Technical Report

Safe Roads: Traffic management and road safety platform for the ATCLL infrastructure

30th of May 2023

Authors

103154 João Fonseca

103183 Diogo Paiva

103541 Pedro Rasinhas

103668 Gonçalo Silva

103696 Catarina Costa

Advisors

Prof. Drª Susana Sargento, Prof. Drº Pedro Rito, MSc. Ana Almeida, Hugo Leal

Project in Informatics 2022/23

Licenciatura em Engenharia Informática

Universidade de Aveiro

Abstract

The idea of having a modern inter-connected city is getting stronger with the growth of Wi-Fi connected devices and the evolution of IoT. Therefore, it was proposed the development of a traffic management and road safety platform, which could process and analyse real-time radar and camera data to generate traffic events around the city.

Adding to the already existing congestion events, whose algorithm was adapted from last year's project, it was possible to detect potholes and vehicles driving in the wrong way. To detect potholes, we trained a YOLOv8 model with a labelled pothole dataset. The resulting model is periodically run against camera footage originated from the sensors in our data processing module. The detection of vehicles driving in the wrong way consists in an algorithm which uses vehicle radar positions and considers a line to separate the two ways of traffic in the road. Above and below this line it is considered a specific correct way of traffic, and vehicle positions shift must comply with it, otherwise they are considered to be driving the wrong way.

Integration with data from HERE and OpenWeather enriched the data collection. From HERE, it was possible to obtain traffic events and flow lines on the road, allowing for a greater diversity of information to display in the platform. From OpenWeather, it was possible to obtain weather data to display on the platform graphs and to help creating correlations with traffic flow under different circumstances.

The web page is integrated in the existing ATCLL platform, one is able to view and filter events by type, source, location and date interval. Furthermore, it is possible to view event details and check traffic flow on the map. It supports graphs where it is possible to compare events against traffic flow and weather data in a dedicated tab, which also includes relevant static graphs results from correlations.

Traffic flow was correlated with aspects such as weather, school and days of the week. Results confirm higher traffic during bad weather, school periods and on weekdays. A traffic flow prediction model using a LSTM deep learning approach was developed. Its performance was satisfactory, however due to the lack of updated data which results in imprecise predictions for the near future, its use on the platform was not implemented.

Acknowledgements

Our team would like to acknowledge and thank all the Aveiro Tech City Living Lab (ATCLL) team for all their help and availability during the conception of the project. We extend special appreciation to our advisors, Prof. Drª Susana Sargento, Prof. Drº Pedro Rito, MSc. Ana Almeida and Hugo Leal for their exceptional guidance and for providing us with this remarkable opportunity. Furthermore, we would also like to acknowledge the contributions of Pedro Figueiredo and Gonçalo Perna, whose assistance greatly contributed to the success of our project.

Acronyms

4G - 4th Generation

5G - 5th Generation

ALPR - Automated License Plate Readers

API - Application Programming Interface

ATCLL – Aveiro Tech City Living Lab

C-V2X - Cellular Vehicle-to-Everything

IoT – Internet of Things

IT – Instituto de Telecomunicações

LIDAR - Light Detection and Ranging

LoRa - Long Range

LoRaWAN - Long-Range Wide Area Network

LSTM - Long short-term memory

RADAR - Radio Detection and Ranging

SLPs - Smart Lampposts

TSM - Traffic Sensor Management

V2V - Vehicle-to-Vehicle

Wi-Fi – Wireless Fidelity

YOLO – You Only Look Once

Index

1 Introduction

This report aims to detail the context, implementation, results, and discuss the elaboration of the project Safe Roads: Traffic and road safety management platform for the ATCLL infrastructure. This project was done under the context of the Projeto em Informática course.

1.1 Context

Aveiro is a city that is aiming to get more and more technological, making progress towards becoming a smart city, where various devices, vehicles and stations are interconnected and exchanging information to improve the quality of life for citizens. This connectedness allows for the sharing of information with the public, enabling them to access useful insights and services. As a result, it makes Aveiro a city with good urban sustainability and liveability.

The ATCLL consists of an advanced communication infrastructure and urban data management platform that provides a large-scale technology laboratory for researchers, startups, and other stakeholders to develop and demonstrate innovative concepts and products. ATCLL uses fibre technology, reconfigurable radio units, and 5G network services to support the infrastructure. It's a service developed jointly by the Municipality of Aveiro, Instituto de Telecomunicações (IT), Altice Labs and the University of Aveiro.

Aveiro, part of the European STEAM City project, has positioned itself as a dynamic technological laboratory. This laboratory comprises stations located in 44 different areas across the city, incorporating various communication technologies such as 4th and 5th Generation Mobile Network (4G/5G), Vehicle-to-Vehicle (V2V) communication, infrastructure based on IEEE 802.11p, Cellular Vehicle-to-Everything (C-V2X), Long Range (LoRa) and Long-Range Wide Area Network (LoRaWAN), as well as Wi-Fi. These stations are strategically installed in SLPs (Smart Lamp Posts) and buildings throughout the city. They are equipped with environmental sensors, Radio Detection and Ranging (RADAR) systems, Light Detection and Ranging (LIDAR) sensors, video cameras, and edge computing capabilities.

All of this infrastructure is seamlessly linked to the IT data centre, which includes edge and cloud computing resources, as well as data aggregation units. This integrated infrastructure creates an ideal environment for the development and experimentation of future services and applications in a real urban setting.

Since the beginning of the project, the ATCLL group has been detecting vehicles via radars and people through the analysis of video streams. These already existing projects were essential to better understand how to receive data in real-time from the city's stations and process the information, persisting it and analysing the obtained data. The group already processed some information from the SLPs, that served as a starter point for the project and was given access to the ATCLL broker.

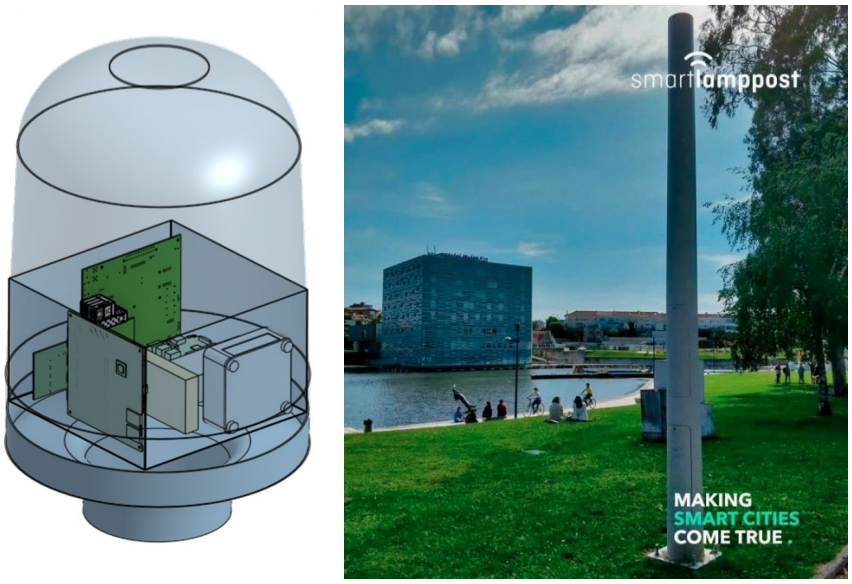

Fig. 1 – Example of a Smart Lamp Post.

1.2 Motivation

To further enhance the ATCLL infrastructure and continue enhancing the quality of life for citizens, while expanding research projects, a proposal has been put forth to develop a platform that analyses different types of data and displays different types of events, level of traffic flow in the city and graphs regarding the events, weather data and traffic flow.

1.3 Goals

The main goal of this work is to develop a platform for the ATCLL dedicated to road safety, by collecting events and analysing traffic flow. The gathered data from the different types of sensors and APIs are analysed in real-time and alerts on the different types of events are generated and displayed on the platform. Radars should be used to detect vehicles that are driving in the wrong way by the means of a custom algorithm, and video cameras should detect the presence of potholes on the road. A weather API should allow the displaying of temperature and humidity values, but also be used to correlate with events and traffic flow. Correlations between aspects such as weather, school and days of the week should be developed. A traffic flow forecasting model should be trained with historical data, and should be able to predict future fluctuations of traffic around the city.

2 State-of-art

In this chapter, we will present our overview on the state-of-art, as part of the development of this project. We will expose some of the technologies and projects related to our own, Safe Roads, and what aspects they share with ours.

2.1 Aveiro Tech City Living Lab

Under the Aveiro Tech City Living Lab initiative, the City of Aveiro assumed itself as a living technological laboratory. This laboratory consists of stations, with various communication technologies, which are installed on Smart Lampposts and buildings throughout the city.

This infrastructure is connected to the IT's datacentre, which served as our main source of inspiration because our road safety system is integrated into this platform, already developed by the ATCLL team.

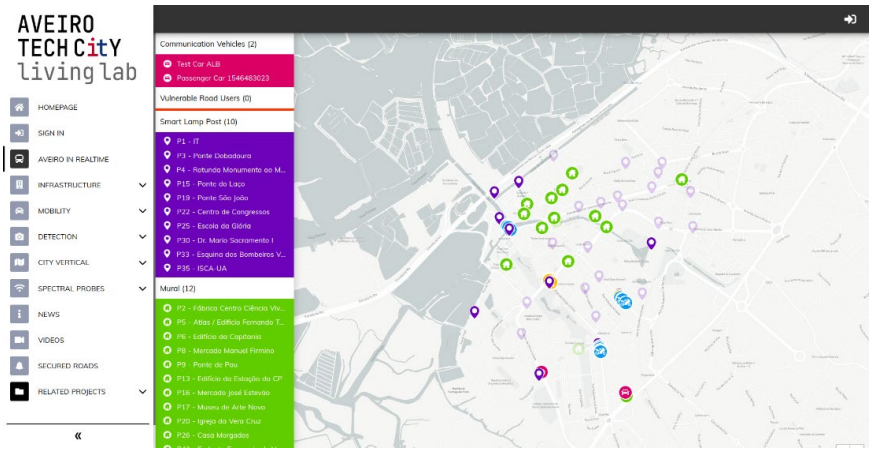

Fig. 2 – Aveiro in Realtime page on the ATCLL web platform.

2.2 Genetec – Traffic Sensor Management

This project, developed by Genetec, consists in a security centre module that can ingest near-real-time traffic data captured form various traffic sensors. It analyses the gattered data to provide object counts and classifications, object speeds, and traffic volumes. Thresholds can then be set on this data to generate automated alerts.

The main goals of the TSM are:

- **Monitoring of traffic flow:** by tracking the traffic volume, speed, and occupancy of moving vehicles and pedestrians throughout your infrastructure or within specific geofenced areas;
- **Tracking of area occupancy:** by collecting occupancy data in defined areas based on vehicle count, object count, people count, or device count;
- **Detecting developing incidents:** by setting thresholds to receive alerts when traffic flow changes unexpectedly, consequently using a robust correlation engine to identify and validate incidents;
- **Management of parking areas:** to help drivers find available parking spots.

The TSM uses a variety of compatible sensors, such as roadside traffic sensors, LIDAR, ALPR, video analysis, wireless network access point and IoT sensors. This work shares some similarities with ours especially in terms of the data collection sector and overall goals.

Fig 3: Traffic Sensor Management dashboard.

2.3 Adaptive Traffic Signal Control System

This GitHub project intends to build a self-adaptive traffic light control system to solve traffic congestion. This methods is based on You Only Look Once (YOLO), an object detection based on a deep convolutional neural network. Traffic signal phases are optimized according to collected data, mainly queue density and waiting time per vehicle, to enable as much as more vehicles to pass safely with minimum waiting time. YOLO can also be implemented on embedded controllers using a transfer learning technique. This project shares some similarities with ours, since it uses YOLO and its bounding boxes to detect objects.

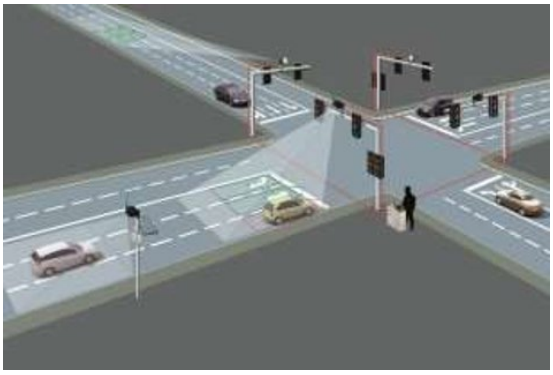

Fig. 4: Visualization of the system installed in a traffic intersection.

2.4 Last Year's Project – Safe Roads

Safe Roads is an already existing project implemented in the ATCLL infrastructure, that integrates Twitter and a city sensing platform to generate alerts based on sensor information and relevant tweets.

The Twitter bot publishes alerts generated by the platform and also analyses tweets of possible traffic jams and reported incidents. This bot uses machine learning to identify tweets that are likely to be about traffic incidents and their locations. Using the ATCLL cameras and YOLO, they detect when a pedestrian crosses the street outside the crosswalk. With the radars, they chose to implement a way of detecting cars above the speed limit.

They also opted to create periodic statistics to demonstrate the system's usefulness, by publishing an alert every hour when there is enough traffic to report. This can help civil authorities understand the road movement thought the day, and identify areas where they can act, for example, by enforcing speed control in areas where there are a lot of cars going above speed limit.

There are many similarities between this project and ours. Both projects analyse information from various sources, such as the cameras and the radars, and our project implemented an algorithm used to detect congestions.

3 Conceptual Modelling

This chapter provides the requirement gathering process, from which we were able to identify actors and their use cases, allowing us to formalize system requirements, taking into account some assumptions and dependencies associated with implementing them.

3.1 Requirements Gathering

From meetings with the advisors, brainstorming sessions as a team and analysis of similar projects, we were able to gather some requirements. These include the development of a web platform to visualize our results, generation of events based on real-time sensor data, integration with events from external mobility applications, analysis of uncommon periods in the city by correlating traffic flow, events, weather and unusual periods, and the training of a machine learning model with historical data to predict traffic flow.

3.2 Actors

Combining all of the ideas from the requirements gathering, we were able to identify two main types of actors:

- **Citizen:** makes use of the platform to check traffic events in the city of Aveiro to plan his daily commute;
- **City Authority:** makes use of the platform to check traffic events and traffic behaviour in the city of Aveiro, with the objective of analysing the influence of events, weather and unusual periods on traffic flow.

3.3 Use Cases

Different actors will use the system for different reasons. As previously mentioned, while an authority seeks to use our system to have a more analytical view, a ordinary citizen will use it as a real-time information tool, to plan his daily commute

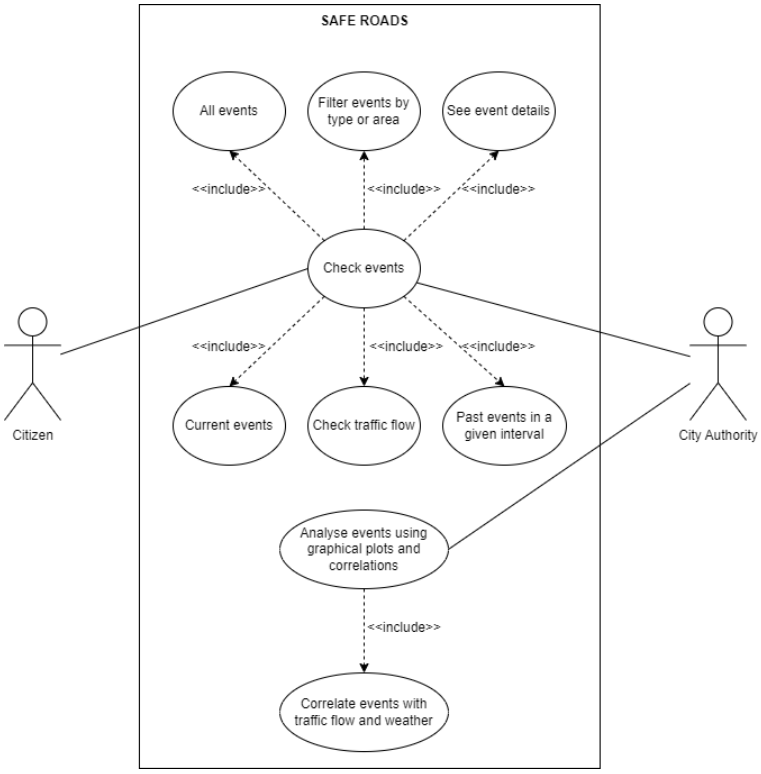

Fig. 5: Use cases diagram.

3.3.1 Citizen

The citizen wants to avoid running into traffic congestions, accidents, and restricted roads. He is also interested in knowing the likelihood of a traffic jam during his commute. Based on this information, he can plan his commute ahead, avoiding eventual hassles along the way. He should be able to view all the events, with the possibility of filtering them by some parameters such as type, source, location and data interval, check their details, and check current traffic flow.

3.3.2 City Authority

The city authority will be considered as a user that wants to use our system to analyse how events, weather and unusual periods can affect the traffic flow. For that, it requires access to an history of past events. Graphs to conveniently compare events, traffic flow and weather data should be available to it. Based on this information, it can support transportation planning decisions to improve overall traffic flow and safety in the city.

3.4 Requirements

3.4.1 Functional Requirements

Reference	Functional Requirements
RF-1	The system should show a map of the city of Aveiro.
RF-2	The system should display the events on the map and on a list.
RF-3	The system should allow filtering of events based on type, source, location and data interval.
RF-4	The system should provide events detected on the ATCLL infrastructure and from mobility applications, such as HERE.
RF-5	The system should support events of the following types: accident, road work, congestion, road hazard, immobilized vehicle, closed road, police presence, flood, pothole, wrong way.
RF-6	The system should display the details of an event when the user clicks it on the map or on the list.
RF-7	The system should display traffic flow on the map using colours.
RF-8	The system should provide a graphic plot to visualize traffic flow in the provided time, period for the specified area.
RF-9	The system should allow for a correlation analysis between traffic flow, events, weather and unusual periods.
RF-10	The system should comply with the ATCLL, HERE and OpenWeather politics and regulations.

3.4.2 Non-Functional Requirements

There are 4 types of non-functional requirements: performance (RNF-1, RNF-2), usability (RNF-3, RNF-4), compatibility (RNF-5) and maintainability (RNF-6).

Reference	Non-Functional Requirement
RNF-1	The system should provide a low response time for every interaction.
RNF-2	The system should detect repeated events.
RNF-3	The system should provide an intuitive interface and be easy to navigate.
RNF-4	The system should be fully integrated in the ACTLL infrastructure.
RNF-5	The system should be able to integrate data from other systems, applications and platforms (HERE, OpenWeather).
RNF-6	The system should be well documented.

3.5 Assumptions and Dependencies

- **Assumptions:**
  - The ACTLL, HERE and OpenWeather are always up and available.
  - Users can fully interact with the platform without any help.
  - The system doesn't experience performance bottlenecks.
  - The events from external systems are always correct.
  - All devices can access the platform.
- **Dependencies:**
  - On data that is provided by HERE and OpenWeather.

4 Procedure

In this chapter, we will explore the details of the system implementation as well as the thought process that led us to it. We will start by discussing our architectural choices and the deck of the technologies used, including the importance associated with each of them. Following this, we will identify all of the entities involved in the process, and how they interact between each other. Finally, we will explain how the deployment of our system in the ATCLL infrastructure was performed.

4.1 Technology Architecture

The architecture can be separated into three different modules: one which contains submodules that have already been implemented and don't need any change (sensor data, the IT API and the IT database), another which contains the submodules that have already been implemented but need some change (presentation layer and the IT broker), and the last one which contains the submodules that were totally developed by us (data processing, API, database and connection to external systems). In the presentation layer submodule, the technology in use was Nuxt.js and it was kept that way. Communicating with the backend, we have an API developed with FastAPI. Associated with the API we have a MongoDB database. We used MQTT as the message broker, which will receive data from previously defined topics and temporarily store the information coming from the sensors. It will also be used for the communication between our backend submodules, transmitting information from the data processing submodule to the API submodule. We integrated data generated from external systems (HERE and OpenWeather) and also made calls to the IT database to get access to their historical data.

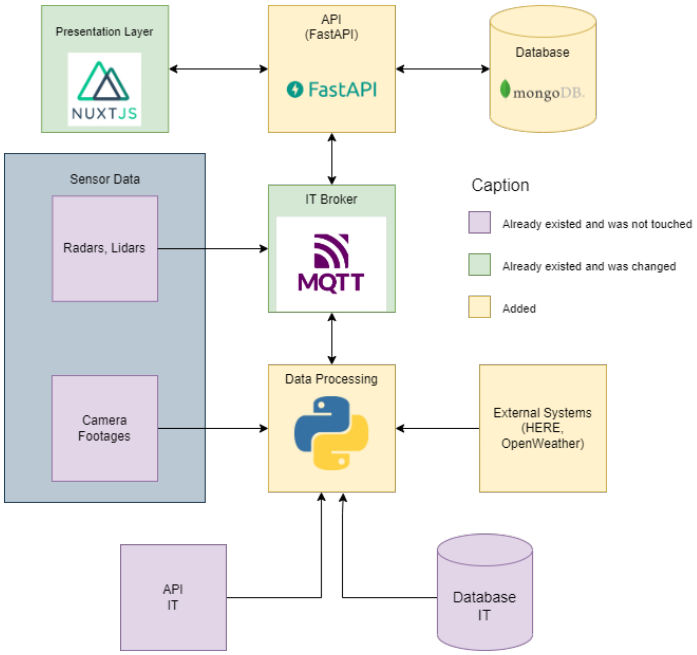

Fig. 6: Architecture diagram.

