Development, one ARIMA model will be trained for each station. To make this independent of the use case, the number of stations (i.e. trains) is extracted from the historical dataset. Before training the models, a filtering is applied to the dataset, and all steps are executed in parallel. Finally, all models are registered with different names.

Rollout

Again, the criterion for model selection is the last trained model. However, there is another layer of selection in this case, as the selected model must take location information into consideration, as one model is trained for each station. This layer will filter the model at the Model Serving step. In the end, the correct format is applied to ensure NGSI-LD [7] compliance.

As defined in 2.1.4, an update process is introduced to prevent unjustified computing overloads. In this sense, a daily train at late-nights is triggered, as well as four hourly data updates for each model. The Inference service and Train service are independent, so user requests will be answered while update or training processes are being executed.

A complex gridsearch routine has been implemented to find the best parameters for each model. However, due to its high computational cost, a rotative selector is implemented to execute gridsearch for a small part of the models each day. All models execute this in a sequential manner, always following the same order, to ensure that the parameters of all models are recalculated before the sequence starts again. If a model is not selected, it will retain its last parameters. Although the accuracy of a model is not compromised when the gridsearch routine is not applied, the distribution of input data can vary over time. This system increases the robustness of the models.

4.1.5.2 Architecture

The selected technologies correspond to those viewed in 2.1.4, since all the requirements posed by the use case are met. Those technologies are:

- **MLFlow [9]**. With this tool, we have the ability to carry out Model Tracking and Model Monitoring. It provides a user-friendly interface for monitoring all elements involved in the model creation process, including training parameters, training metrics, and model registration.
- **KServe [12].** It is the chosen platform for Model Inference. It offers a comprehensive set of tools for the Model Serving stage, including interconnected APIs for data preprocessing, performing inferences, and postprocessing.
- **Crontab [13].** It is a crucial component as it enables scheduled execution of tasks such as Model Creation and Data Preparation with a specified frequency.

The technologies and processes that take part in this service are encapsulated in a Docker image, as it was defined as the outcome of task 3.2. In Figure 28, this architecture is shown.

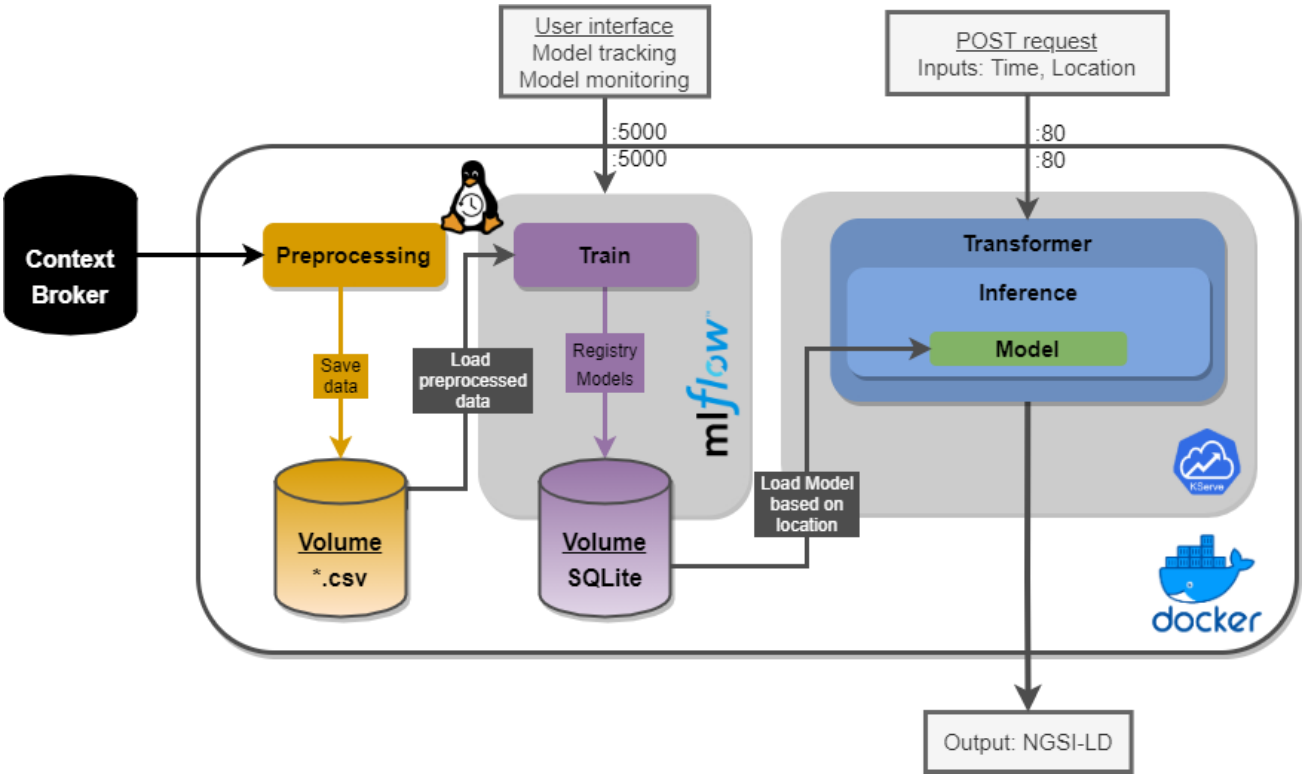

Figure 28: Architecture Bikes availability forecasting service

Preprocess phase involves acquiring new data from the appropriate context broker, followed by cleaning and preprocessing, before being combined with the existing historical data. The processed data is then saved, creating an updated version of the table, which is loaded for the next stage: training. The training step is carried out using MLFlow and involves executing the steps outlined in the model creation phase, leading to the creation of a new set of models that are registered in a SQLite database. The system is then ready to respond to incoming requests through the inference platform.

The connection between the training step and the inference step is maintained through the database, where all models are registered. Upon receiving a request, the latest trained model is loaded from the SQLite database and used in the inference platform, making the two services independent of each other. During the training step, users can still make requests, but it is scheduled to occur daily during off-peak hours.

To minimize computational costs and avoid multiple training sessions per day, the system updates the data in the latest trained model every hour, updating the model and registering it in the same manner as a new model. Two routines are set up through Crontab, including a daily training routine that executes the Preprocess and Train phases and an hourly update routine that executes the Preprocess and certain steps of the Model Creation phase. MLFlow, KServe, and Crontab are launched at startup.

For data and model persistence, Docker volumes are utilized. The historical data is stored in a .csv file in the Docker volume, ensuring its preservation in case of service interruption. The SQLite database, also configured in the volume path, stores all the information needed to register the latest trained models upon startup.

4.1.6 Conclusions

The process of developing and deploying a machine learning model for bike availability forecasting has been outlined. Data preprocessing techniques are selected to meet model stations as Markovian Queues. In the case of multiple stations, multiple models are considered in the service.

In addition, communication with the stakeholders of the use cases has played a significant role in determining expectations and identifying appropriate technologies and processes to aid in this endeavor. Technologies such as MLFlow and KServe can streamline machine learning operations, and these cutting-edge technologies have been implemented for this project.

To maintain consistency and ease of deployment, all components have been packaged into a single Docker image. This approach ensures efficient deployment of the bike availability forecasting service. However, in the future, a microservice architecture may be considered to enhance flexibility and scalability.

It is worth noting that this machine learning model for bike availability forecasting is constantly evolving, and the inclusion of meteorological regressors is a potential area for future improvement. The current design of the system is such that the addition of these regressors is relatively straightforward and will enhance the model's accuracy and effectiveness.

5 Noise annoyance forecasting services

5.1 Noise forecasting

5.1.1 Description

Noise forecasting is a service that predicts the level of noise disturbance in a specific location within a specified time frame. The aim of this service is to provide information about future noise intensity so it can be used to compute its expected impact. This service output is sent to the noise annoyance calculation service for prediction of future noise annoyance, which will then be sent to the traffic recommendation generation service, as depicted in Figure 29.

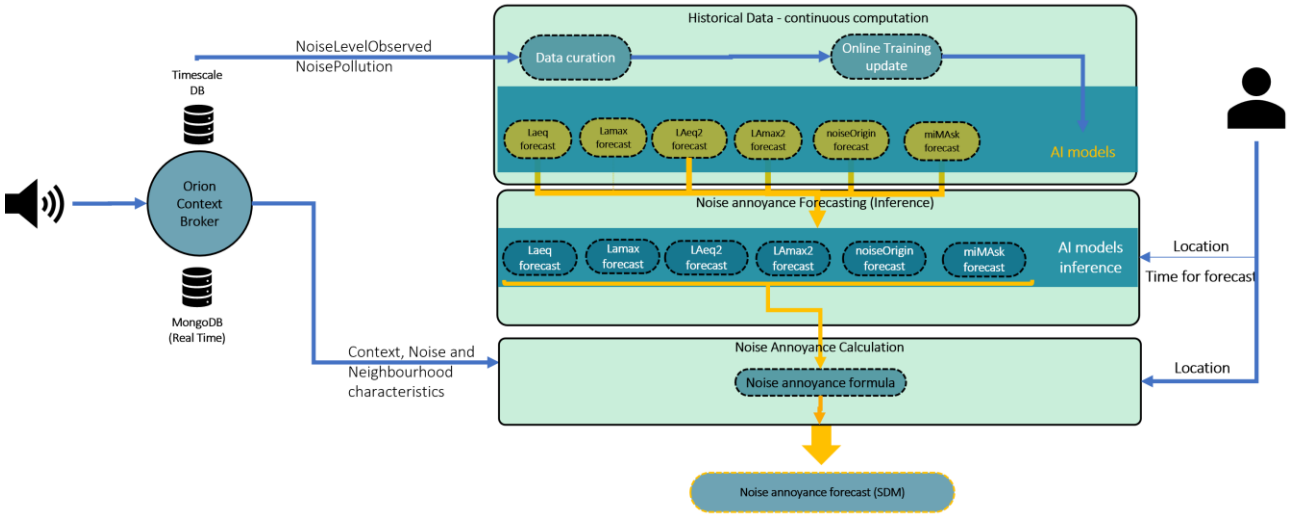

Figure 29: Overview of the whole noise annoyance forecasting services

5.1.2 Data analysis

5.1.2.1 Historical data

The historical data sets involved in the training of the AI model and the implementation of the noise annoyance forecasting service play a crucial role in its accuracy. In the case of Nice use case, the data set includes the noise levels in a specific location (Sensor with the following coordinates: 43°40'56.4"N 7°13'58.1"E) from 01/01/2020 until 2022. This data provides a comprehensive view of the noise situation in the area over a two-year period and can be used to train the AI model to make accurate predictions about future noise levels.