4.2 Domain Model

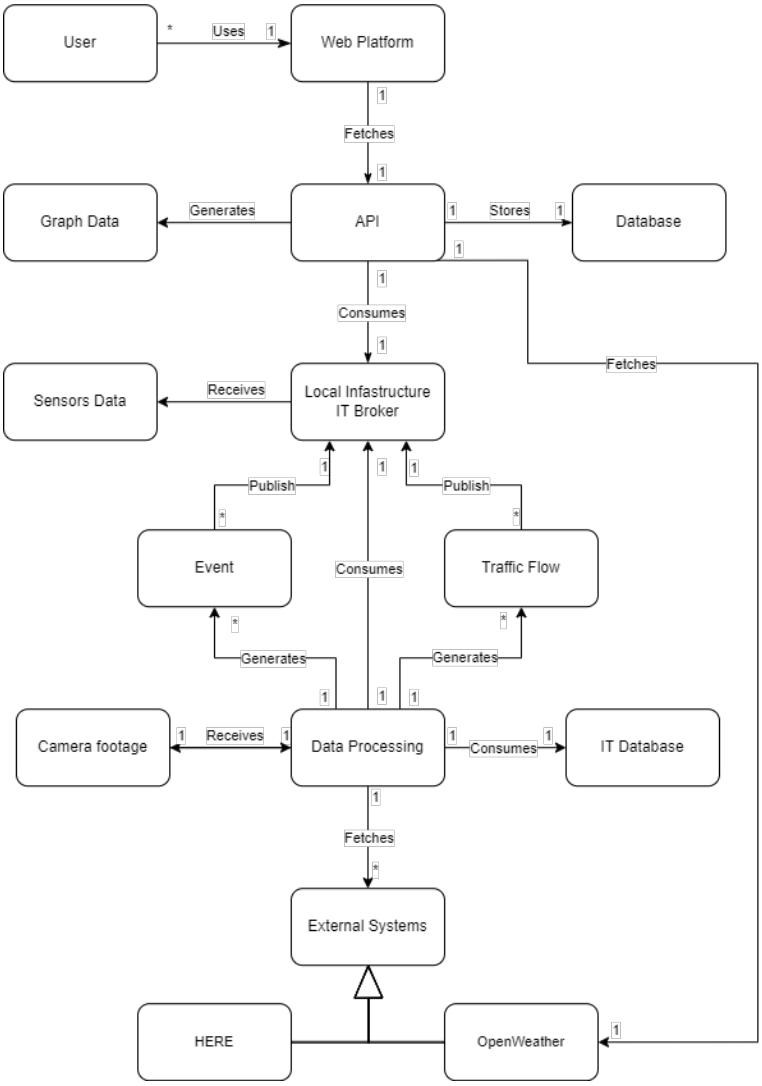

Fig. 7: Domain model diagram.

4.3 Deployment

Our system will be deployed on a virtual machine, hosted in the IT2 building. It will be constantly running all of our backend and frontend modules. Our frontend will be deployed in the development ATCLL web platform[1] and our API endpoints can be accessed through a specific endpoint[2].

[1] Available at https://dev.aveiro-living-lab.av.it.pt/.
[2] Available at https://api.atcll-data.nap.av.it.pt/.

5 Frontend

This submodule, in its most significant part developed and built before us, was built on Nuxt.js, a free and open-source JavaScript library based on Vue.js, Node.js, Webpack and Babel.js. Nuxt.js is inspired by Next.js, which is a framework that has similar purpose, but is based on React.js.

Considering as a starting point the already developed platform, we added our own Safe Roads tab, which at a first glance shows all the events in the city, along with the traffic flow lines in the road. A board on the top left side is displayed and allows the user to filter events by type, source, location and date interval.

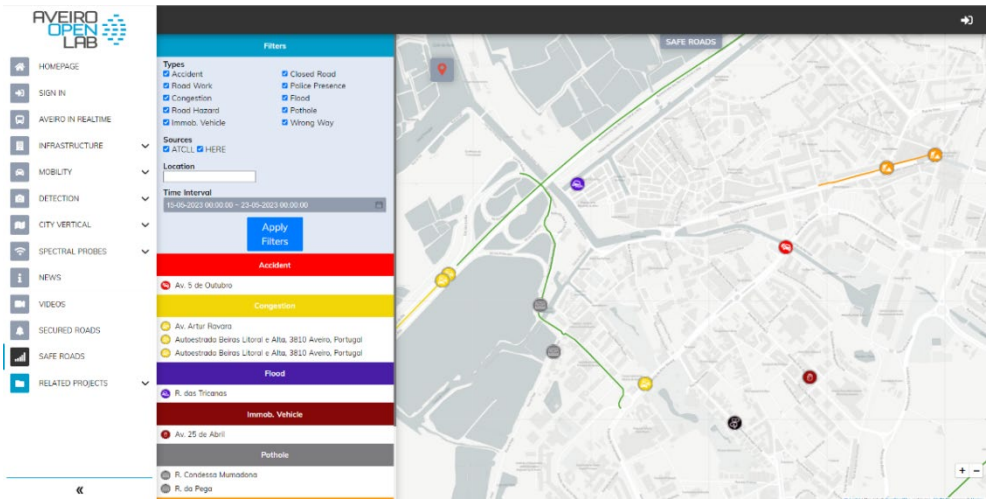

Fig. 8: Safe Roads map page with filter board and event list.

By clicking on an event, either on the map or on the list, we are presented with its details, regarding its type, course, location, description, start and expected end date, and the date it was generated.

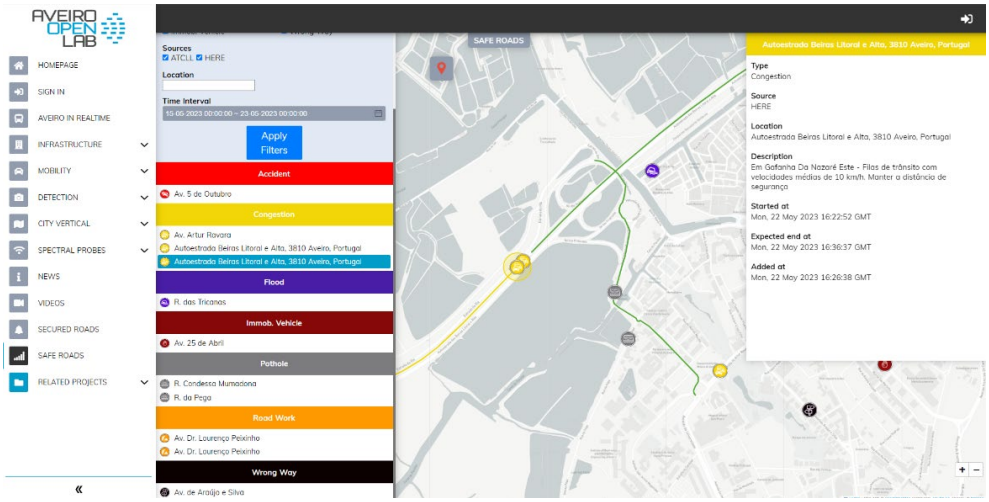

Fig. 9: Safe Roads map page with event details.

We also added a button to toggle between showing the event details and graphs. In the graphs' page we display some dynamic graphs with the number of events, traffic flow and weather data (temperature and humidity) for the date interval defined in the filters. There are also included some static graphs, including relevant correlation analysis graphs and the traffic flow prediction model's performance.

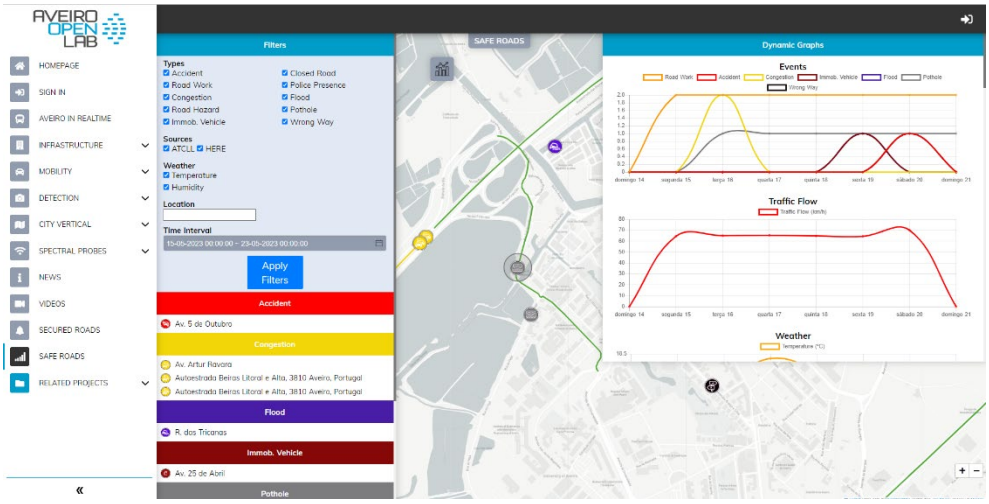

Fig. 10: Safe Roads graph page with dynamic graphs.

5.1 Sidebar

To allow the users to access the page, we implemented a new section on the file index.js, to redirect to our page.

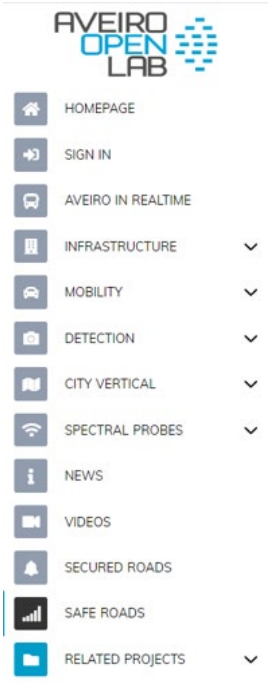

Fig. 11: ATCLL updated sidebar.

5.2 Main Page

5.2.1 Previously Implemented Resources

- **Map:** using Leaflet and Open Street Maps (OSM), two open-source resources for map making.
- **Page structure:** SidemapLeft and SidemapRight components.

5.2.2 New Implementations

- **Filtering:** the user is able to filter events, by selecting options displayed on the "Filters" board. There are four main sections: "Types", "Sources", "Location" and "Time Interval". If the user is on the graph page, there is another main section called "Weather", which allows the display of the respecting weather graphs. This allows for the user to control all the information that is displayed on the map.

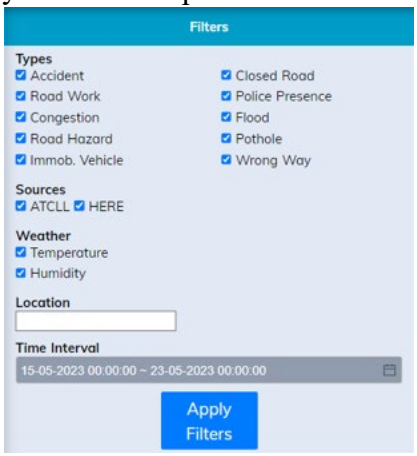

Fig. 12: Filter board.

- **Traffic Flow:** using the information gathered from the APIs regarding the traffic flow, we are able to display on the map the level of traffic flow in certain roads. If the colour of the street is green, it means that the traffic flow is in a low level; if it is yellow, it means that the traffic flow is at a moderate level; if it's red, it means that the traffic flow is at a severe level.

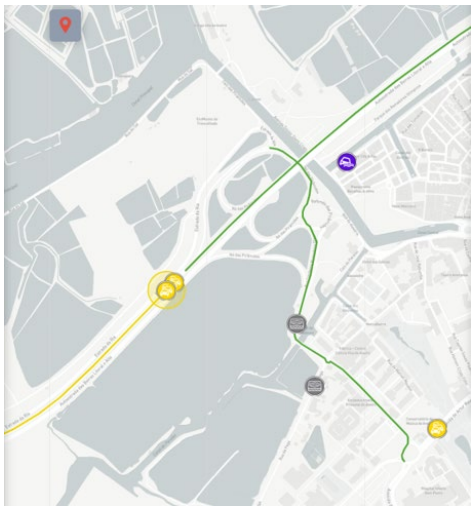

Fig. 13: Traffic flow lines.

- **Graphs:** we added graphs informing on the number of events, traffic flow and weather data (temperature and humidity), in which the user can make comparisons. Since the prediction of traffic flow is not dynamic, due to the model requiring updated data to be accurate, which the IT database was not able to provide, we couldn't implement a graph that displayed those predictions.

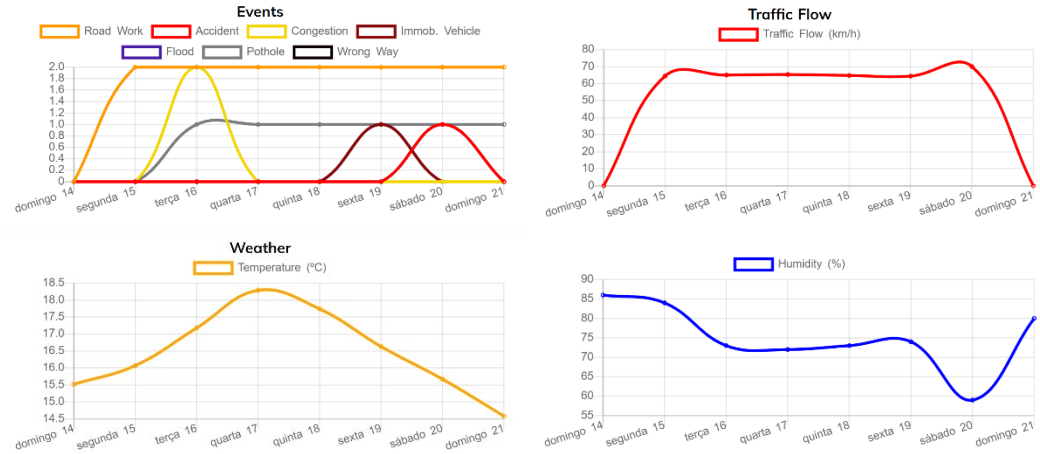

Fig. 14: Dynamic graphs.

- **Correlations and traffic flow prediction:** in addition to the graphs, we have included some images illustrating correlations and the performance of the traffic flow prediction model. These static graphs aim to provide a visual representation of the analysis we conducted of the historical data, which aims to enhance the user's understanding of the data, and simplify the interpretation of the findings.

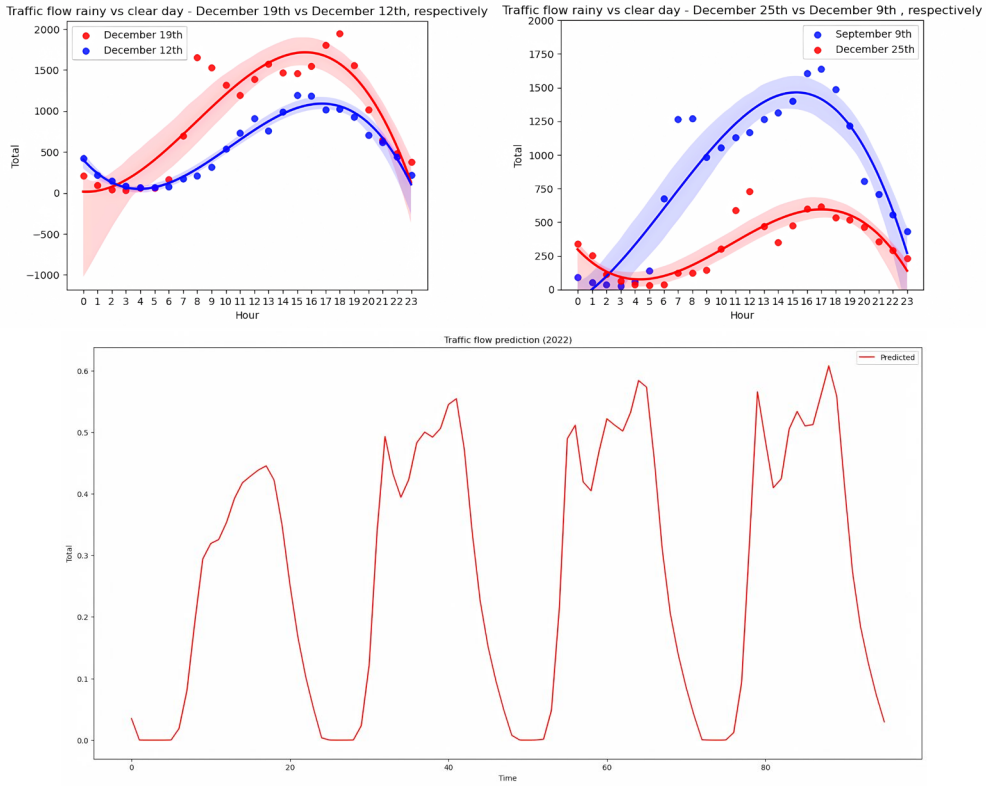

Fig. 15: Static graphs.

- **Fetching the API:** due to presents requests on the project made by previous developers, we were able develop our own *axios* method, that allows us to work with our API without compromising the rest of the developed platform.

6 Backend

Our backend module is divided into two main submodules:

- **API:** developed in FastAPI, stores the events and traffic flows on a MongoDB database and has endpoints to get and filter events, traffic flows and graphs.
- **Data Processing:** developed in Python, receives the data from ATCLL sensors and the external APIs (HERE and OpenWeather), analyses it and creates the events and traffic flows that will be sent to our API for later storage.

These modules are able to communicate by using MQTT. We created two topics to exchange information between these two modules: "/events" and "/flows".

The "/events" topic stores the events generated in the data processing module and converted events fetched from the HERE API. The "/flows" topic stores the traffic flows generated in the data processing module and converted traffic flows from the HERE API.

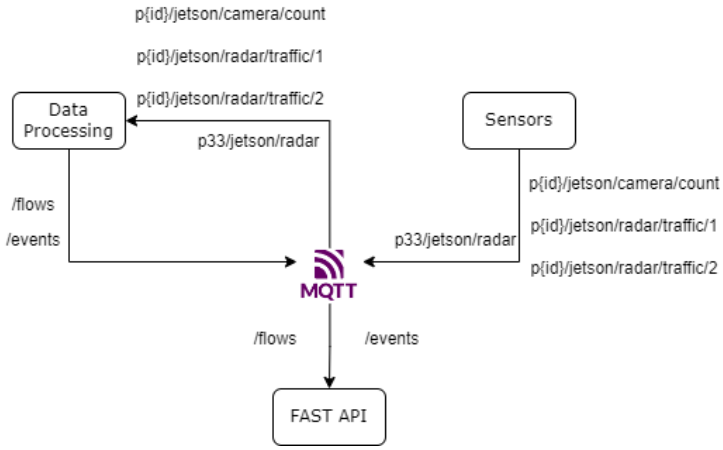

Fig. 16: MQTT data exchange.

6.1 Analysis of the sensor's data

We are collecting ATCLL SLP data that is being published via MQTT topics 8"p{id}/jetson/camera/count", "p{id}/jetson/radar/traffic/1", "p{id}/jetson/radar/traffic/2" and "p33/jetson/radar"9, which provide useful insight about the traffic in the city of Aveiro. The events and traffic flows created by the data processing module are triggered by the values returned by the radars and cameras around the city of Aveiro.

6.1.1 Traffic congestion events

One of the objectives of this system is to check in the traffic in real-time around the city of Aveiro, and for that we analyse the data that comes from the ATCLL radars and cameras. We adapt this process from the project of the previous year and because of its similarity. We collect from each SLP radar/traffic topic the average speed of all cars and the total number of cars in a lane and join it with the number of people from the camera topic. With that information we do a series of checks to make sure the combination of those values should be considered high traffic or not. There are some thresholds in these checks: average speed, number of cars, number of people and traffic count threshold.

If the values combined follow at least one of our predetermined conditions in their order, the traffic count of that SLP will increase by one. Otherwise, the count will be reset to zero. And when the traffic count threshold is exceeded by the traffic count, it is considered a traffic congestion, but it will only be published and consequently persisted by the API, if there wasn't a similar event in the last thirty minutes. This approach allowed us to change the thresholds in order to get more or fewer traffic alerts, that was helpful to test our system and check if it was working properly and if we want our system to be more "sensible" we also can. These traffic alerts are generated in real-time, so it doesn't use any machine learning or historical data, and is based only on predetermined conditions.

6.1.2 Data From HERE

The events and traffic flows that come from HERE don't need to be processed along with the data from the ATCLL radars and cameras. The events and traffic flows are only parsed into the structure that we created for our system.

6.1.3 Event and Traffic Flow Structure

After all the data processing and fetching of the external APIs, the data processing submodule sends a JavaScript Object Notation (JSON) message with that information to the MQTT topics. The API subscribes to all the topics and expects to receive these kinds of messages.