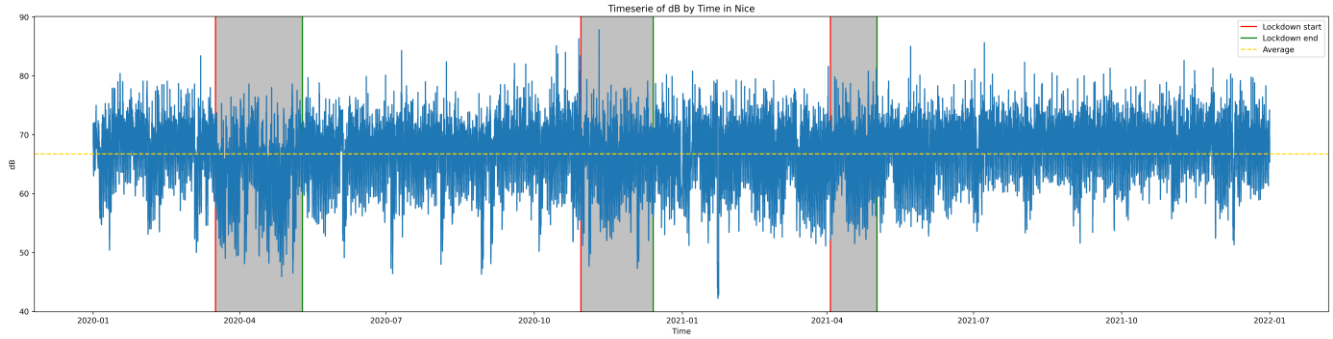

Figure 30 : Initial Noise dataset used in the use case of Nice.

Figure 30 shows the recorded data for the use case of Nice. Covid lockdown periods were highlighted in order to show that not all the available data can be used in the study. It shows that the training dataset of the forecasting models might require to be adjusted and benchmarked in order to avoid negative impacts from period of time that will never happen again.

5.1.2.2 Pre-processing

Before the data can be used in the calculation of noise annoyance, it is important to perform a thorough cleaning and pre-processing process to ensure accuracy and reliability of results. This process involves reviewing the data for any errors, discrepancies, or missing information, and correcting or removing these issues as necessary. Furthermore, the data may need to be transformed or combined to make sure it is in the appropriate format for the prediction.

The pre-processing step is crucial to the success of the noise annoyance forecasting service, as it guarantees that the results are trustworthy and dependable. By carefully examining and adjusting the data, the performance of the service is also enhanced and the possibility of errors or inaccuracies in the results is minimized.

As shown in the Figure below, we removed all the rows with NaN (Not a Number) values inside any column(s) (12 Rows = 0.01% of the dataset) and we got rid of aberrant data like lonely spikes using quantile (1411 rows = 2.01% of the dataset).

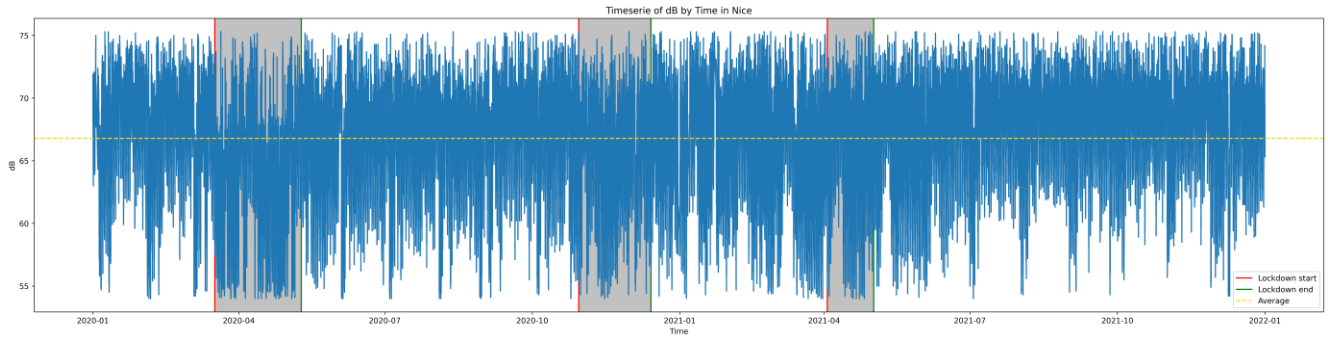

Figure 31: Pre-processed Noise dataset for the use case of Nice.

5.1.3 Model Development

5.1.3.1 Model description

The development of the noise prediction service is a crucial aspect of ensuring its accuracy and effectiveness. The AI model must be able to effectively analyse the historical traffic data sets, considering the various factors that impact the noise intensity, and make predictions about future noise levels. To ensure this, the service must perform several key steps, including data preparation, model selection, and model training, the service must choose an appropriate AI model that is capable of handling the complexity of the data and the variety of factors that impact the noise. This may involve evaluating several different AI models and selecting the one that provides the most accurate predictions. The models that were benchmarked so far in the case of traffic forecasting include K-Nearest Neighbours (KNN), Decision Tree and Random Forest. By comparing the performance of these models using different historical data sets for training, it was determined that KNN outperforms the others with an accuracy above 85%.

In terms of features, the model takes the following inputs to provide a forecast:

- type of day (week or weekend).
- Hour of the day.
- Noise at the following lags (hours before):
    - 1h, 2h, 3h, 24h, 72h and 168h

The outputs of the model consist in all the noise intensity values, from the time at which the request was sent to the noise level at the requested time and location. Also, it is worth noting that each model corresponds to a unique location. Therefore, in the case of GreenMov, there is one model for each of the locations of the use cases that require this service.

5.1.3.2 Training

In the case of Nice, the results of the benchmarking are seen in the figure below for different sizes of the training dataset. Indeed, we can see that the size of the dataset used for training also has an impact on the forecast accuracy. In the case of Nice, a training dataset of 2 weeks provide good enough accuracy based on this service's KPI.

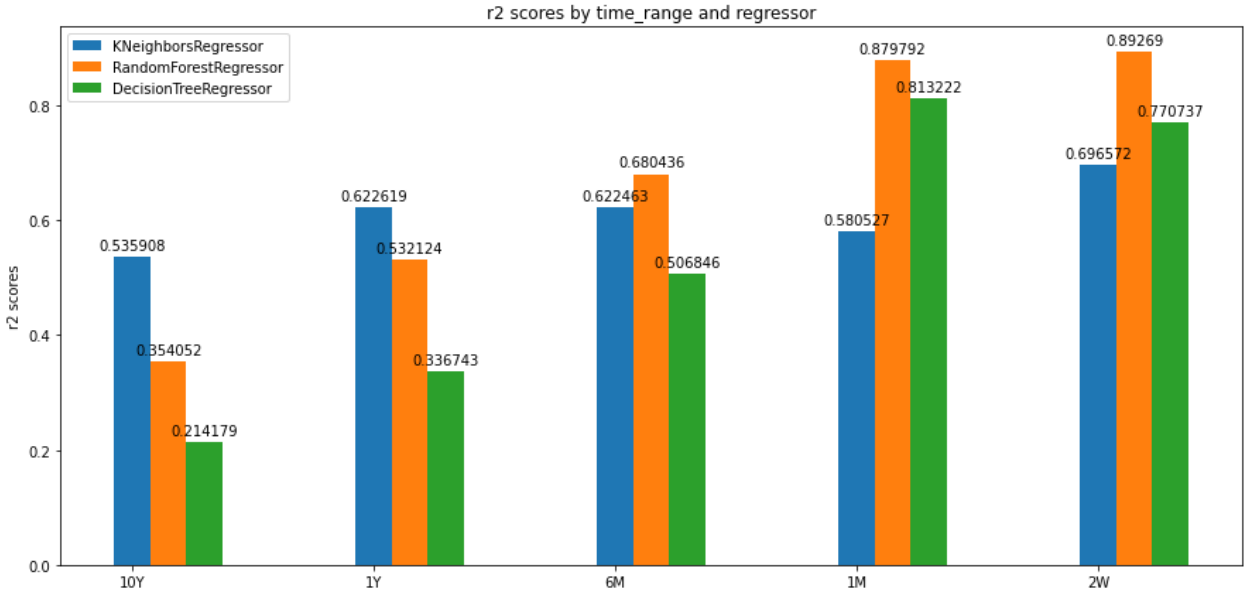

Figure 32: Results of AI models benchmark for the use case of Nice.

5.1.3.3 Results

Figure 32 displays the r^2 scores for the different AI models used, and for different sizes of training dataset, with a maximum r^2 accuracy of 89%. These figures were obtained using a train/test split of 70%.

Figures below show graphically the comparison between real measurements and predictions. We can see in Figure 34 that there is a significant gap between forecast and measurements, but these are due to a measurement error, given the fact that the recorded values were considerably low compared to all previous historic data.

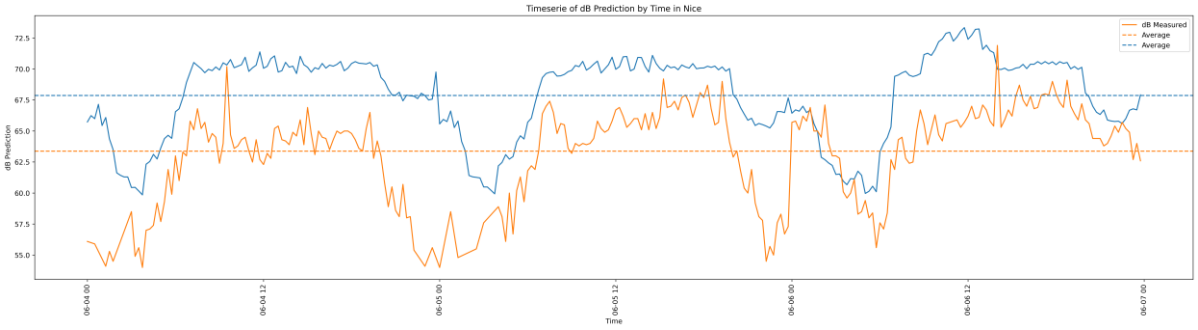

Figure 33: Measured vs Predicted noise over 3 days in the use case of Nice.

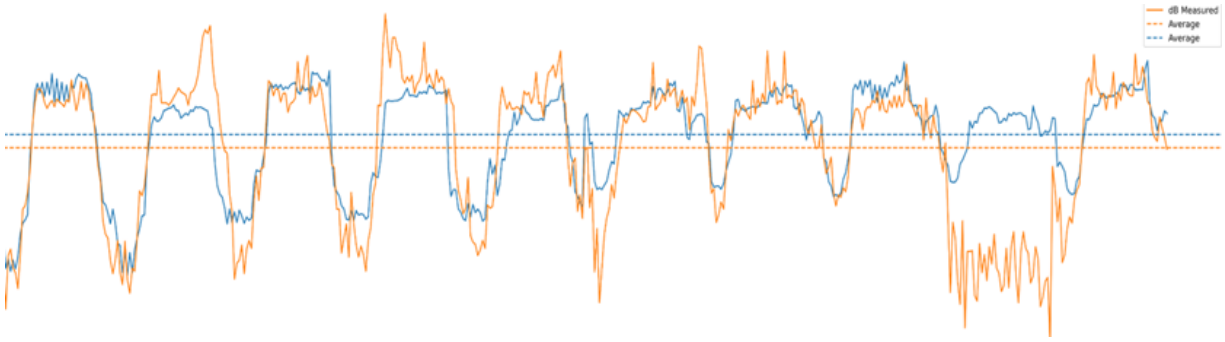

Figure 34: Forecast (blue) vs Ground Truth (orange) for one week in 2022

5.1.4 Model Deployment

Once the model was designed, the use cases have been able to implement it in their infrastructure. This subsection presents the process followed for service deployment.