There are two types of messages:

- **Events:** these have information about a specific event. They have information about the type, source, location, description, the points so that it can be draw in the frontend, the start, and the end dates, and the date it was added.


```
{
    "type": "accident",
    "source": "here",
    "sourceid": "2806373953884673234",
    "description": "Acidente",
    "location": "Av. 5 de Outubro",
    "geometry": [
                    {
                        "points": [
                            {
                                "lat": 40.640887,
                                "lng": -8.648147
                            }
                        ],
                        "length": 72.0
                    }
                ],
    "start": "2023-05-20T08:20:33Z",
    "end": "2023-05-20T08:30:33Z",
    "timestamp": "2023-05-20T08:20:33Z"
}
```


Fig. 17: Event message structure.

- **Traffic Flow:** these have information about a specific traffic flow. They have information about the source, location, average speed, and the segments and the date it was added. Each segment has a jam factor associated, which is the level of traffic and the points so that in the frontend we can draw the lines.

```
{
   "source":"here",
   "location":"Azurva",
   "avgspeed":12.5,
   "segments":[
      {
         "jam_factor":1.0,
         "geometry":[
            {
               "points":[
                  {
                     "lat":40.63764000311494,
                     "lng":-8.593329973518848
                  },
                  {
                     "lat":40.63764997757971,
                     "lng":-8.593629961833358
                  }
               ],
               "length":26.0
            },
            {
               "points":[
                  {
                     "lat":40.63764997757971,
                     "lng":-8.593629961833358
                  },
                  {
                     "lat":40.63767998479307,
                     "lng":-8.594530010595918
                  }
               ],
               "length":77.0
            }
         ]
      }
   ],
   "timestamp":"2023-05-17T21:28:16Z"
}
```


Fig. 18: Traffic flow message structure.

6.1.4 API Documentation

In order to provide the events and traffic flows in the web platform, our API is subscribed to all the topics created by us in the MQTT broker, and receives the JSON messages generated by the data processing submodule to store in the MongoDB database. Our page in the ATCLL web platform fetches the events, traffic flows and graph data from our API, so that they will be shown in the platform.

Endpoint	**Purpose**	**Parameters**
/events	Get all events or filter events with the parameters	type source location start end
/flows	Get all flows or filter flows with the parameters	start end
/graphs	Get labels and data to make the graphs and filter with the parameters	type source location start end weather

6.1.5 Pothole Detection

As a way to extend the events we got through external APIs and to take advantage of the cameras installed on the pole infrastructure all over the city of Aveiro, we were proposed to develop a machine learning model to detect the presence of potholes on the road. The chosen model was YOLOv8n, a state-of-the-art object detection model. A pre-trained version of this model with the COCO dataset, that detects 80 different types of objects including cars and people, is already in use on the ATCLL infrastructure smart lamp posts[3].

Before training the YOLO model, we needed some annotated data to train it on. In YOLO, this works by having images and associated text files with bounding box annotations about each image. The name of the annotation file needs to match the name of the image file. Each line of the annotation file is based on the Darknet format, and identifies the type of object and its surrounding bounding box on the image: the first value represents the object class, the second and third values identify the relative horizontal and vertical centre point of the bounding box, and the fourth and fifth values identify the relative horizontal and vertical length of the bounding box edges. Instead of collecting images of potholes on the Internet and doing the annotation by hand using an appropriate tool, we did some online research and found a pothole dataset with 665 annotated images in the format recognized by YOLO[4].

After getting the data to train the model, we developed a very simple python script to invoke the training library provided by Ultralytics. A config.yaml file was created to setup the training environment, which referenced the training data location and labelled the pothole class. We ran the training process using 100 epochs in one of the group member's personal desktop computer, and with each epoch taking around 5 minutes to complete, the whole training process took around 8 hours and 20 minutes to complete. Analysing the training results, which contained statistical graphs and results of the model application on test images, we were pretty confident that the resulting model had a good accuracy. To further corroborate the performance of the model, we applied it to two sample videos containing potholes[5], and drew the bounding boxes along with the confidence value of the detection alongside.

It was now time to apply our pothole detection model to the camera streams received from the ATCLL infrastructure. We ran into a small issue when we tried to install our model to run inside the smart lamp posts, since only one object detection model is able to run at any given point. Since the smart lamp posts already run the YOLOv8n trained on the COCO dataset, which is used by other components in the infrastructure that depend on it, we didn't find a feasible way to alternate between the two. Instead, we collect images from the cameras from time to time in the data processing module, and there we apply our model to the frames. If we detect a pothole on some of the frames, we generate a new event. In the end, we didn't have the need to run the pothole

[3] Available at https://github.com/ultralytics/ultralytics.
[4] Available at https://public.roboflow.com/object-detection/pothole.
[5] Available at https://www.shutterstock.com/pt/video/clip-1093430081-bangalore-india-14th-august-2022-view-rain and https://www.shutterstock.com/pt/video/clip-1028973710-bad-rural-road-pits-holes-puddles-after.

detection model at all times given its static nature. The pothole detection is run every day at 13h00, for 10 seconds. The duration of the event is therefore considered to be of one day, and should continue to be created until the pothole on the road is fixed. Since we wouldn't receive the object detection through the ATCLL broker with the geographical coordinates of the pothole, we had to make that conversion in the data processing module. Applying the method already in use in the ATCLL infrastructure smart lamp posts[6], we are able to convert any pixel coordinates to geographical coordinates just by knowing the geographical coordinates of 4 non-linear coplanar points in the image frame. We considered the centre of the bounding box as the position of the pothole on the road.

[6] Available at https://link.springer.com/chapter/10.1007/3-540-44839-X_11.

6.1.6 Wrong-way Driving Detection

Using YOLO

To further extend our event pool, we took advantage of the real-time information already available in the ATCLL broker, related to vehicle positions on the road. We were proposed the development of a method to detect vehicles driving on the wrong side of the road. After some online research, we came upon an article[7], which proposed a method of detecting wrong-way vehicles if we are able to track them on the image frame. This method detected vehicles using the YOLOv3 algorithm, and tracked them with a centroid tracking algorithm. In our case, we wouldn't need to detect and track the vehicles on the road, since work towards that had been previously setup in the ATCLL infrastructure, with the use of radar technology, which has more accurate results than analysing an image frame. However, to help us visualize and test our method in an early stage, we created a similar environment as the one explained in the above article.

We were granted access to real-time imagery from SLP 33, located in the intersection near Bombeiros Velhos, in Aveiro. This access was made available through a RTSP stream from the camera. We then developed a simple Python script that allowed us to extract each frame from the stream. Using AS-One[8], a Python library capable of detecting and tracking objects using YOLO models, we were able to feed its detector with the image frames we got from the SLP camera to detect and track vehicles. We decided to use the YOLOv8n model trained on the COCO dataset, since it is freely available and is capable of detecting vehicles.

After setting up the development environment, we could finally start thinking on the method to detect vehicles driving in the wrong way. In the previously mentioned article, it was considered that a vehicle was driving in the wrong way if its central point was moving up on the image frame. This happened because the correct way of traffic would move towards the camera, which could be considered the same as the central points of the vehicles moving down on the image frame. However, they used their method in a one-way street. In our case, we had a two-way street, so we had to figure out a way to distinguish both traffic lanes and analyse each one individually.

Our first approach considered the position of the vehicles to be the central point of their respective bounding box, determined by the YOLO model. We then divided both ways of traffic by drawing a line on the image frame, almost parallel to the road axis, with only two points on the image frame. Using those two points, we could create a function in the code that would represent that line. This allowed us to temper with the point coordinates until the line was positioned to our liking. This function behaved just like mathematical functions: by inputting the x coordinate on the image frame, we would get the correspondent y coordinate on the line that matched the x coordinate. This allowed us to determine, for the x coordinate of the vehicle, if its y coordinate was above or below the line, allowing us to discover the correct heading it should be moving.

[7] Available at https://arxiv.org/abs/2210.10226.
[8] Available at https://github.com/augmentedstartups/AS-One.

Considering that the y coordinates in the image frame increased from top to bottom, vehicles going Glicínias-Bombeiros Velhos (going right-left in the image frame) should have a y coordinate value lesser that the line for any given x coordinate value, vehicles going Bombeiros Velhos-Glicínas (going left-right in the image frame) should have a y coordinate value greater that the line for any given x coordinate value. Given that their position was the centre of their bounding box, the line had to be adjusted so it would be positioned a bit above the road axis. To figure out if the vehicle was moving in the correct way, we had to keep track of each vehicle's position in the previous image frame. Knowing that, we could discover is the change in the x coordinate for each vehicle was valid given their position on the road: for vehicles going Glicínias-Bombeiros Velhos, the x coordinate value should decrease, whereas for vehicles going Bombeiros Velhos-Glicínias, the x coordinate value should increase. To avoid most false positives, caused by very small shifts in the position of the vehicles' bounding boxes (specially when stopped), we added a threshold to only consider in the detection shifts greater than 5 pixels.

To validate our method, since all vehicles during our experiment were following traffic laws, we flipped from detecting vehicles driving the wrong way to vehicles driving the right way. Doing this was easy, since it only required changing the considered correct way for each lane. Running the detection again, every vehicle on camera was now detected to be driving in the right way, so we were pretty confident that our method's results were satisfactory.

Fig. 17: Wrong way driving detection visualization using YOLO.

We got in contact with our advisors to show how our method worked, but also to gather information on how we could transfer it to use the radar information, which returned geographical coordinates of the vehicles positions, and how could we draw the line separating the traffic lanes in this case. We found out that the considered position for vehicles closely matched the bottom centre of the bounding boxes detected by YOLO in our experiment, so that meant we had to change once again the position of the line, moving it now below the road axis. To figure out the

geographical points of the line, we could use an external map provider (like Google) to mark points on the road that closely matched the position of the line we drew on the image frames, using the zebra crossing lines as guidance.

Using Radars

We decided to adapt our method to use the radar data that comes from the sensors. Our first approach was using the same function, but instead of pixels using geographical coordinates. However, there were too many errors. Some coordinates that were in the right side of the line appeared to be on the other side, revealing too many cars driving in the wrong way. As a solution, we decided it was best to define a line for each way of traffic, allowing for a slight error threshold. This improved our results, reaching a similar level of success as the camera approach.

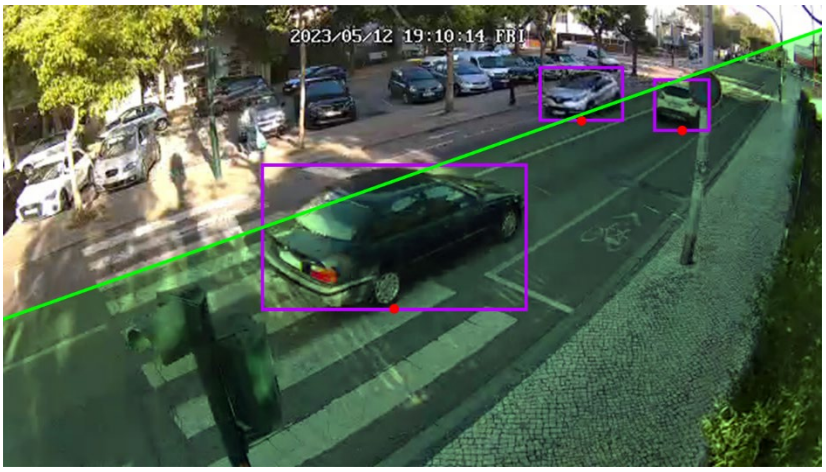

Fig. 18: Wrong way driving detection visualization using radars (Bombeiros Velhos-Glicínias).

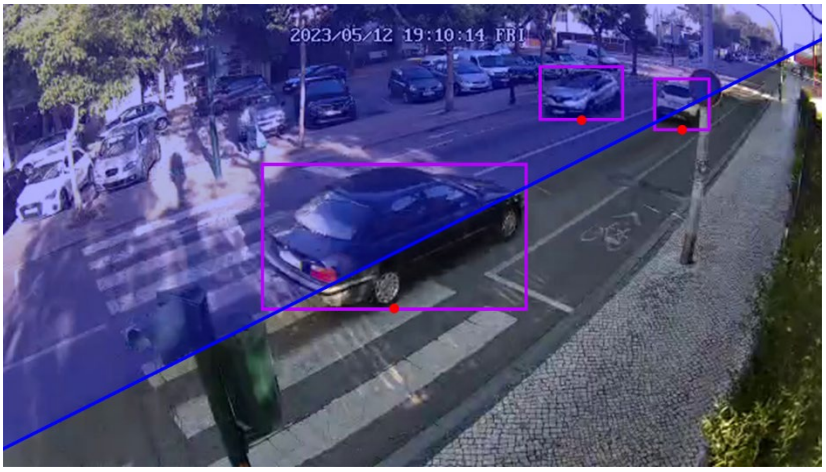

Fig. 19: Wrong way driving detection visualization using radars (Glicínias-Bombeiros Velhos).

6.2 Correlations in Traffic Flow Analysis

When examining variables like days of the year (e.g., school days versus no school days) and weather conditions (e.g., rainy versus sunny days), correlations are crucial to the analysis of traffic flow. For effective traffic management, transportation planning, and ensuring road safety, it is crucial to comprehend these correlations.

One significant aspect is the correlation between traffic flow and the specific days of the year. By analysing patterns in the traffic volume on different types of days, authorities can make informed decisions regarding traffic management strategies, and can even allocate resources more efficiently. Implementing traffic control measures near educational institutions can help mitigate the congestion during peak hours.

The following graphs use the dataset provided by the advisors. The "no school" days go from 01/09/2022 to 18/09/2022 (even though that range contemplates the exams special season). The "school" days are from 19/09/2022 to 04/10/2022. For the following graphs, we only considered the data from the radar 35 (ISCA), as it is the one with the highest traffic volume.

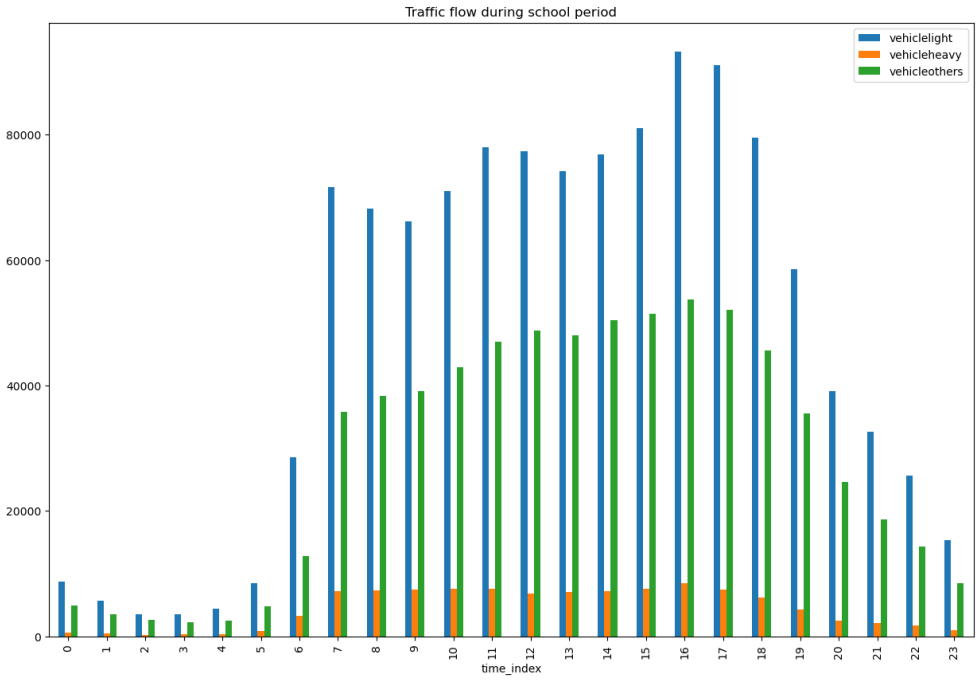

Fig. 20: Sum of traffic volume by hour, from 19/09/2022 to 04/10/2022 (during school period).

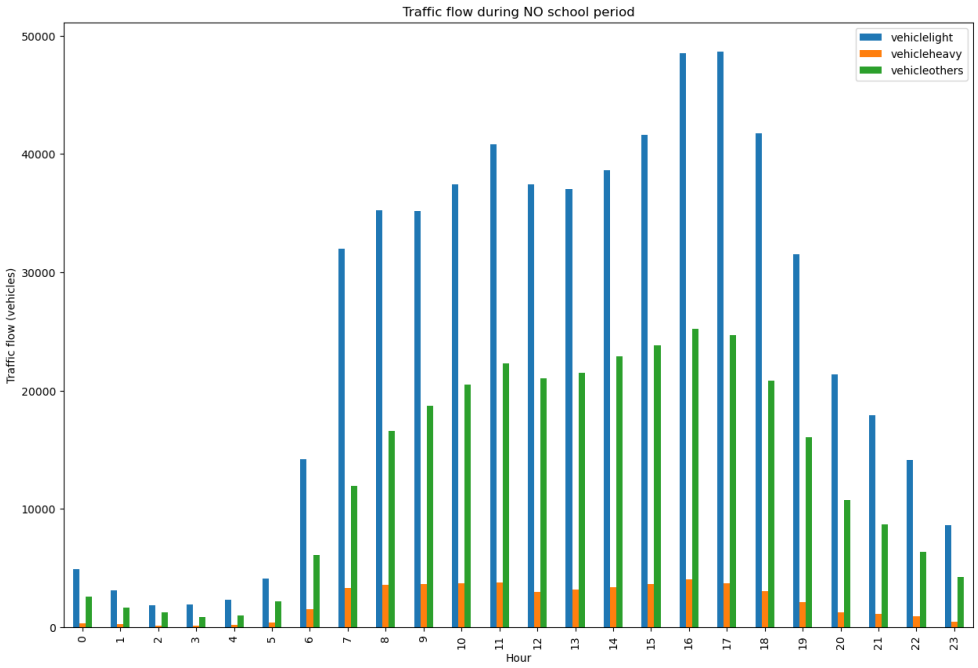

Fig. 21: Sum of traffic volume by hour, from 01/09/2022 to 18/09/2022 (NOT during school period).

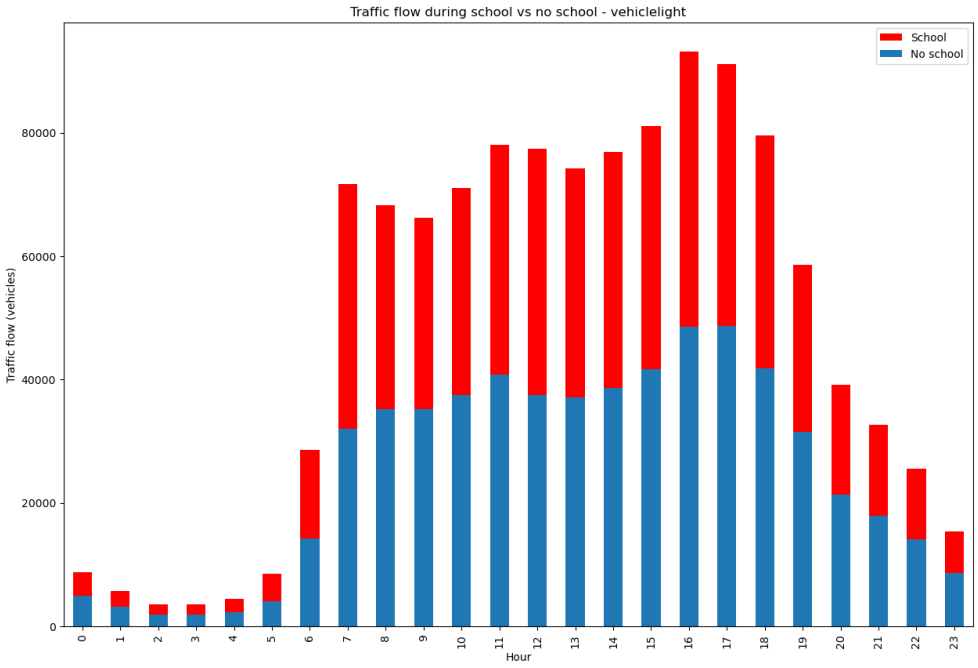

Fig. 22: Comparing Traffic volume of light vehicles, grouped by hour, on different types of days (school vs no school).