5.1.4.1 Workflow

The workflow that was followed to achieve the final model is presented below, in Figure 35. First, data was gathered in order to be able to analyse the data and determine the traffic forecasting model. Therefore, data was gathered from the use cases. It was then cleaned (as explained in the pre-processing section), augmented with complementary data or information (such as last hours data, type of day, ...). These steps are gathered in the data preparation section.

Then, in the model creation phase, several models, features and training requirements were assessed in order to find the optimal combination. One model is designed for each location. Models that were tested include K-Nearest Neighbours (KNN), Decision Tree, and Random Forest, but will also include XGBoost. Then, the features (inputs) that were selected consist in the type of day, and previous noise level (of the last hour, the 2 last hour, until 168 hours ago). The training can use years of data, or can be restricted to a few weeks or months, in order to not capture specificities from summer when requested to provide a forecast for winter. Therefore, a benchmark of training dataset's size was done to determine the best size of training dataset to use in the selected forecasting solution, as shown in Figure 32. These different benchmarks led us to a specific combination of model, training size and features that will be used in the noise forecasting service.

Finally, in the roll out phase, we leverage the architecture, the inputs and the training requirements to provide a real time solution that determines the predicted noise level at the required time. The working steps of a forecast computation are as follows:

- First, from the requested location of the forecast, the model that corresponds to the location is selected.
- Then, the last N data are requested to the context broker in order to retrieve the noise intensity at the previous times (last hour, last 2 hours, last 24 hours,...).
- The noise level of the next hour is predicted.
- Using this forecast as an input of the next model's inference, the model is used to compute the noise intensity forecast in the next 2 hours.
- This new forecast is used as an input to forecast the noise level in 3 hours.
- And this process is repeated until the time of the requested forecast is reached.

The result of the final forecast (for the requested time) is then formatted as NGSI-LD [7] and fed back to the user.

Finally, at the end of each day, the daily data is retrieved from the context broker and the model is trained again using the new training dataset that consists in the previous training dataset minus one day, and plus the last data from the last day. The resulting updated parameters of the model are then stored using joblib to be used for next day's forecasts.

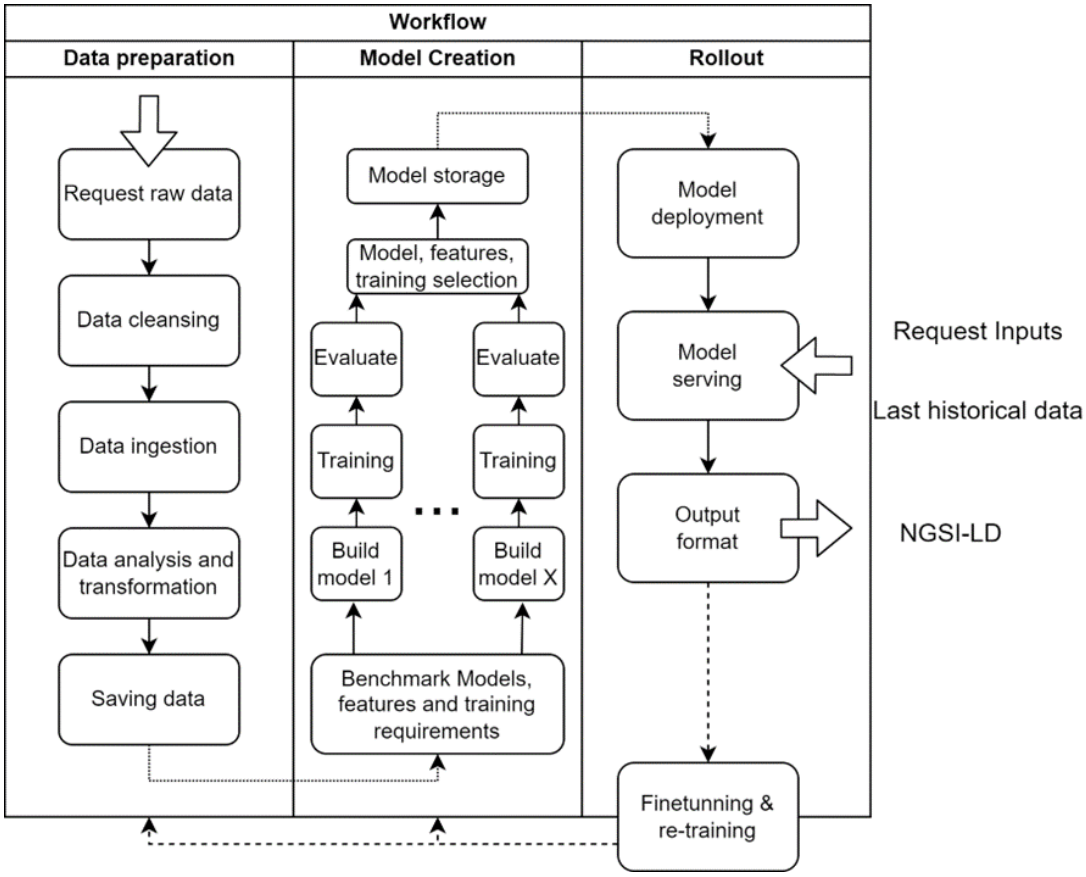

Figure 35: Workflow for Noise forecasting service for each location

In the next section all the technical aspects and selected technologies to accomplish the functions described are further discussed.

5.1.4.2 Architecture

Figure 36 shows the overall architecture of the traffic forecasting service. The whole service is dockerized using docker-compose for fast and easy replication, monitoring and maintenance. FastAPI is used to provide an API to the users of the service. When FastAPI service receives a request for a forecast, it launches a set of functions that will retrieve the models' parameters corresponding to the location requested using joblib library. If no model exist for this location, then a random forest model is built in real-time with a choice of estimator equal to the average of the random forest models used for forecasting at close locations and trained with training data (historic values) requested to the context broker. The efficiency of random forest models makes it possible to run such a process in real-time. If the model already exists, the noise forecasting service will request the last values from the context broker to get the current values of noise emissions, that will be used for the inference.

Then, the inference phase consists in a series of inferences, from an inference of the next hour's noise level to the noise intensity at the requested forecasting time. Therefore, the noise level of all the intermediary times (hours) between the time at which the request was sent and the time for which the forecast was asked for are predicted. All these intermediary predicted values are sent to the context broker for storage in case it can be used by other users. Once the desired forecast time is reached, the data is converted in the right NGSI-LD [7] format and sent back to the end-user.

Then, at the end of the day, a cron job ensures that the model of each location is retrained to capture the trend changes. The updated parameters of the model used are stored using joblib library.

Therefore, inputs of the service are the time, location of the forecast, but also the last historic data from noise level at the given location. Then, the output consist in all the noise level values, from the time at which the request was sent to the noise level at the requested time and location.

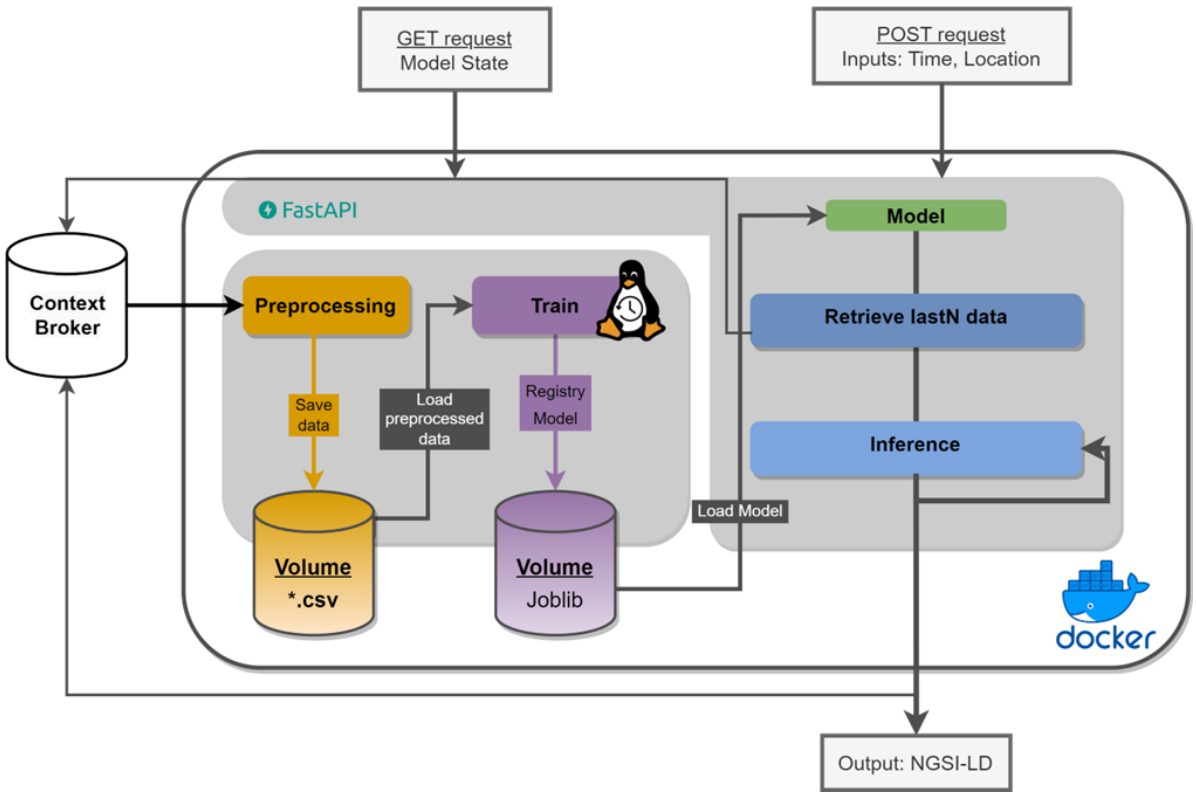

Figure 36: Architecture of the noise forecasting service

5.1.5 Conclusions

Noise level time series follow a periodic pattern that is predictable. The process to provide a forecast consist in data cleaning first, to clean historical data. Then, several models and configurations were benchmarked in order to identify what forecast solutions fits best the shapes of traffic intensity evolution. It was identified that random forest is an accurate technology that does not require too many training data as only two weeks of data are required to achieve an accuracy above 85%. This makes it a convenient technology to be used in an API

configuration. FastAPI allows an easy and fully configurable solution for requests URLs, along with an automatically generated swagger documentation that helps users' guidance in their requests.

The proposed service was dockerized in order to enable quick replication. Each sub service (fastAPI, python functions, ...) has its own docker-image that is managed by a single docker compose file.

5.2 Noise Annoyance calculation

5.2.1 Description

The noise annoyance calculation service aims to provide a comprehensive picture of the noise situation in a given area and to highlight when the noise level and source are not acceptable at a given location and time. One of the key parameters used is the type of noise source based on the area, which can be used to identify the sources of noise pollution such as traffic, industrial activity, or construction sites. Another important parameter is the average age of the residents in the area, as different age groups can be more or less sensitive to noise.

The noise level is also a crucial parameter, as the service calculates the noise annoyance using various models, such as the A-weighted equivalent continuous sound pressure level (LAeq). The resulting noise annoyance calculation provides an assessment of the impact of the noise on the people living in the area. This impact can be significant, affecting the health, well-being, and quality of life of residents.