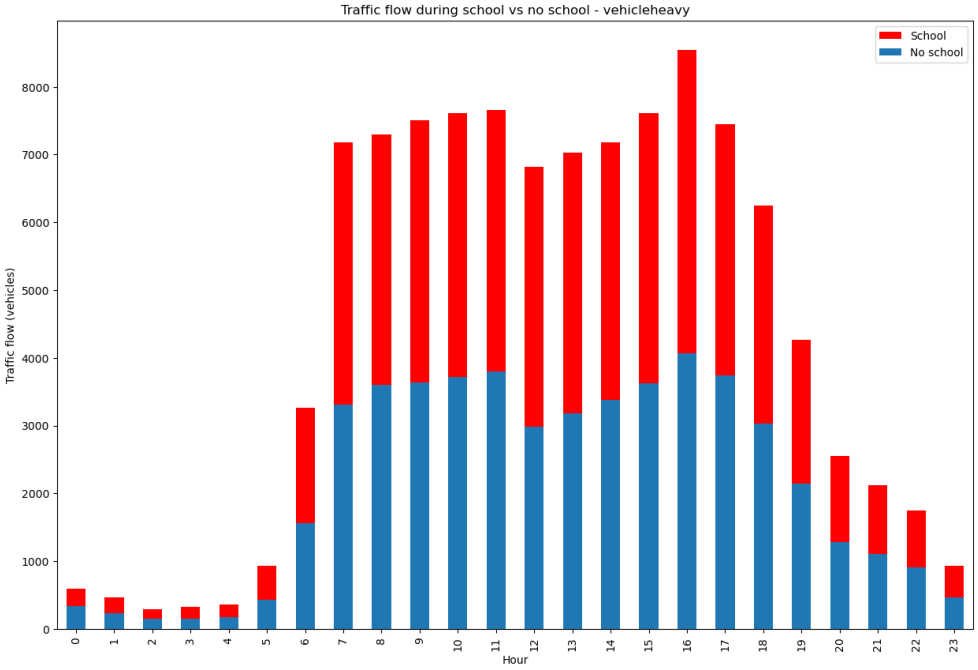

Fig. 23: Comparing Traffic volume of heavy vehicles, grouped by hour, on different types of days (school vs no school).

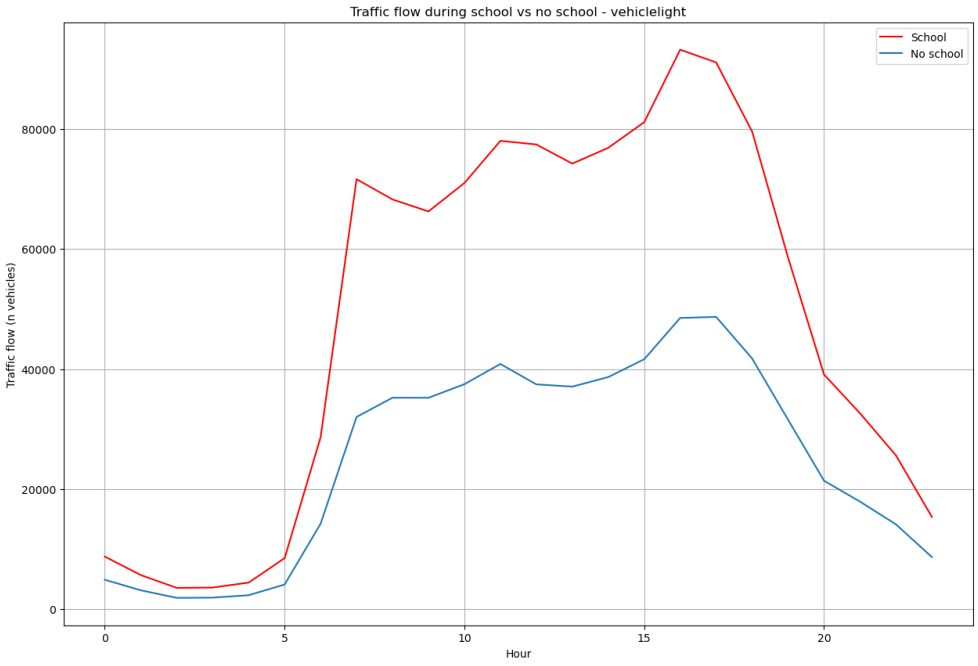

Fig. 24: Comparing Traffic volume of light vehicles, grouped by hour, on different types of days (school vs no school, line).

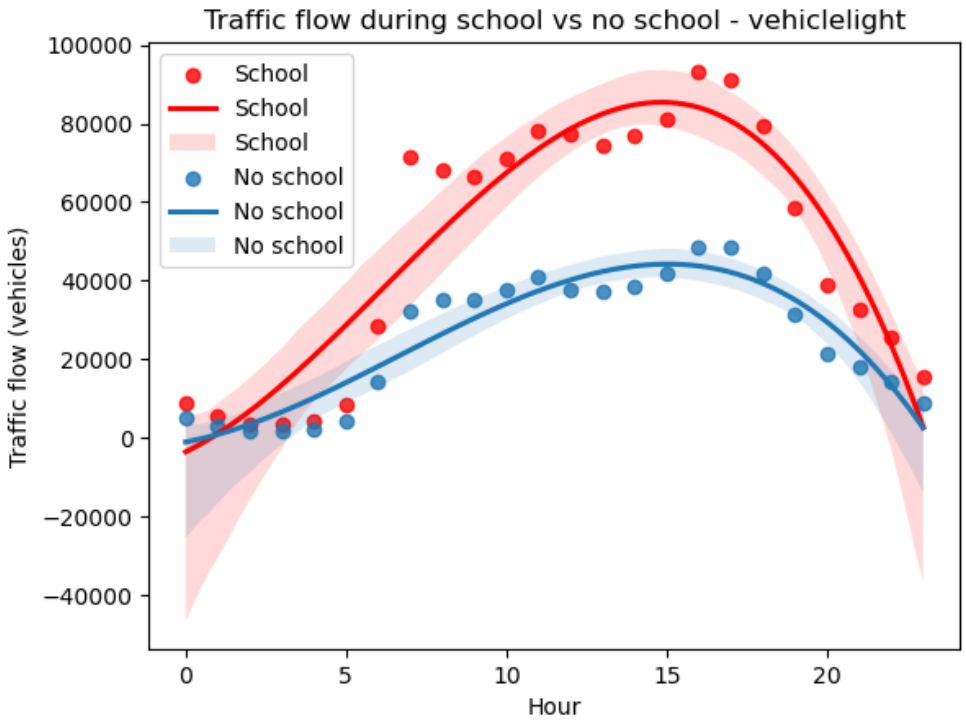

Fig. 25: Comparing Traffic volume of light vehicles, grouped by hour, on different types of days (school vs no school, regression).

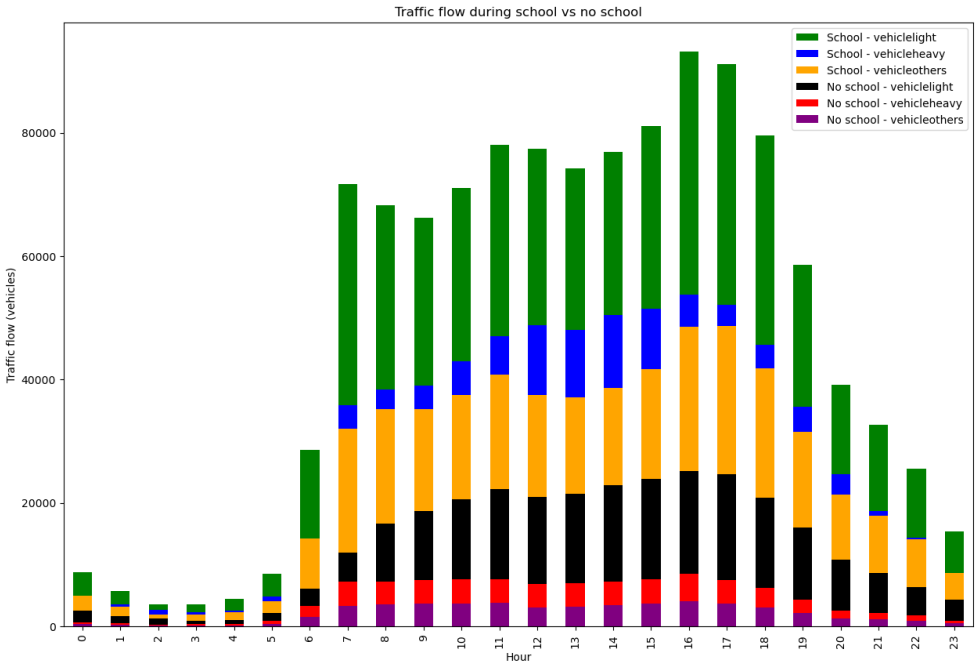

Fig. 26: Comparing Traffic volume, grouped by hour, on different types of days (school vs no school).

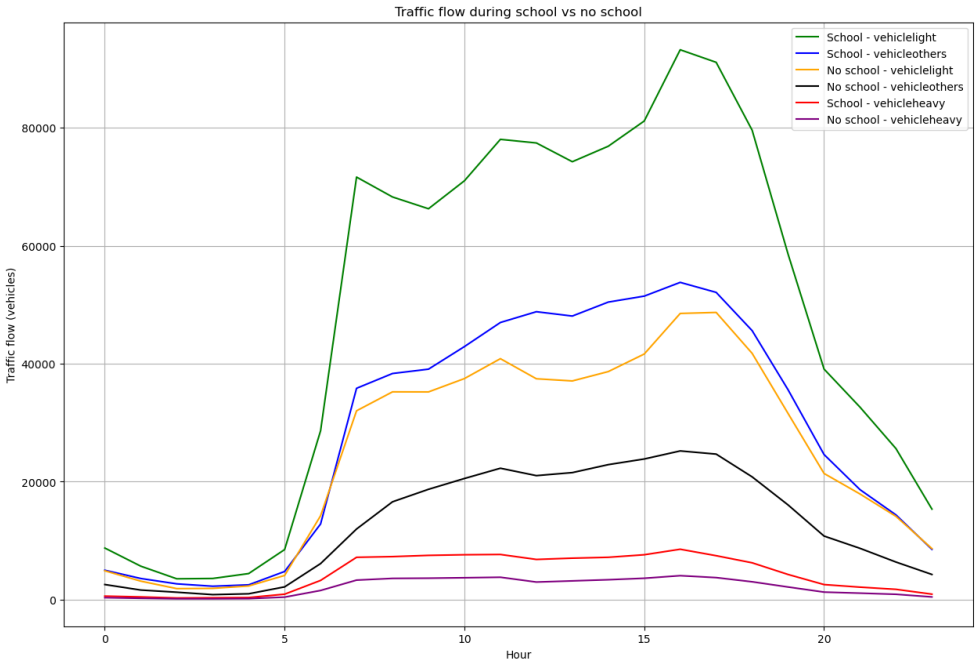

Fig. 27: Comparing Traffic volume, grouped by hour, on different types of days (school vs no school).

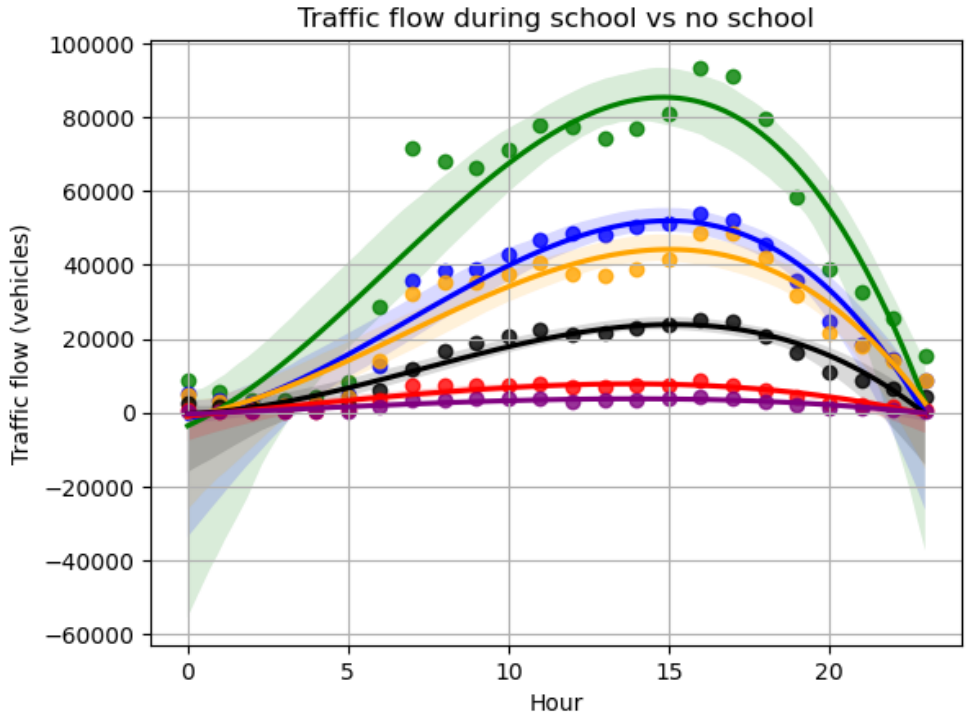

Fig. 28: Comparing different traffic volume, grouped by hour, on different types of days (school vs no school). Green/orange: light vehicles during school/no school; blue/black: other vehicles during school/no school; red/purple: heavy vehicles during school/no school.

Another crucial correlation to explore is between traffic flow and weather conditions. Adverse weather, such as rain or snow, can significantly impact traffic by affecting the visibility or the road conditions (slippery/wet). This kind of correlation allows transportation agencies to implement weather-responsive strategies, such as adjusting the traffic signals based on the current weather conditions, or deploying additional resources during bad weather, which would possibly improve the traffic flow and reduce accident risks.

6.2.1 Comparing good weather: school vs no school

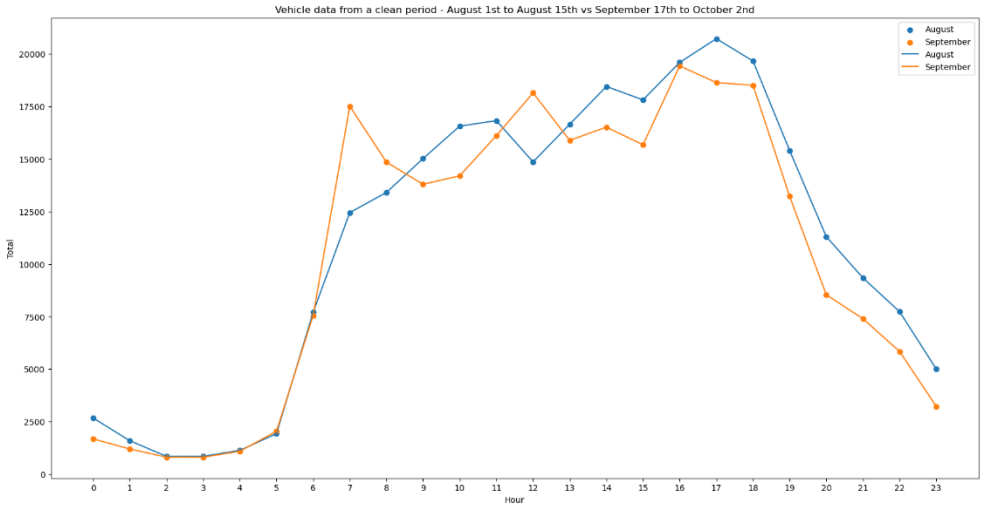

Fig. 29: Comparing traffic flow on favourable weather conditions during no school period (blue) and school period (orange).

The days used on the graph to represent the period with no school correspond to the first two weeks of August (starting at Monday) while to represent the period of school we choose the final weeks of September (day 17th corresponding to a Saturday) and the first couple of days of October. The graph above might not be the best to demonstrate the possible correlation between the no school vs the school period. Although, to use a somewhat significant sample size, those were the best days we could get from the dataset. It's also important to note that the August time period might not belong to a school period, but it corresponds to the peak of the summer, which can explain why the values are so close to one another.

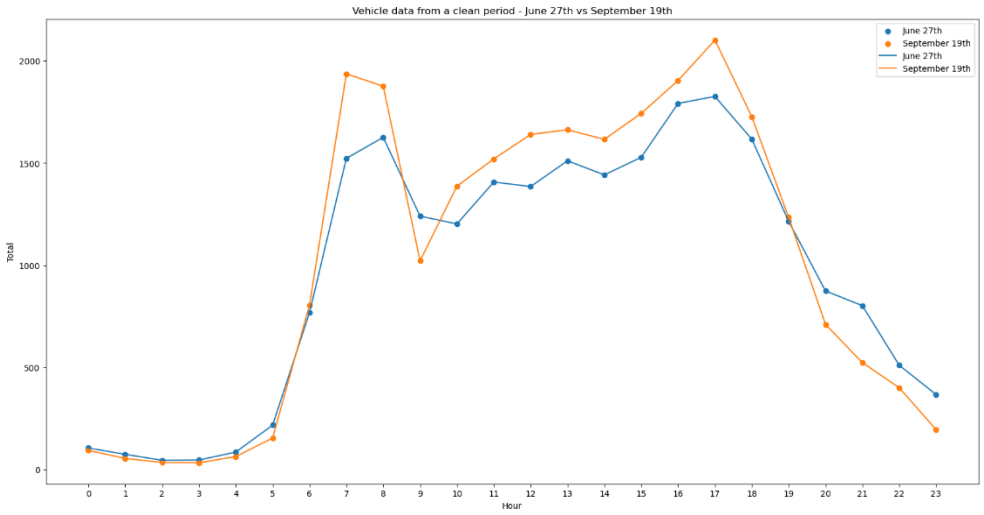

Fig. 30: Comparing traffic flow on favourable weather conditions during no school period (blue) and school period (orange).

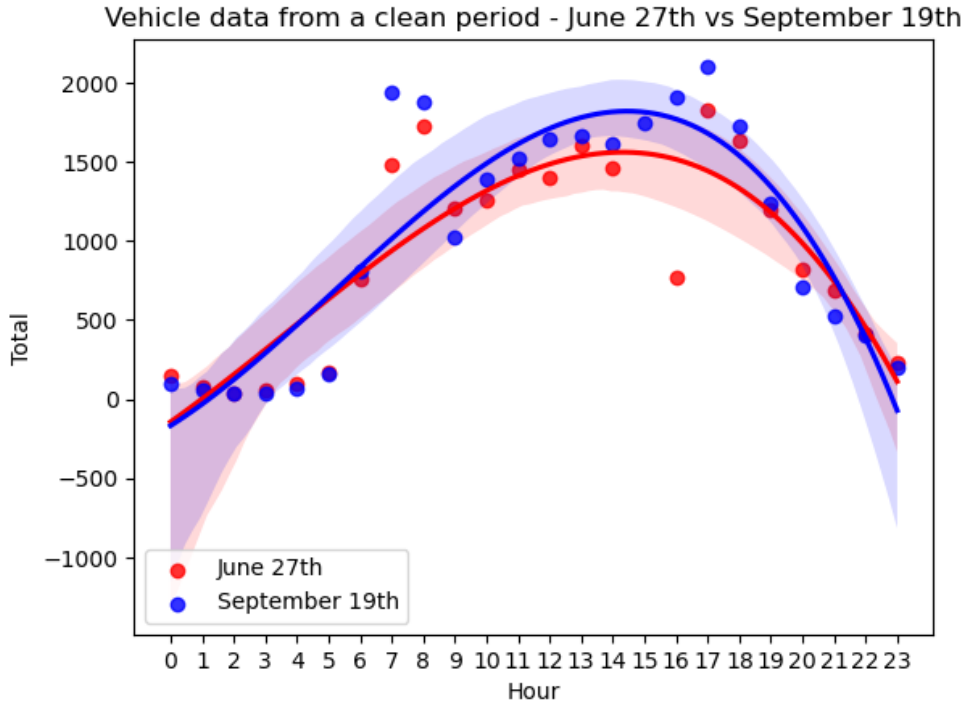

Fig. 31: Regression of traffic flow on favourable weather conditions, during school periods (blue) and during no school periods (red).

We should consider that the no school day chosen (27th of June) corresponds to peak summer, therefore it's understandable that the traffic flow is that high.

6.2.2 Comparing bad weather vs good weather during school

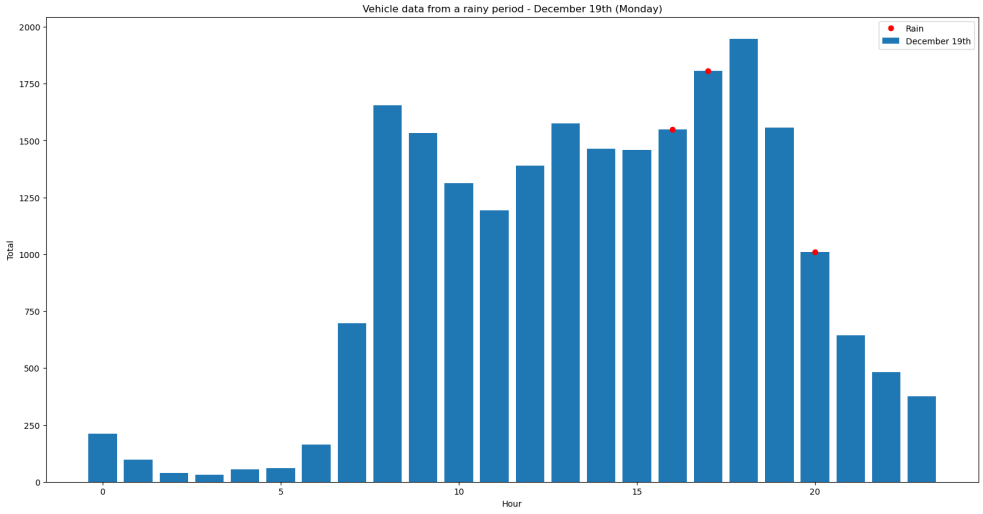

Fig. 32: Traffic flow on December 19th (rainy period, with raining hours marked with a red dot).

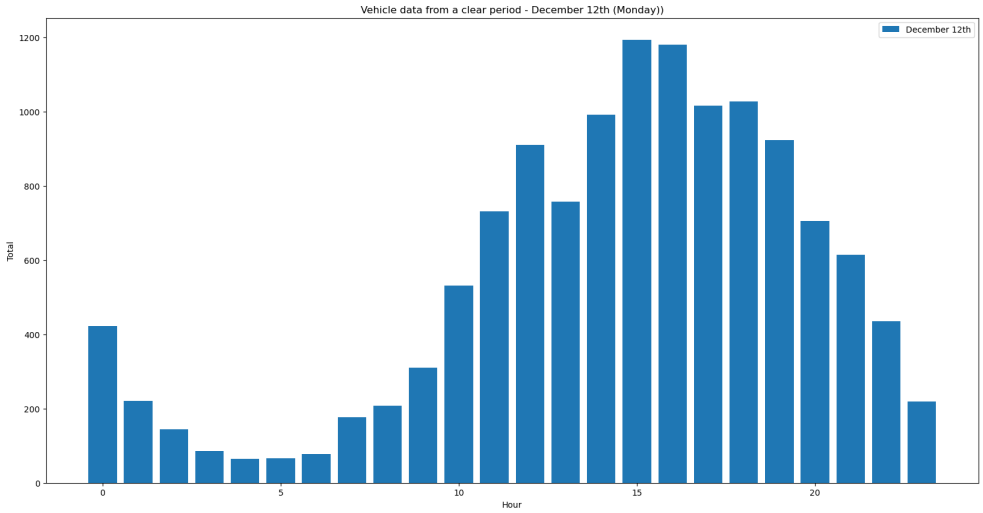

Fig. 33: Traffic flow on December 12th (clean period).

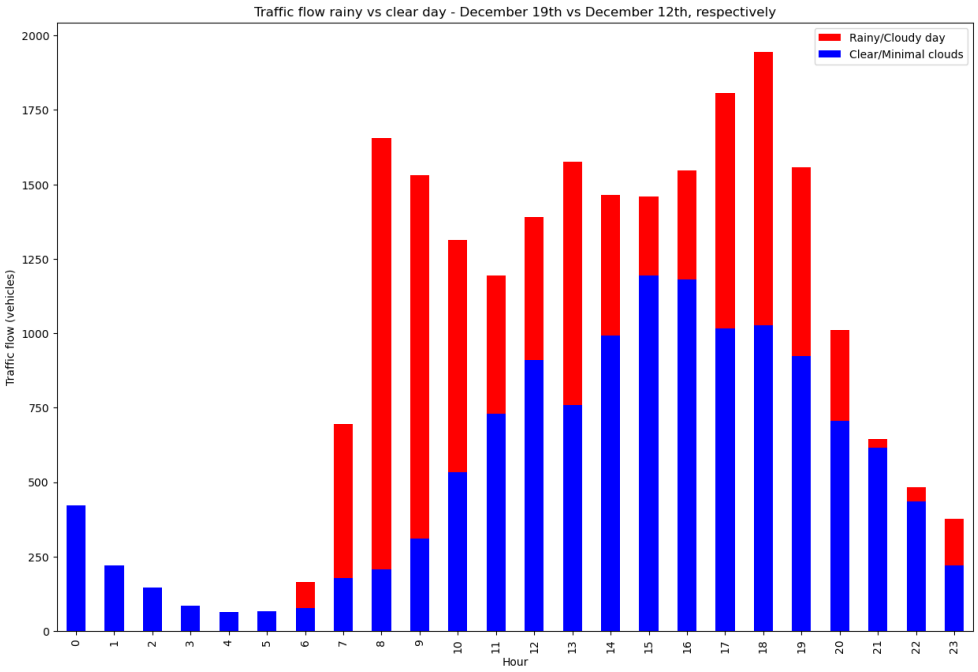

Fig. 34: Comparing the traffic flow on December 19th with December 12th. This graph can describe really well how the adverse weather influences the traffic flow.

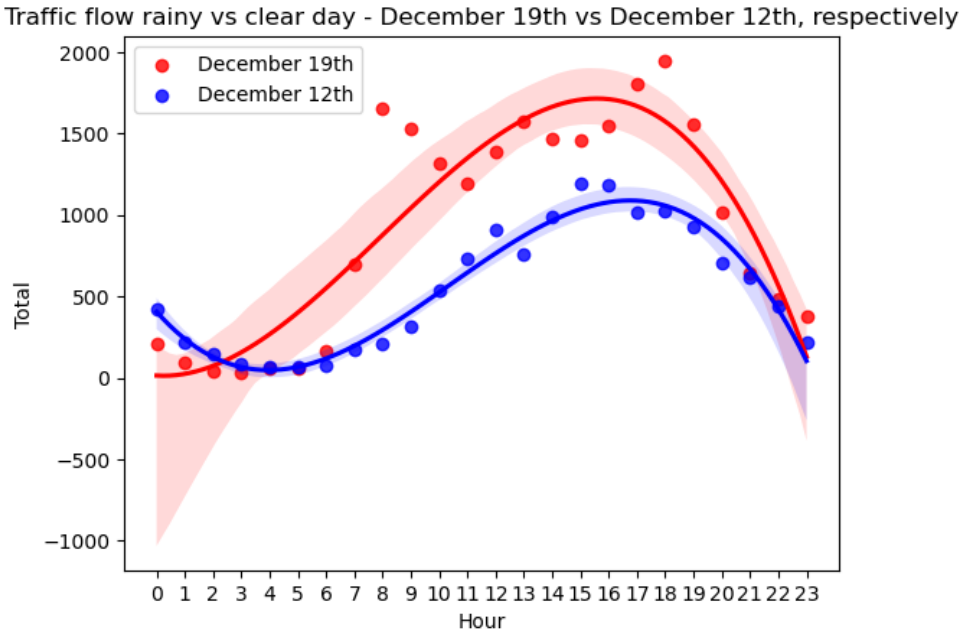

Fig. 35: Regression of traffic flow on adverse weather conditions (red) vs good weather (blue) during school periods.

6.2.3 Comparing bad weather vs good weather during no school

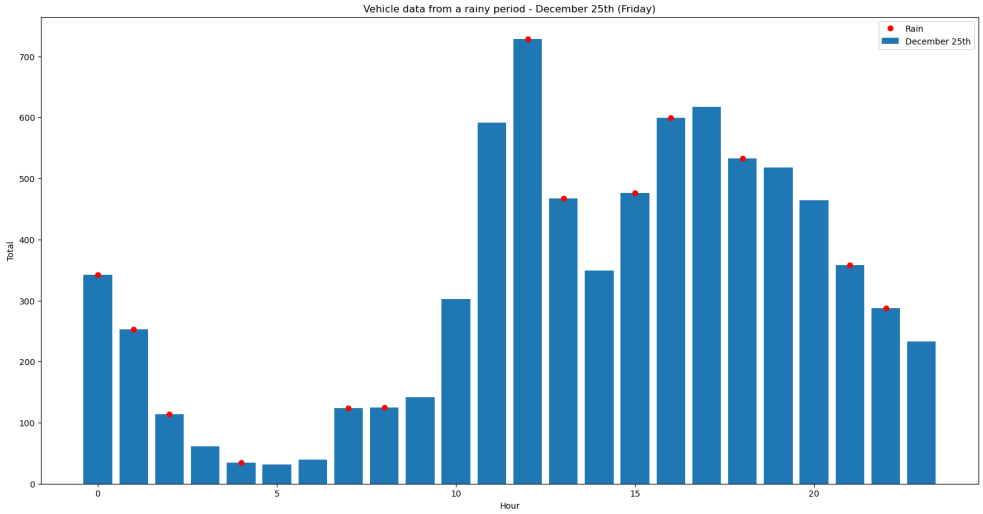

Fig. 36: Traffic flow during a rainy day (December 25th).

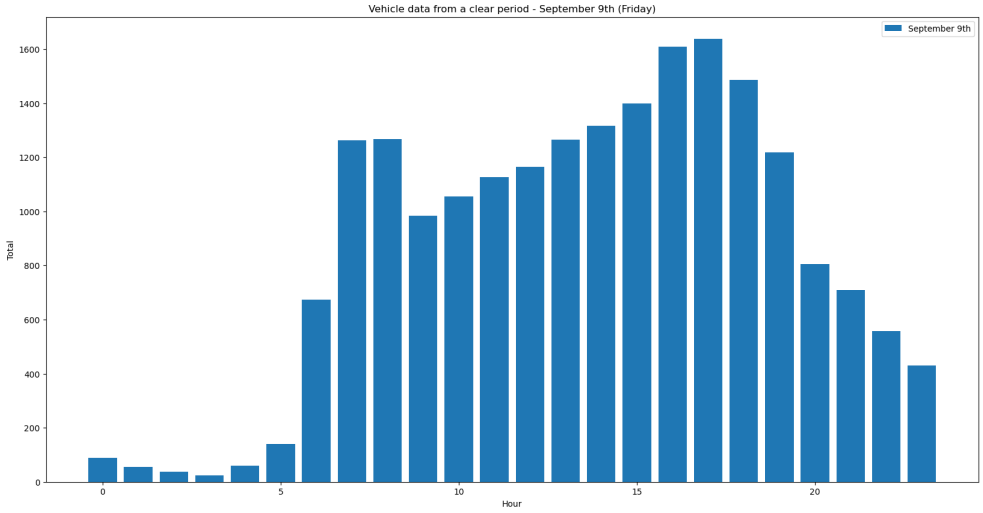

Fig. 37: Traffic flow during a clean period (September 9th).

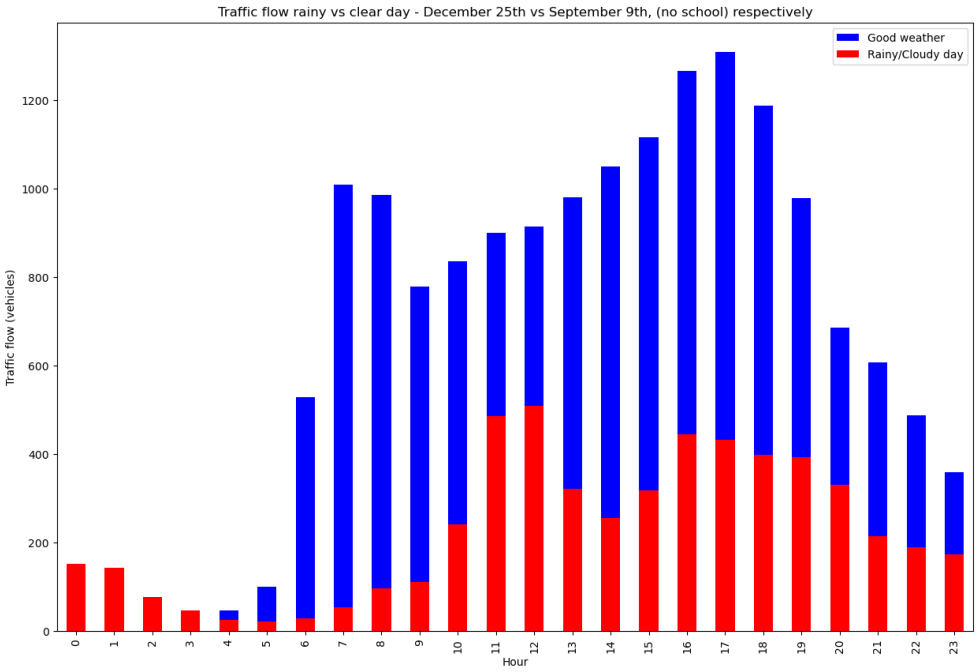

Fig. 38: Traffic flow comparison during no school period (clear weather (blue) vs adverse weather (red)). This graph may not be the most accurate representation as it was specifically chosen to showcase the best day in terms of rainfall, which happened to be Christmas day (December 25th).

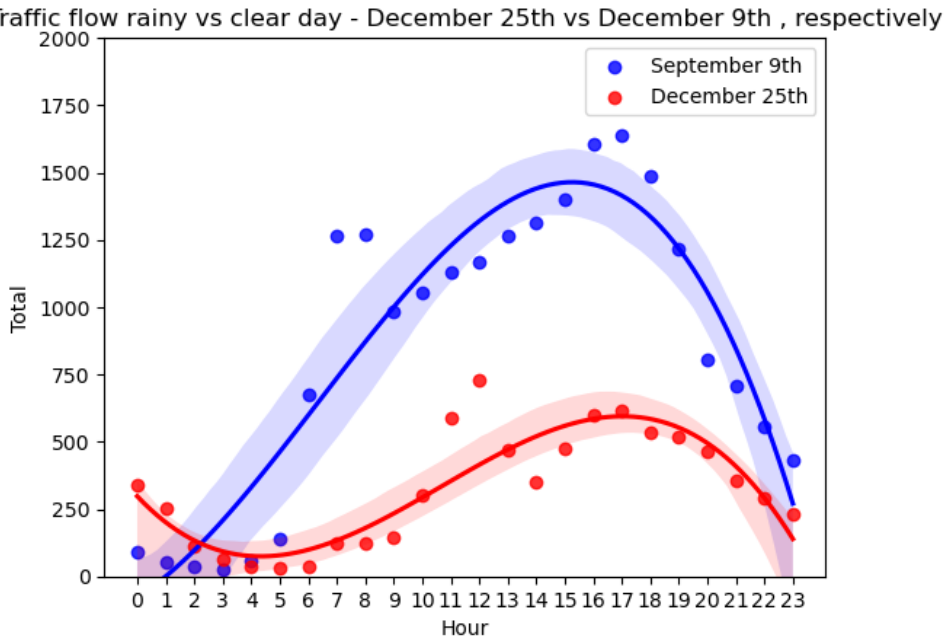

Fig. 39: Regression of traffic flow during no school period on adverse weather conditions (red) vs good weather (blue).

6.2.4 Comparing bad weather: school vs no school

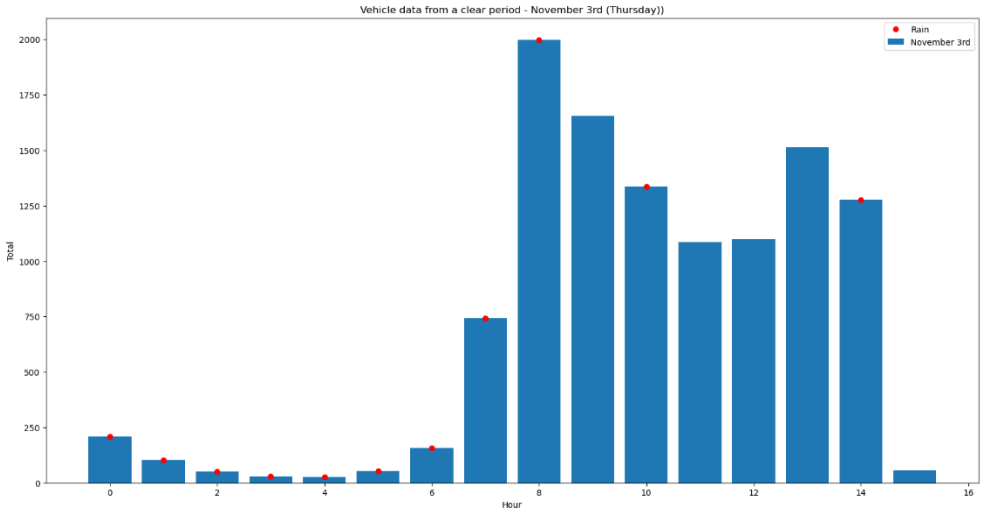

Fig. 40: Traffic flow on November 3rd (day with adverse weather).

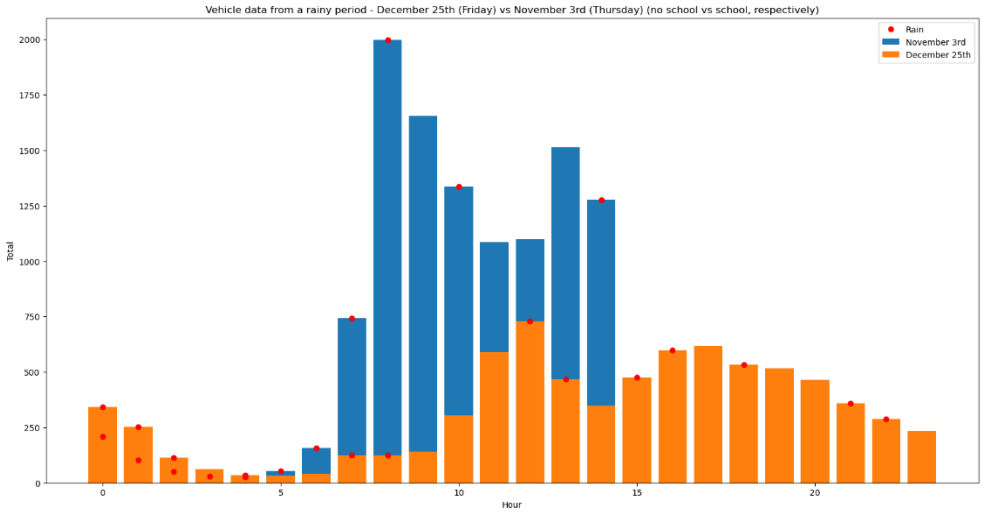

Fig. 41: Traffic flow comparison on adverse weather, school (blue) vs no school (orange).

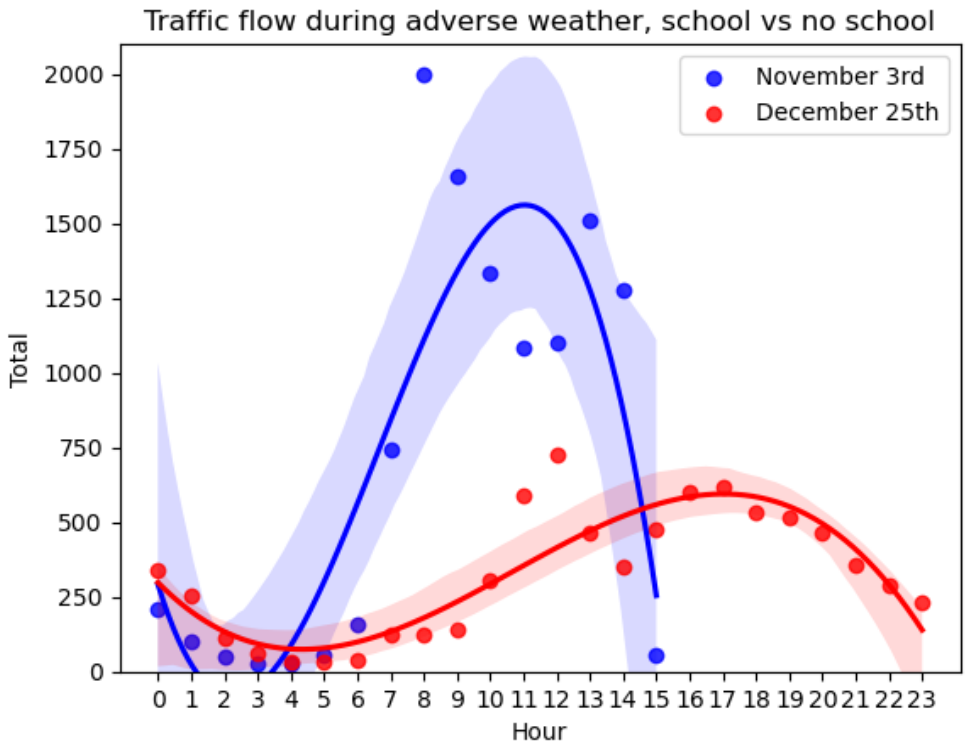

Fig. 42: Regression of traffic flow on adverse weather conditions during school (blue) and no school periods (red).

Another correlation we felt necessary to address was comparing weekdays with school vs no school and comparing weekend days under the same conditions. The data for the correlations, as before, were obtained through another couple of datasets provided by the mentors. To maintain a certain consistency the data utilised is from the radar 35 (ISCA). For the "school" days it was used data from the full month of March 2023 and for the "no school" days was used the month of July 2023. But there is a problem with the dataset that contains the July data. Probably the radar had technical problems and was not functional for some period, which could lead to some less correct correlations.