5.2.2 Model Development

5.2.2.1 Preprocessing

The data requirements to compute the noise annoyance in an area involve the following:

- Area : The area of the calculation of the annoyance
- Period of time : The measurement time
- Type of area (residential, industrial, commercial…)
- Noise level
- Noise level classification
- Average age level in the area
- Dominant noise source in the area

However, before the data can be used in the calculation of noise annoyance, it must undergo a rigorous process of data cleansing and pre-processing. This involves checking the data for any errors, inconsistencies, or missing values, and correcting or removing them where necessary. Additionally, the data may need to be transformed or aggregated to ensure that it is in the correct format for use in the calculation.

5.2.2.2 Rules

In order to make the calculations, values are aggregated from the data sets as shown in Figure 37 below, a mathematical formula uses these values for the calculation, this formula is inspired from the noise disturbance calculation tool developed within IKCEST - International Knowledge Centre for Engineering Sciences and Technology under the Auspices of UNESCO [7].

The formula used is :

$$Noise\ level\ value\ +\ Average\ age\ level\ value\ +\ Noise\ source\ value\ +\ Type\ of\ area\ value$$
$$= \boldsymbol{Noise\ annoyance\ level}$$

Noise sources (based on the area)	Value
Industrial and construction	2
Road and air traffic	2
Enetrtainement and commercial	1.5
Domestic	1

Average age level	Value
0-35	1
35-50	1.5
50	2

Noise level	Value
40-50 dB	1
50-60 dB	2
60-65 dB	3
65-70 dB	4
70-80 dB	5
> 80 dB	6

Type of area	Value
Residential	2
Commercial	1.5
Mix	1.5
Industrial	1

Figure 37: Noise annoyance calculation parameters.

The outputs from the calculation are shown in the Figure 38 below:

Noise annoyance	Value (from to)
Very calm	0-1
Calm	1-2
Good	2-3
Acceptable	3-4
Medium	4-5
Moderate	5-6
Annoying	6-7
Very annoying	7-8
Unsupportable	8-9
Dangerous	9-10
Very Dangerous	over 10

Enviromental impact	Value
Good	2
Medium	5
Moderate	6
Unhealthy	7
Dangerous	8
Extremely dangerous	Over 9

Figure 38: Noise annoyance calculation outputs.

Using the values and the outputs from the Figures above, we can make a calculation example :

For an industrial area with average age level of 40 and a noise level of 55 db the noise annoyance calculation is : $2 + 1{,}5 + 2 = 5{,}5$ Which corresponds to a moderate noise annoyance.

5.2.3 Model Deployment

5.2.3.1 Architecture

The internal architecture of the service is proposed in Figure 39. A docker container enables an easy replication of the service. The architecture includes the application of the formula proposed in the previous section along with a local database to store the parameters of the formula.

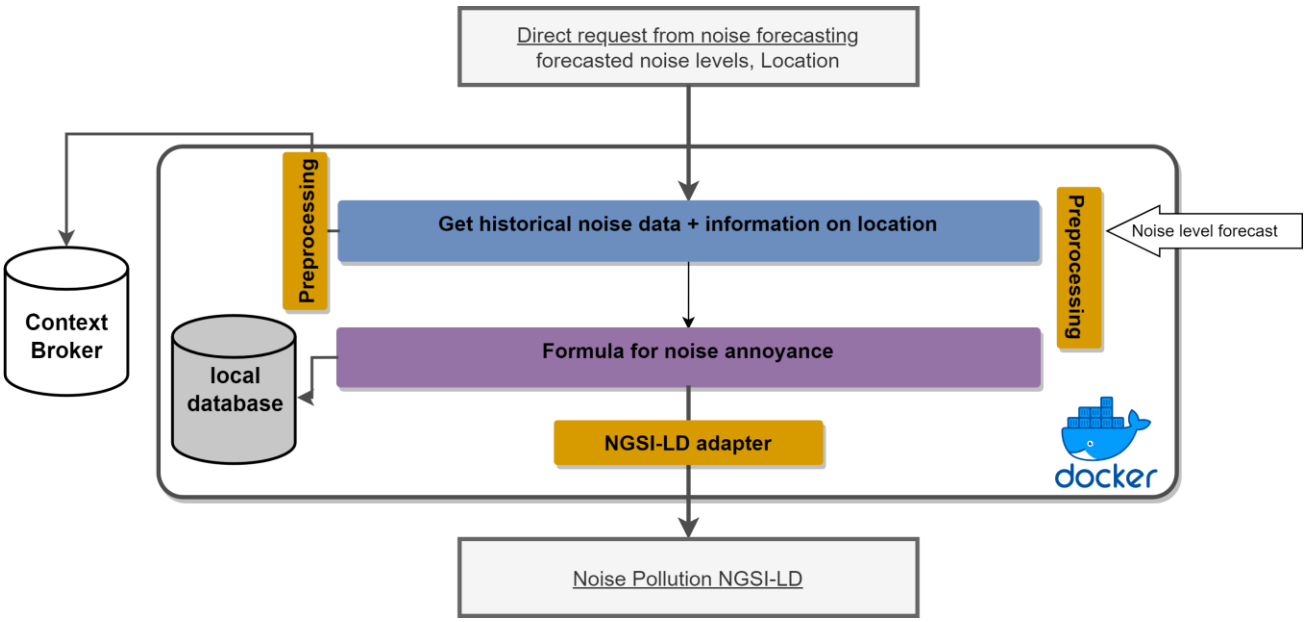

Figure 39: Noise annoyance calculation architecture

5.2.3.2 Workflow

Following the workflow highlighted in Figure 39, the noise annoyance calculation service receives the request directly from the noise forecasting service along with the location and the forecast for the future noise levels. Then, the service will request the context broker to get access to noise pollution information for the considered location. This provides information such as the building type, the noise origin, etc...