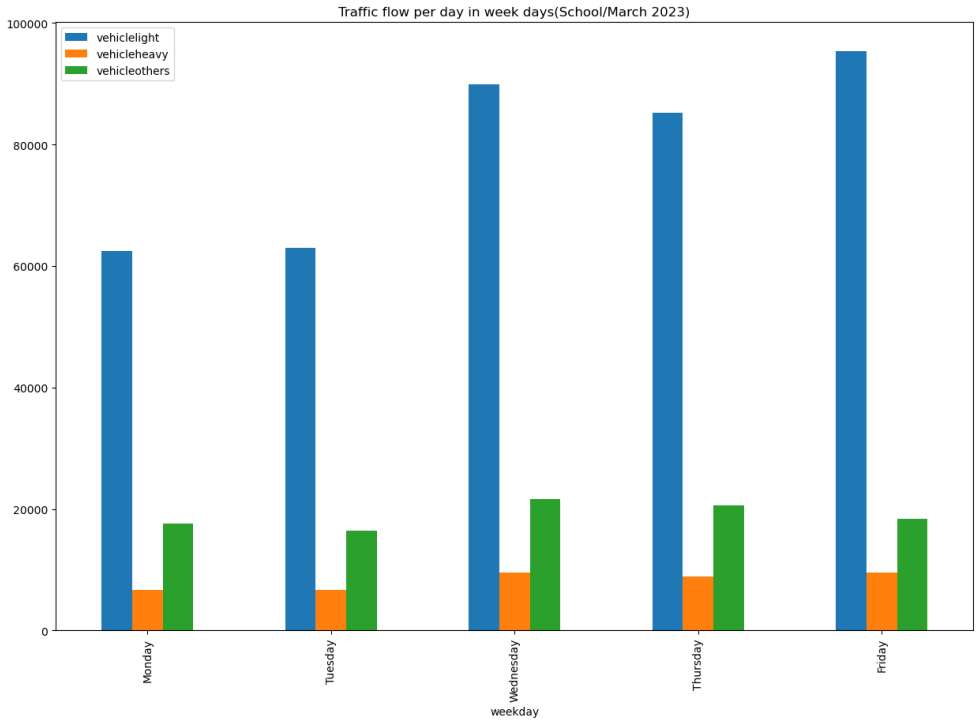

Fig. 43: Sum of traffic volume by weekday, month of March (during school period).

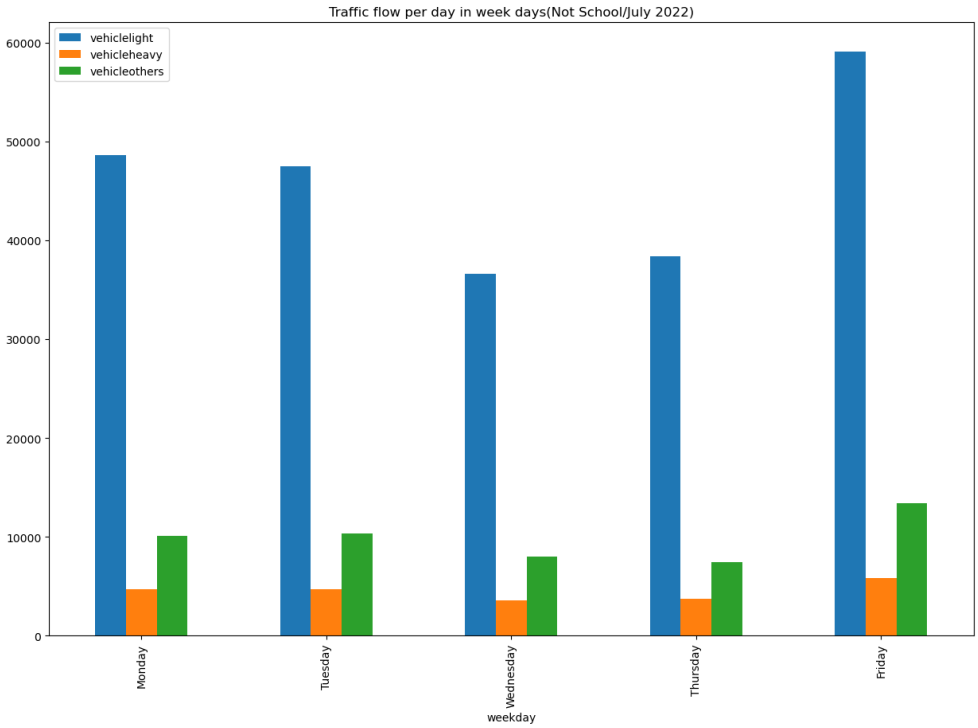

Fig. 44: Sum of traffic volume by weekday, month of July (during "no school" period).

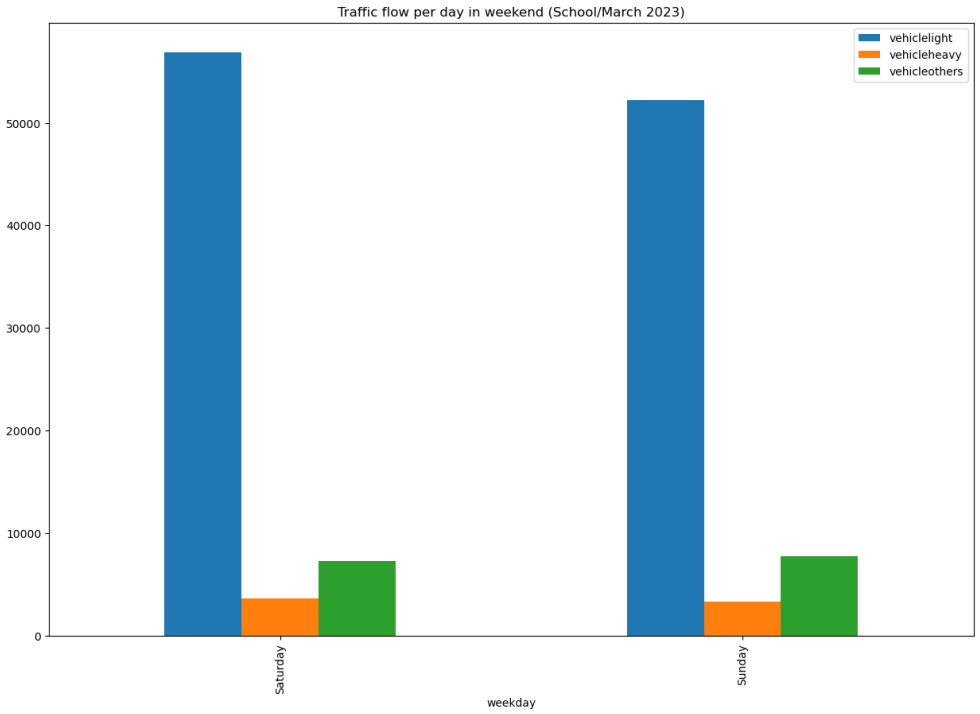

Fig. 45: Sum of traffic volume by weekend day, month of March (during "school" period).

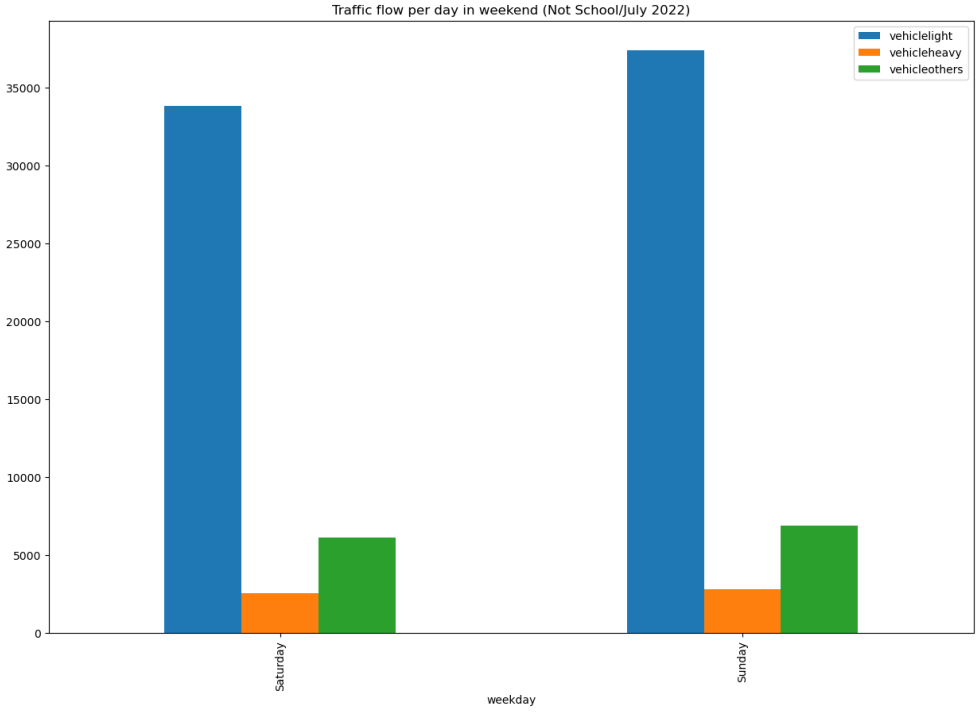

Fig. 46: Sum of traffic volume by weekend day, month of July (during "no school" period).

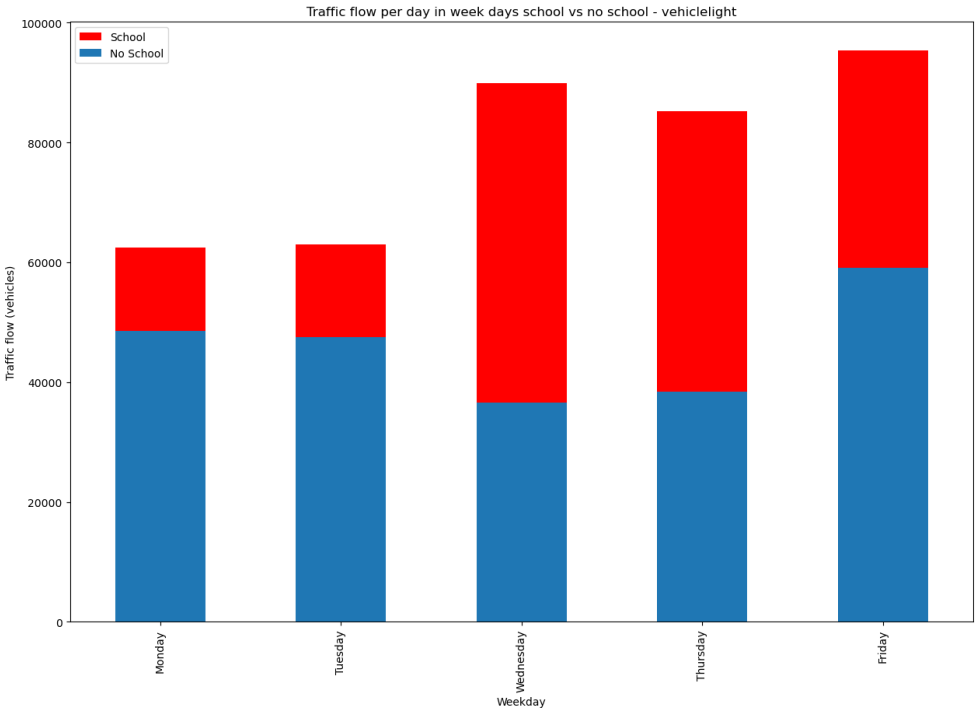

Fig. 47: Comparing Traffic Volume of light vehicles, aggregated by weekday, on different types of days (school vs no school).

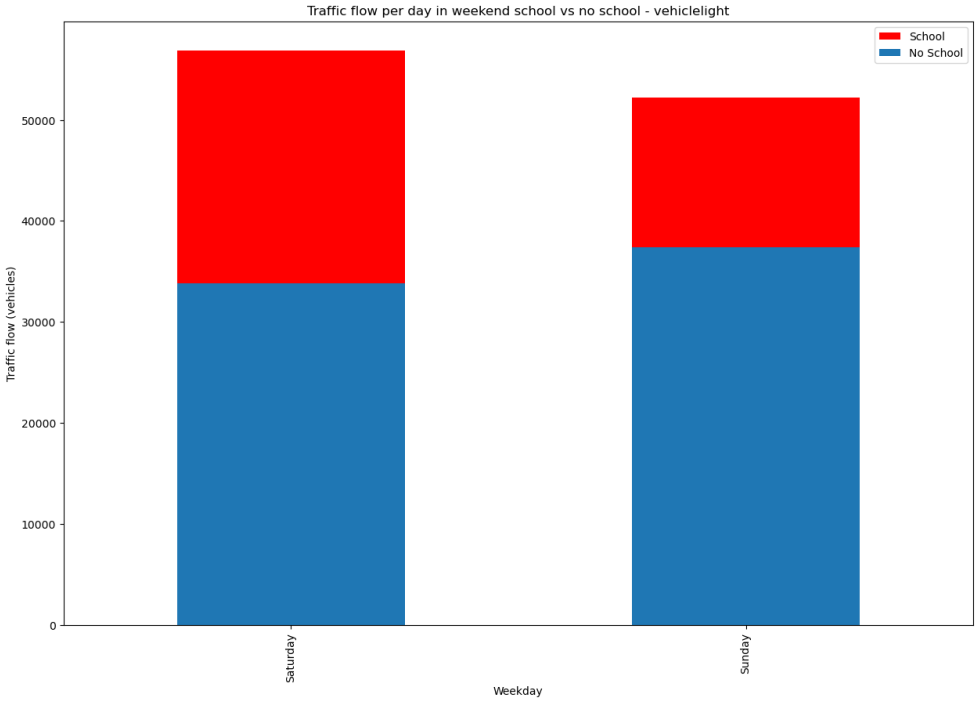

Fig. 48: Comparing Traffic Volume of light vehicles, aggregated by weekend day, on different types of days (school vs no school).

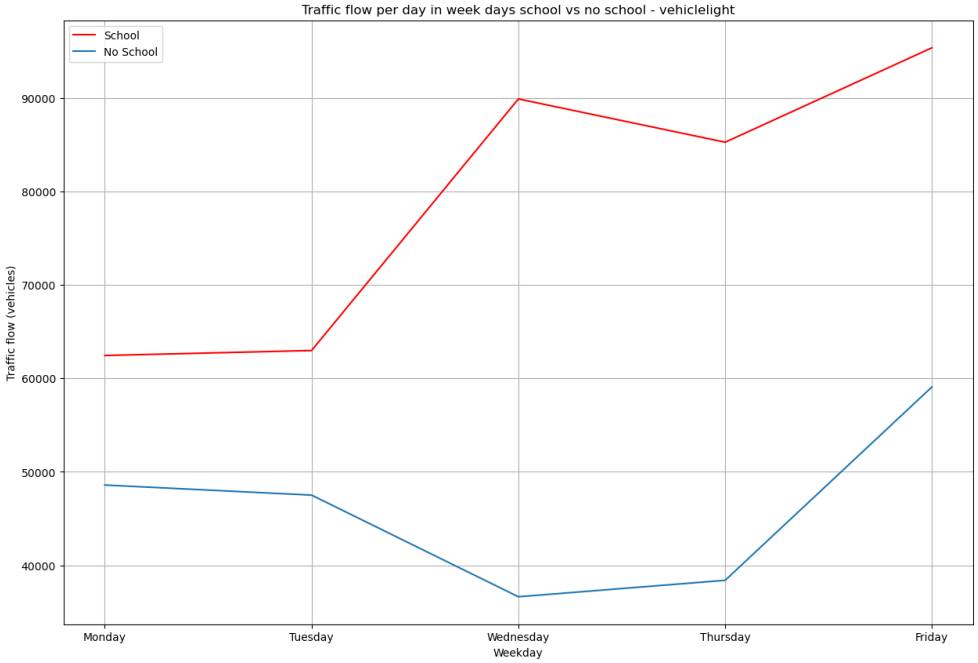

Fig. 49: Comparing Traffic Volume of light vehicles, aggregated by weekday, on different types of days (school vs no school).

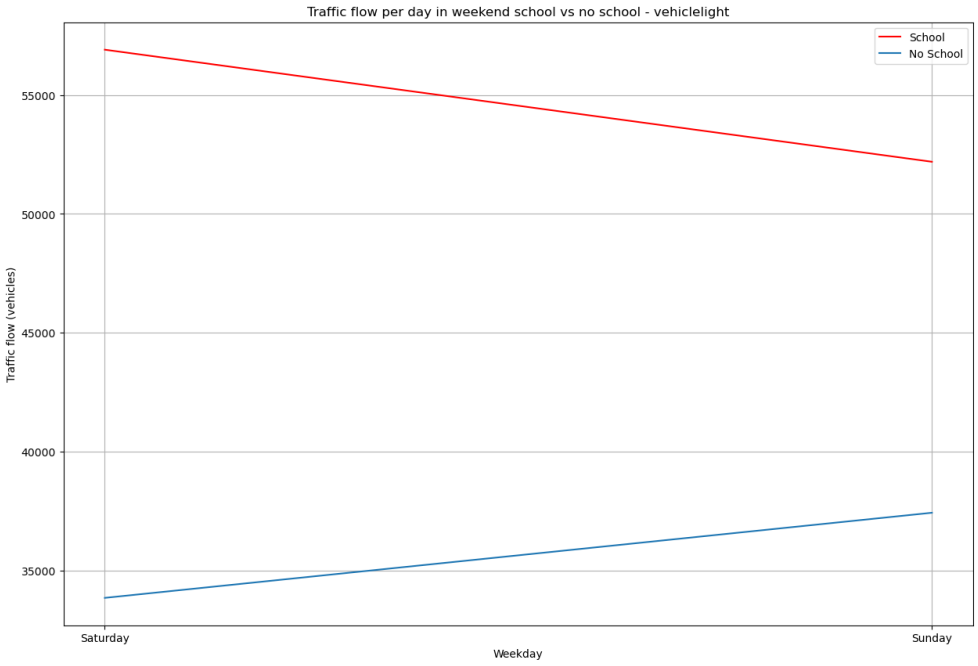

Fig. 50: Comparing Traffic Volume of light vehicles, aggregated by weekend day, on different types of days (school vs no school).

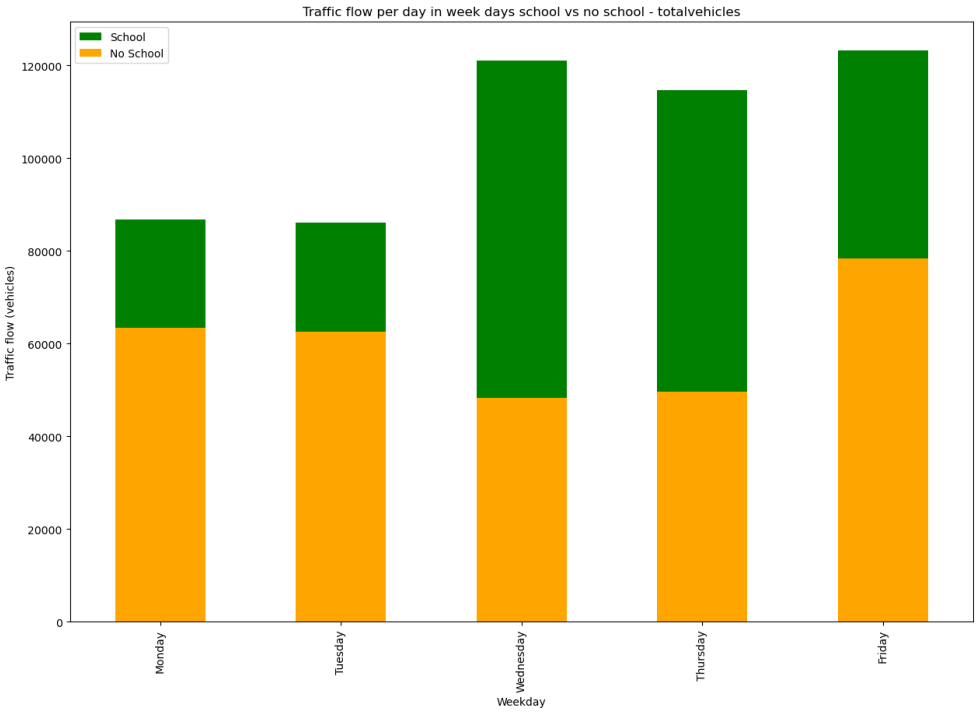

Fig. 51: Comparing Traffic Volume of all vehicles, aggregated by weekday, on different types of days (school vs no school).

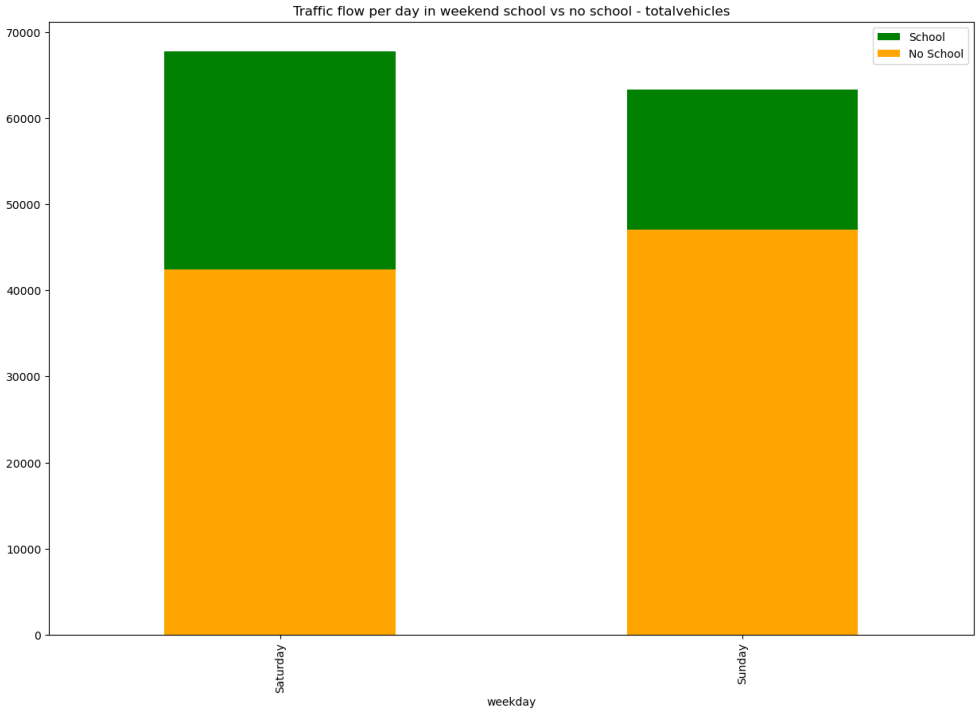

Fig. 52: Comparing Traffic Volume of all vehicles, aggregated by weekend day, on different types of days (school vs no school).

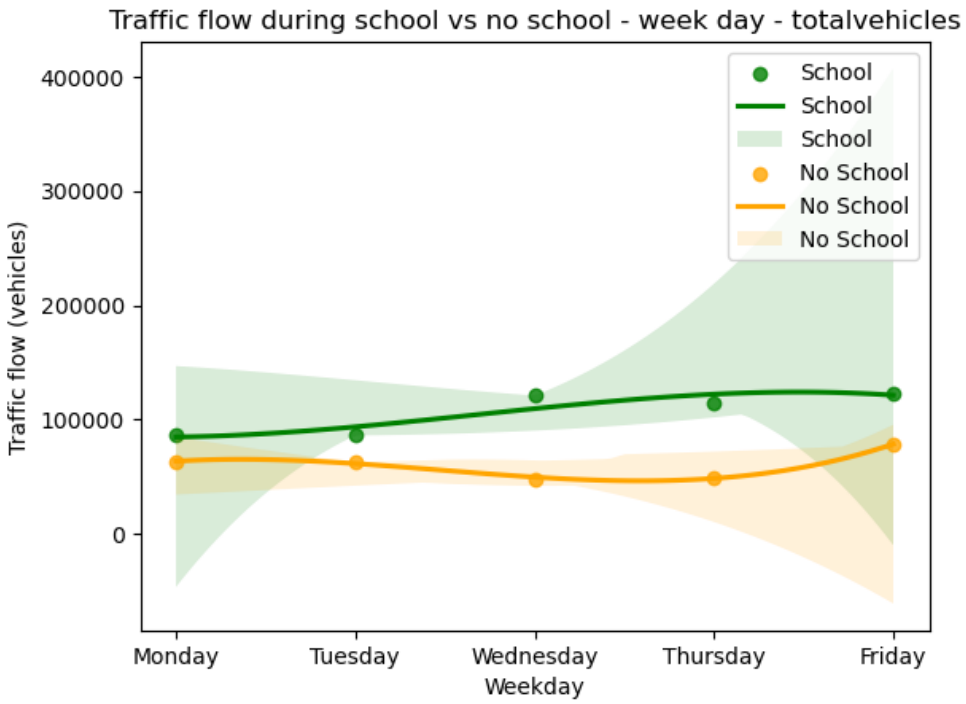

Fig. 53: Comparing Traffic Volume of all vehicles, aggregated by weekday, on different types of days (school vs no school) (regression).

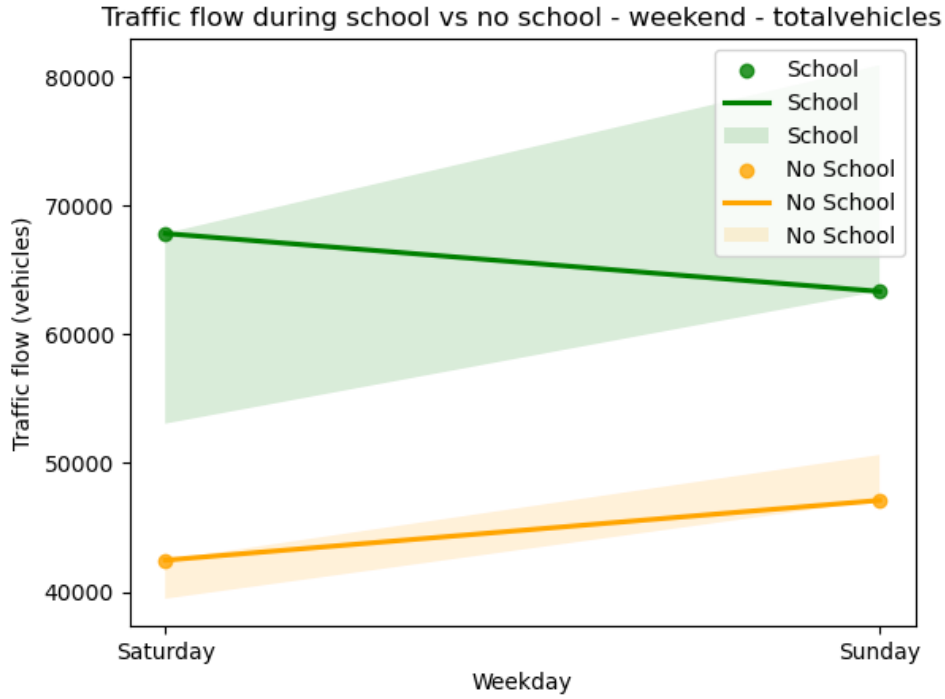

Fig. 54: Comparing Traffic Volume of all vehicles, aggregated by weekend day, on different types of days (school vs no school) (regression).

This last graph is the perfect example of what we mentioned in the beginning, the lack of values causes the regression to not make much sense. If we look at the "school" period section, we can see the tendency going upwards but the line going downwards. That happens because of some missing values but also because maybe regression isn't the most appropriate method of analysis to be applied here since there are only two points to represent in the graph.

Understanding these correlations also facilitates the development of predictive traffic models. With the integration of data on traffic flow, weather conditions and days of the year these models can provide accurate forecasts of traffic congestion. People can then plan their journeys better by choosing alternative routes or travel hours, to avoid traffic congestion.

In conclusion, analysing correlations on traffic flow, days of the year and weather conditions is crucial for an effective traffic management, transportations planning and road safety. By identifying these relationships, authorities can conduct targeted interventions, enhance the travel experience and guarantee effective vehicle circulation on the roadways. These correlations improve the accuracy of the predictive models, which in return should improve the traffic flow management.

As the API from ATCLL has a few problems, all the data used below and above was provided by the mentors. The data starts at January 12th and goes until 25th May of 2023. The data utilized comes exclusively from the radar 35 – ISCA, which exhibits the highest traffic volume, particularly during school periods.

6.2.5 Traffic Flow Prediction: ARIMA vs LSTM

In our approach we used both statistical (*ARIMA*) and Deep Learning (*LSTM*) methods to find the most suitable combination of models for accurate predictions. Before getting into detail on both of those approaches, lets analyse the data to try and understand some of its characteristics and underlying patterns.

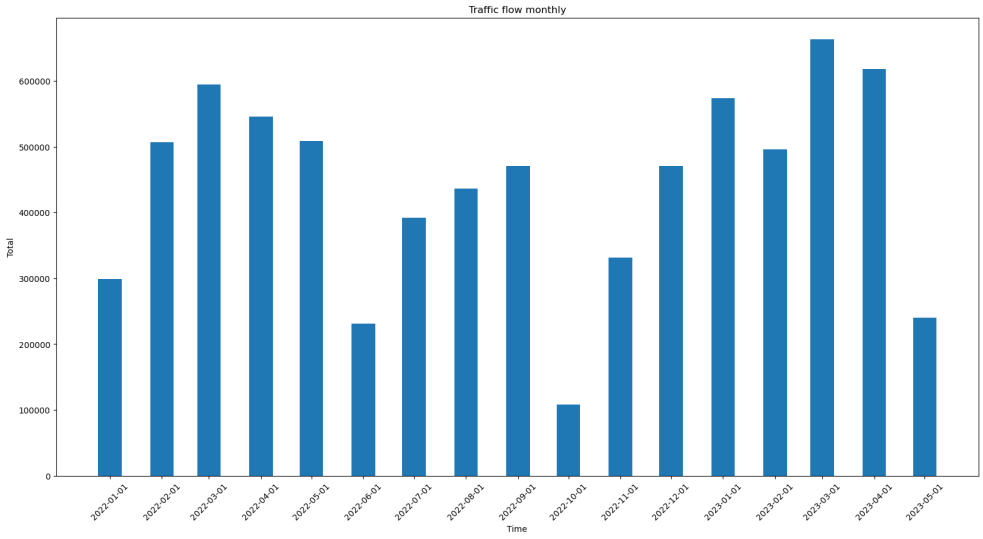

Fig. 55: Plot of the traffic volume per month from January (11th) 2022 until May (25th) 2023.

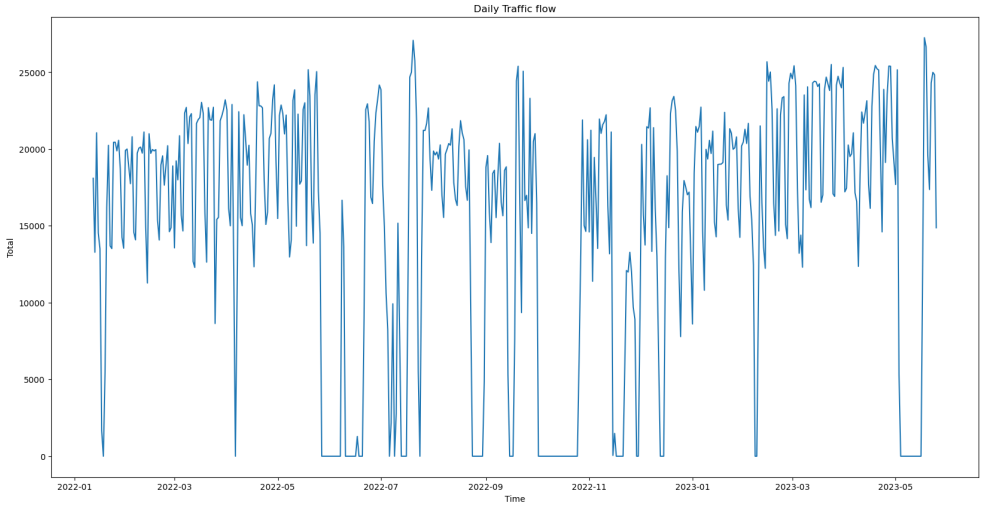

Fig. 56: Plot of the daily traffic volume.

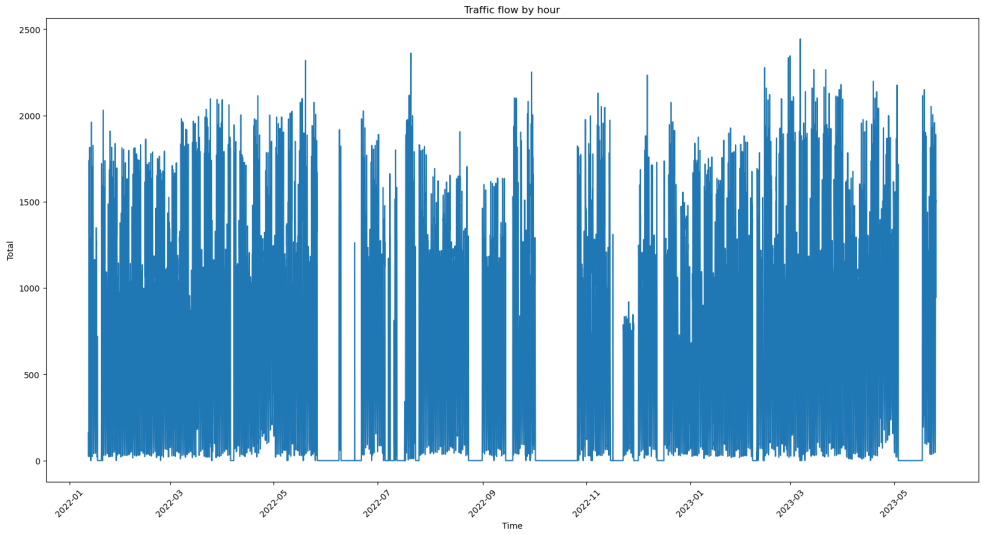

Fig. 57: Plot of the hourly traffic flow during 2022.

From images these images, we can see that there are some missing values. To solve this problem, we opted to use *IterativeImputer* from the *fancyimpute* library. In order to do so, we substituted all values that were equal to zero by *nan,* and then we use the *InterativeImputer.* We consider that this is a relatively good approach, as the actual number of zero values should be close to null, it's odd that during an hour not a single car goes by the radar-35 at ISCA.

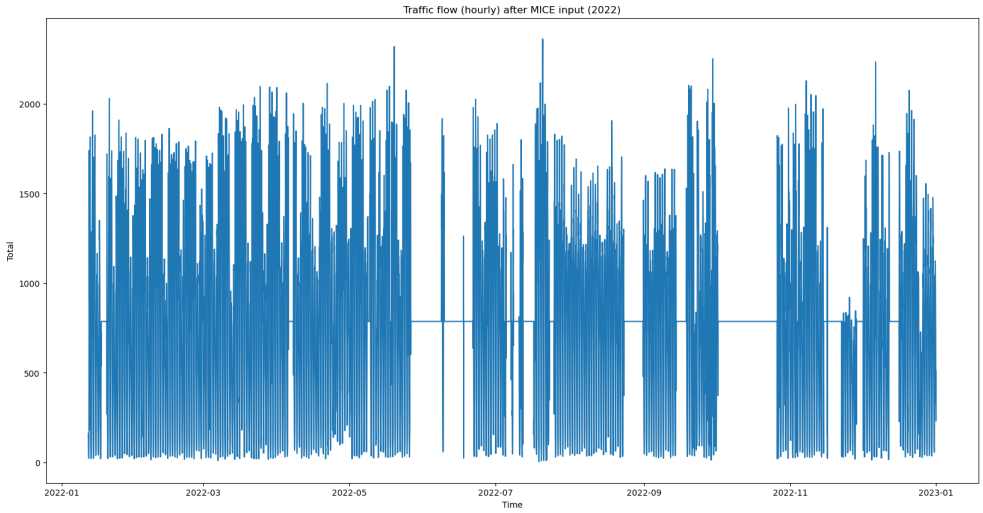

Fig. 58: Plot of the hourly traffic flow, after values inputting.

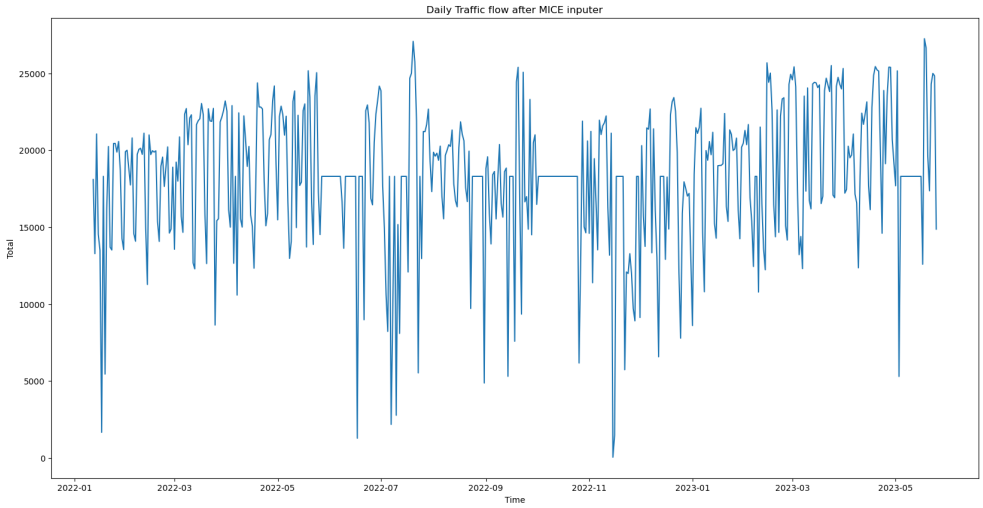

Fig. 59: Plot of the daily traffic volume, after values inputting.

From comparison between figures 58 and 57, and figures 56 and 59, we can see a clear difference during June and October, especially. Its better (for our model) to have some values, even if not completely accurate than to have values equal to zero.

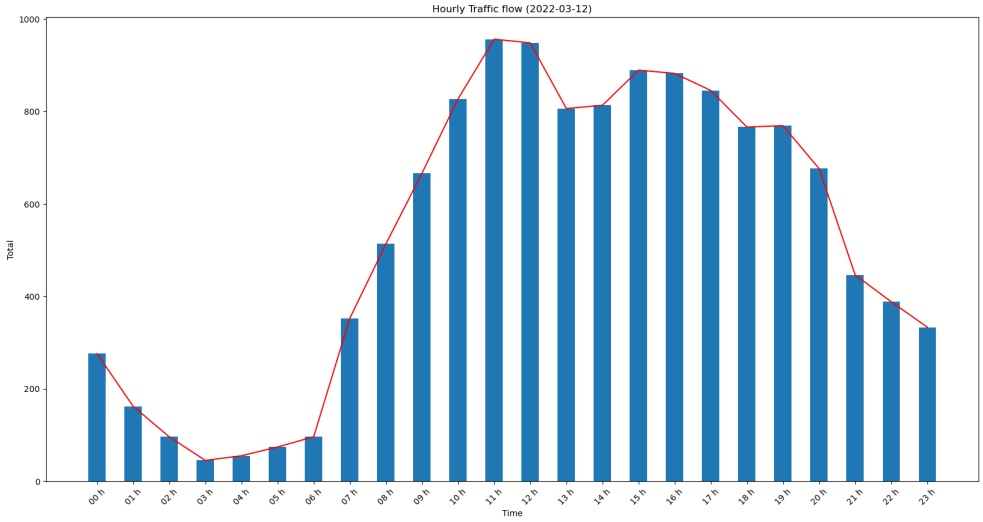

Fig. 60: Plot of the hourly traffic flow on March 12th, with the tendency line.

Following the data analysis, let's discuss the statistical approach using *ARIMA*. We based our knowledge on the TAA classes and on some of the references mentioned below. *ARIMA* is a statistical modelling technique used to analyze and forecast time series data. *AutoRegressive Integrated Moving Average*, also known as *ARIMA*, belongs to a class of models that combine the concepts of autoregression (AR), integration (I), and moving averages (MA) to capture the underlying patterns and dynamics within the data. *Autoregression* concentrates on the relationship between an observation and its lagged values, while the integration involves differencing the series, to achieve stationarity. And the moving averages consider the dependency between an observation and the residual errors from a moving average model.

In order to accurately predict a variety of time-dependent phenomena, ARIMA models' three key parameters: lag order (p), degree of differencing (d), and order of the moving average (q); These parameters must be carefully chosen for an accurate prediction.

We trained our ARIMA model using:

- p = 2
- d = 0
- q = 3

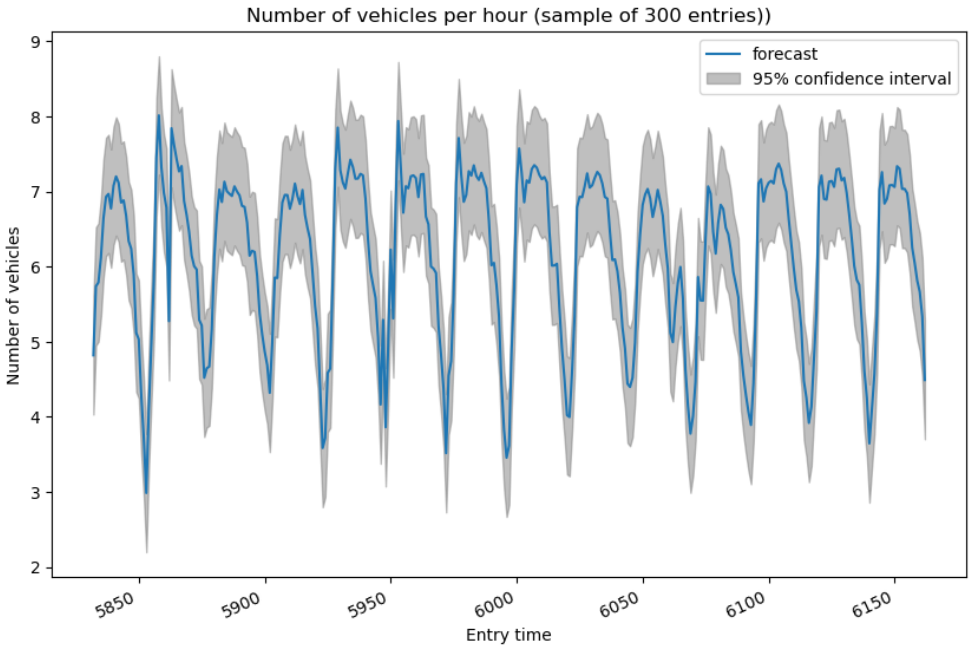

Fig. 61: Plot of predictions interval, of the last 300 data entries.