Using this information, the formulas described above are used to compute the noise annoyance index, that is then encapsulated into the NoisePollutionForecast smart data model and sent back to the noise forecasting service.

5.2.4 Conclusions

Although noise annoyance is a subjective variable, it can be computed based on local characteristics, such as the building types and source of noise, and based on the noise level. In the case of the noise annoyance forecasting service, the noise forecasting service provides the forecasts that can then be used to compute an annoyance index for a given location, using NGSI-LD [8] formatted data “noise pollution”. The output of this service will then be used to automatically generate traffic recommendations that can help reducing the annoyance from traffic intensity.

6 Conclusions

This document is the report of task 3.2, Green Mobility Service Development, and it is the following implementations of the services defined in task 3.1, Green Mobility Services Definition. Two types of services were identified (calculation and forecasting), as well as four aspects to be addressed (Air Quality, Noise Annoyance, Bikes Availability and Traffic). The service will be used in the use cases as follows: Nice will implement Bikes, traffic and Air quality services, Flanders will use bikes service and Murcia/Molina will apply Air Quality services.

All stages from analysing historical data provided by Activity 5 to delivering each Docker image with the machine learning model (or the corresponding rules in case of calculation services) in production are fully explained. The programming language selected to develop the software is Python.

In the case of Forecasting services, firstly, a full statical analysis of historical datasets was made. It provided us useful insights to benchmark and lately choose the machine learning model for each case. Depending on the service, we have implemented machine learning models, such as XGBoost and KNN.

When the balance between resource expenditure and meeting KPIs was achieved, it was time to define the MLOps platform that would enable the system to be deployed into production. Several connectors are developed in order to retrieve new data from the infrastructure. Depending on the use case, we find Context Brokers including Orion-LD or Scorpio, both NGSI-LD compliance. To take advantage of real data provided by the use cases, we have defined daily trains, as well as data updates. Since we will have many models, we introduce technologies like MLFlow, FastAPI and KServe, in order to track trains and deploy the models. All of the above is encapsulated in a Docker image.

As long as there is one docker image per service, a developer could use them as a modular system. For example, if someone wants to use the Air Quality Forecasting but applies other calculation rules, it will be possible, because both services are independent. The same for the opposite case: the developer could use the calculation service but define another module to predict the pollutants. This is possible, to a large extent, thanks to the use of smart data models for information traffic (activity 2).

As future work, two possible lines of work are identified.

- On the one hand, it could focus on improving machine learning models through an approach based on deep learning or researching possible new sources of information. For example, increasing the granularity of the dataset or adding more inputs to the system.
- Similarly, regarding the production deployment of the model, the infrastructure could be segmented to implement a system based on microservices. This would improve the efficiency, reuse, and security of the entire system.

7 References

[1] GreenMov, D3.1 Green Mobility Services Definition, https://green-mov.eu/sites/greenmov/files/public/content-files/2022/GreenMov%20-%20D3.1_Green%20Mobility%20Services%20Definition_v1.0.pdf

[2] Banco Mundial (2022), Población Urbana (% del total), https://www.datos.bancomundial.org/indicador/SP.URB.TOTL.IN.ZS, *retrieved 2023*-02-09

[3] European Environment Agency (2021). European Air Quality Index, https://www.eea.europa.eu/themes/air/air-quality-index, retrieved 2023-01-31.

[4] EMEP/EEA emission inventory guidebook 2009, updated May 2012. 1.A.3.b.i, 1.A.3.b.ii, 1.A.3.b.iii, 1.A.3.b.iv Passenger cars, light-duty trucks, heavy-duty vehicles including buses and motorcycles. EEA, Copenhagen, 2009.https://www.eea.europa.eu/publications/emep-eea-emission-inventory-guidebook-2009/part-b-sectoral-guidance-chapters/1-energy/1-a-combustion/1.a.3.b-road-transport-gb2009-update.pdf

[5] Centro de Estudios y Experimentación de Obras Públicas. Herramienta CO2TA para la evaluación de las emisiones de CO2 del tráfico por carretera, febrero 2013.https://www.miteco.gob.es/es/calidad-y-evaluacion-ambiental/temas/evaluacion-ambiental/Guia_%20de_Usuario_tcm30-190654.pdf

[6] Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range, Wan, X., Wang, W., Liu, J. et al, BMC Med Res Methodol 14, 135 (2014). https://doi.org/10.1186/1471-2288-14-135.

[7] Service, D.R.R.K. (no date) On-line calculation of Noise Pollution Level (LNP) in impact assessment, On-line calculation of noise pollution level (LNP) in impact assessment,Online computing, online calculator, calculator online calculation. Available at: https://drr.ikcest.org/app/sc444 (Accessed: February 27, 2023).

[8] Guidelines for modelling with NGSI-LD, Gilles Privat, Orange Labs, ETSI (2021). https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp_42_NGSI_LD.pdf

[9] A platform for the Machine Learning Lifecycle. MLflow. (n.d.). Retrieved December 10, 2022, from https://mlflow.org/

[10] Flask. Retrieved September 2022, from https://palletsprojects.com/p/flask/ .

[11] FASTAPI. Retrieved February 2023, from https://fastapi.tiangolo.com/

[12] Model Inference Platform KServe. Retrieved November 2022, from https://kserve.github.io/website/0.10/

[13] Daemon to execute scheduled commands, Cron. Retrieved January 2023, from https://man.freebsd.org/cgi/man.cgi?query=cron&sektion=8

[14] Running python functions as pipeline jobs, Joblib. Retrieved November 2022, from https://joblib.readthedocs.io/en/latest/

[15] Machine Learning library in Python, Scikitlearn. Retrieved December 2022, from https://scikit-learn.org/stable/