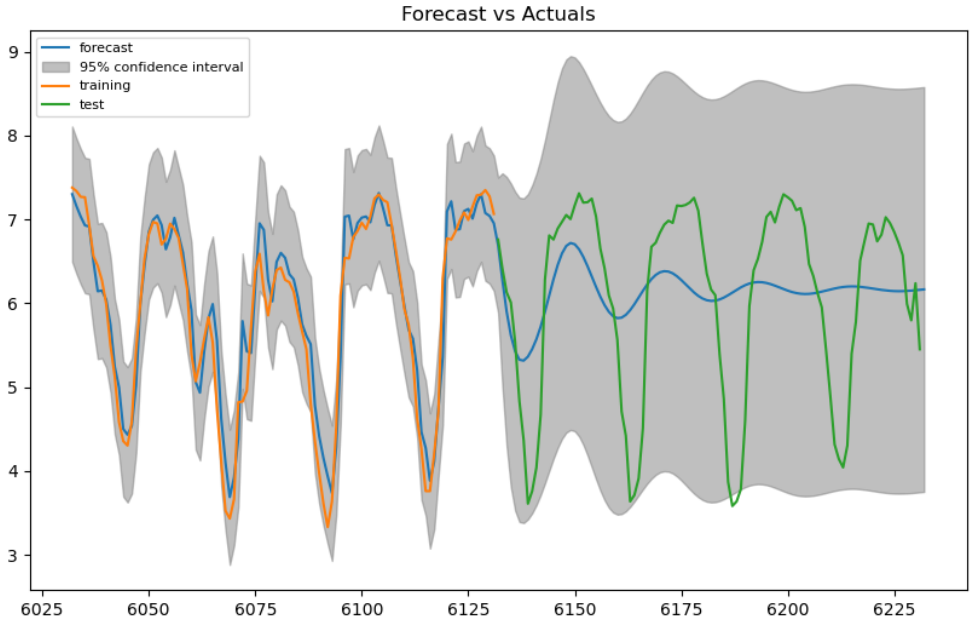

Fig. 62: Plot of the forecasted values in comparison with the actual values.

From figure 61, we can conclude that, as expected, the model can forecast the training data way better than it can forecast the testing data, just because the training data was the same data that we fed the model upon its training. On the actual forecasting, we can conclude that the model is somewhat good in the short term, maybe two days at most.

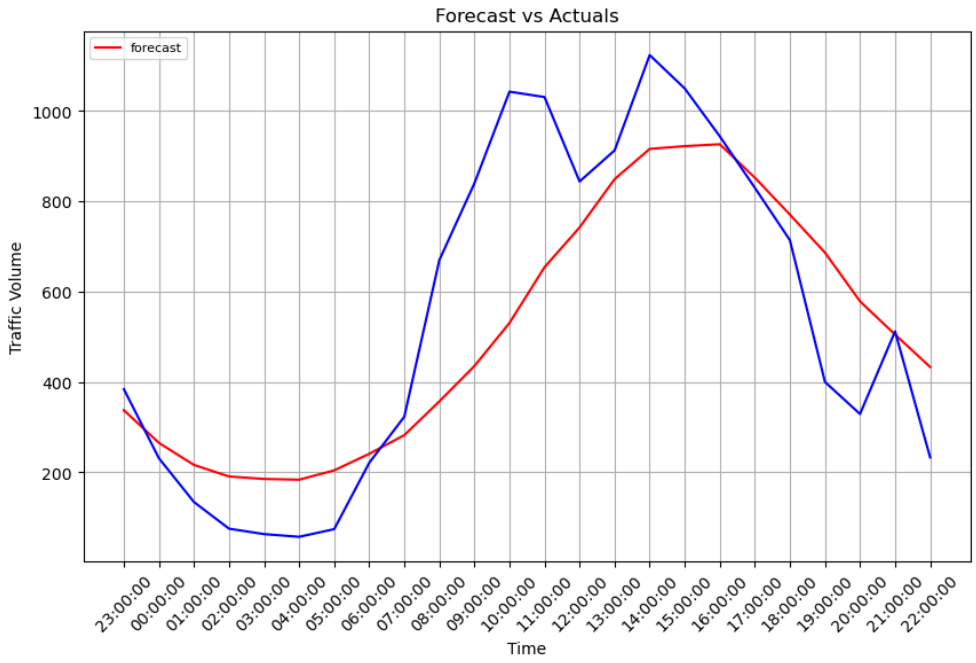

Fig. 62: Plot of the forecasted values in comparison with the actual values of the first day after the training data.

After shaping the statistical approach and realizing that it was not going to be sufficient, we further explored a Deep Learning approach, focusing on the LSTM (Long Short-Term Memory) model. To address this DL approach, we based our knowledge on the TAA classes and on some of the references mentioned below.

LSTM is a type of recurrent neural network (RNN) designed to model and learn from sequential data. LSTM introduces a memory cell that allows the network to capture long-range dependencies and mitigating the vanishing gradient problem. This makes it well-suited for our project. At the core of an LSTM unit is a memory cell that maintains and upgrates its internal state over time. The cell consists of three main components (input gate, forget gate, and an output gate) that control the flow of information (into, out of, and within the cell, respectively), enabling the network to selectively remember or forget information.

To develop our LSTM-based forecasting model we used the "Keras" library. We opted to use 2 LSTM layers with 64 units each, but where the first one as the *return_sequences=True* parameter specified which means that we ensure that the layer return the complete sequence of outputs rather than only the final output.

Finally, we introduce a fully connected dense layer with eight units and apply the Rectified Linear Unit (*ReLU*) activation function. This layer serves to further transform the LSTM outputs and map them to the desired output space. After the models been built, we compile it using the mean squared error (MSE) loss function and the Adam optimizer. The MSE (*Mean Squared Error*)

evaluates the difference between the observed and the predicted values. If the predicted value and the observed value are equal, the error is zero. A good model should have a MSE close to zero. The metrics used to evaluate the performance of the forecasting model on the training data are the previously discussed Mean Squared Error (MSE) and Mean Absolute Error (MAE). MAE computes the average of the absolute differences between the predicted values and the actual values. A smaller MAE indicates that, on average, the model's predictions have less absolute difference from the actual values.

We trained the model using the two LSTM layers, along with another dense layer, we then fitted the model over 30 epochs, which led us to a loss variation shown in the graph below.

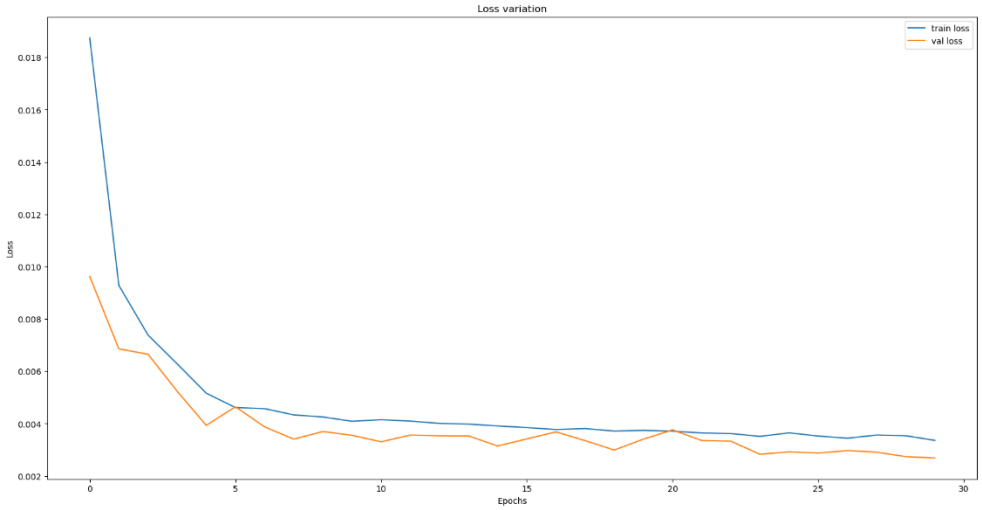

Fig. 63: Plot of Loss variation along the 30 epochs.

We tested our model on the training data, obtaining the following results:

- Train Loss: 0.0031
- Train MSE: 0.0031
- Train MAE: 0.0312

We decomposed our data, so we could have a closer look at the trend, seasonality and the residuals.

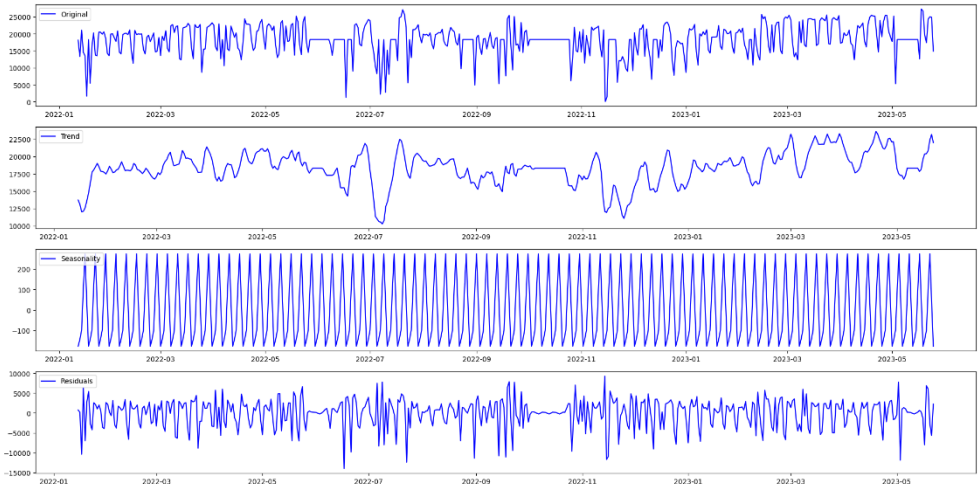

Fig. 64: Time series decomposition (daily data).

From the plots above, we can see that the missing values have a direct impact on the other plots. We can't really see a trend here, but that doesn't mean that there are no patterns in the data. Although, we can infer that there are probably some patterns, most likely cyclical ones, repeating every year (12 months).

Then we evaluated our data to see if our suspicions about it being stationary was correct, using the ADF test. The Augmented Dickey-Fuller (ADF) test is a statistical test for assessing the stationarity of a time series. The ADF test measures the strength of a time series' ability to be characterized by a trend. If the time-series is non-stationary, we need to apply differentiation to make it stationary. Testing the model over the testing data:

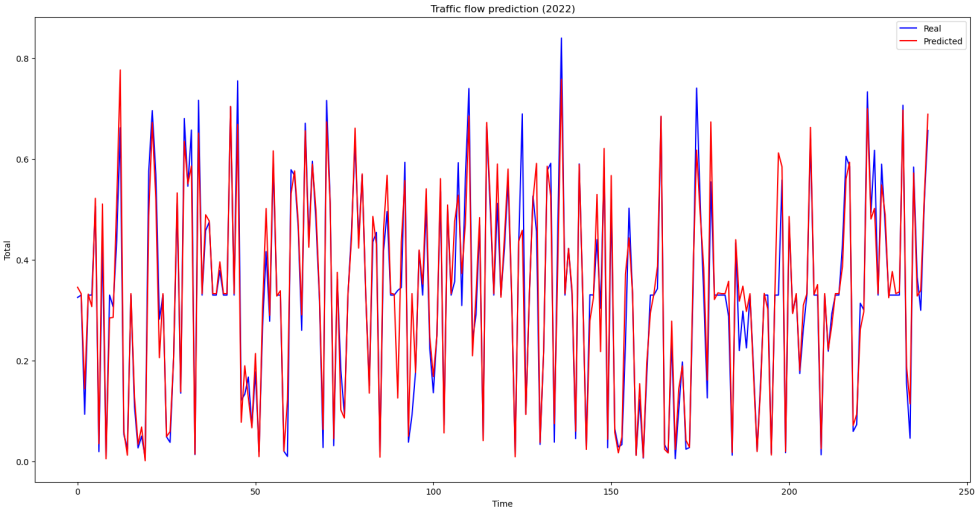

Fig. 65: Predictions vs Actual values.

Overall, we can see that our model adapts well as the lines are very close to one another, which means that the predicted values are not far apart from the actual ones.

To be able to make predictions, we decided to use the last 360 entries (corresponding to 15 days) as the input to the model. Which means that to calculate the next value, the model will consider the previous 360 entries. Predicting the next four days:

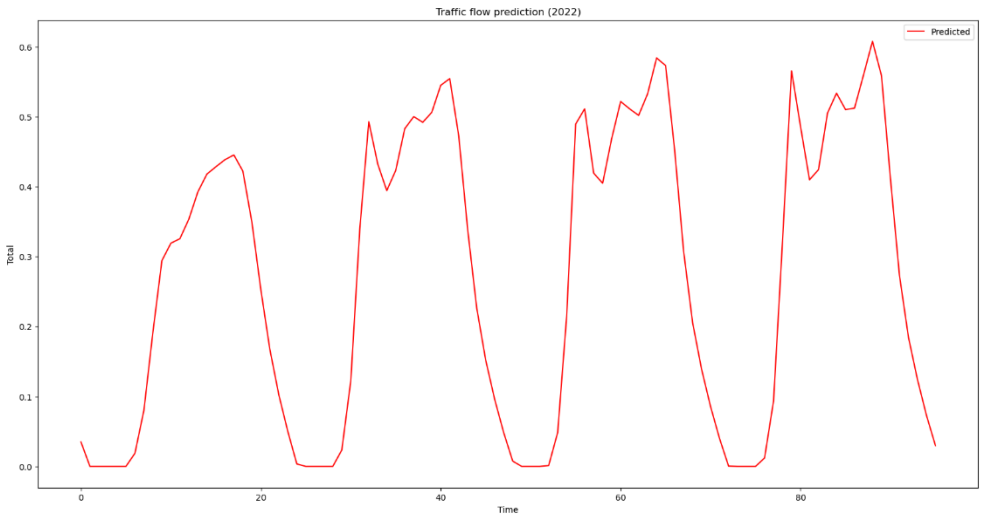

Fig. 66: Four days forecast.

Selected the third day to a more profound analysis.

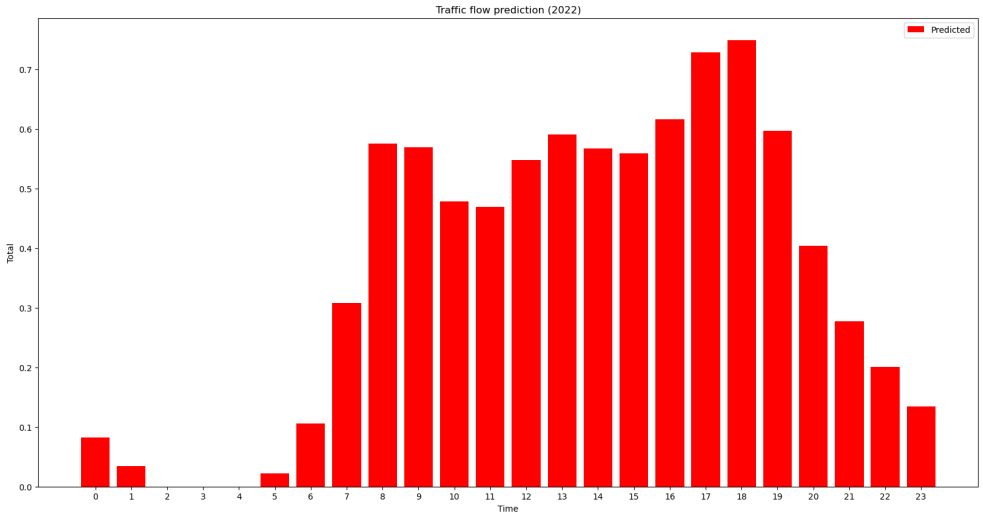

Fig. 67: Daily Forecast (3rd day from figure 66).

6.2.6 Results Discussion

In summary, the models achieved promising results, particularly in terms of hourly predictions, which were the primary objective of the project. We concluded that the ARIMA model works better for short-term forecasting, capturing immediate trends and patterns. On the other hand, the LSTM model, being memory-based, is more suitable for long-term forecasts, effectively capturing more complex dependencies over time.

7 Conclusion

Our project provided us with an excellent overview of what the real world looks like. We had to work on top of an existing infrastructure, which added a layer of complexity to our project meaning that, instead of starting from scratch, we had to adopt the established frameworks and methods, so that we could eventually integrate our work. Because of this, the initial and most complicated part of our project was understanding and interacting with all the existing technologies.

7.1 Summary of the Project

Our project had two main goals, both successfully achieved:

- Aggregation of data coming from multiple sources;
- Develop a forecasting model, using machine learning.

To generate our events, we had to aggregate information coming from different sources: from the infrastructure RADARs, LIDARs and cameras, from HERE and from the OpenWeather API. All the data received, except from OpenWeather, were then stored in our database so that we could furtherly use it to generate events, make correlations and develop and train our ML model. We implemented ten different types of events: accidents, road work, congestion, road hazards, immobilized vehicles, closed roads, police presence, floods, potholes and cars going the wrong way. All these events generate real-time alerts that are then displayed on the map accordingly.

To develop a forecasting model, we first had to understand the data and identify the possible correlations that existed between the traffic volume and the different periods of the day/month/year. This analysis helped us uncover various trends and seasonality patterns within the data, which allowed us to build a more accurate forecasting model. In the end we end up with two different types of models, which allowed us to make some comparisons. The first model, following a statistical approach, focused on short-term forecasting, providing meaningful predictions for the next day. The second model focused on long-term forecasting, following a deep learning approach based on memory, which allowed us to capture accurate trends and seasonality over multiple weeks.

7.2 Achievements and benefits of Safe Roads

Both of our project main goals were successfully achieved, resulting in significant benefits for ATCLL infrastructure. The implementation of our data aggregation system allows users to have a concise and accurate overview of the traffic congestion, road works, closed roads and other relevant information.

This allows individuals to make informed decisions and take appropriate actions to navigate the road network more efficiently. Whether it's planning alternative routes to avoid congestion areas, adjusting travel times to off-peak hours, or simply staying informed about road closures and ongoing road works.

7.3 Limitations

The main limitation in our project is that the forecasting model was not deployed. This can be explained by the connection problems we've had with the ATCLL API, which prevented us from accessing the real-time data necessary to train and develop our forecasting model. Other limitation lies in the fact that only some of the SLPs were operational, which restricted the coverage and availability of data for our project. One final limitation has to do with the OpenWeather API, since it only allows the retrieval of hourly data in a max span of 7 days, for bigger date intervals, we need to call it multiple times to get appropriate data to show in the graphs (weekly, monthly and yearly).

7.4 Future work

Although the current project has produced encouraging results, there are still plenty of opportunities for exploration and improvement in the future:

- **Inclusion of weather conditions:** incorporating weather data from the OpenWeather API into the forecasting model could significantly enhance its accuracy. Weather conditions have a substantial influence on traffic volume and patterns, and incorporating this variable as an input feature can provide more reliable predictions.
- **Advanced feature engineering:** exploring additional features that can further capture the intricacies of traffic patterns could enhance the model's performance. Factors such as holidays, events, road construction, or public transportation schedules might contribute to fluctuations in traffic volume. Including these variables as input features could improve the model's ability to capture and forecast these nuances.
- **Real-time data updates and model deployment:** to deploy the model in a production environment, it is essential to incorporate real-time data updates. Modifying the code to receive and process live data directly the ATCLL API would ensure that the model remains up-to-date and maintains its accuracy over time.

8 References

- https://towardsdatascience.com/machine-learning-part-19-time-series-and-autoregressive-integrated-moving-average-model-arima-c1005347b0d7 , visited in 13/05/2022;
- https://levelup.gitconnected.com/simple-forecasting-with-auto-arima-python-a3f651271965 , visited in 13/05/2022;
- https://towardsdatascience.com/time-series-prediction-with-lstm-in-tensorflow-42104db39340, visited in 19/05/2022;
- https://towardsdatascience.com/exploring-the-lstm-neural-network-model-for-time-series-8b7685aa8cf, visited in 19/05/2022;
- https://www.analyticsvidhya.com/blog/2020/10/multivariate-multi-step-time-series-forecasting-using-stacked-lstm-sequence-to-sequence-autoencoder-in-tensorflow-2-0-keras/, visited in 19/05/2022;
- https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/, visited in 19/05/2022;
- https://medium.com/mlearning-ai/univariate-time-series-forecasting-using-rnn-lstm-32702bd5cf4, visited in 19/05/2022;
- Almeida, A., Brás, S., Oliveira, I., Sargento, S. (2022). Vehicular traffic flow prediction using deployed traffic counters in a city. Future Generation Computer Systems, 128, 429-442. ISSN 0167-739X. DOI: 10.1016/j.future.2021.10.022.
- Project website: https://trafegopi7.vercel.app
- Project repository: https://github.com/ATCLL-Safe-Roads (only backend, since frontend is in a private GitLab repository)