NOVA IMS Information Management School

Master Degree Program in

MDSAA

Data Science and Advanced Analytics

Analysis of alerts, incidents and traffic patterns in Oeste region

Ana Carolina Lopes Dias de Oliveira Gonçalinho

Master Thesis

presented as partial requirement for obtaining a Master's Degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Analysis of alerts, incidents and traffic patterns in Oeste region

by

Ana Carolina Lopes Dias de Oliveira Gonçalinho

Master Thesis presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics, with a specialization in Business Analytics

Supervised by

Supervisor: Miguel de Castro Neto, PhD, NOVA Information Management School

Co-supervisor: Bruno Jardim, PhD, NOVA Information Management School

July, 2024

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Ana Carolina Gonçalinho

Lisbon, 11/07/2024

ABSTRACT

In a world where time is one of the biggest commodities, traffic very often leads to its waste. By having the possibility to know a priori the traffic conditions and incidents, people can plan their daily lives more efficiently, leading to less waste of their time, and less fuel consumption. With the developments in Artificial Intelligence, more specifically in Machine Learning, it is possible to resort to different models to predict traffic flow and future incidents and alerts, using previously recorded traffic data. Former research explores different approaches, mostly focusing on Time Series or Neural Networks, which have already shown that can predict traffic patterns with reasonable accuracy and present the importance of the features. This research aims to develop a high-performance model able to identify major traffic factors and their impact. In this study were applied Machine Learning models as Neural Networks, Decision Trees and ensemble models, i.e., Bagging, Random Forests and Gradient Boosting, and some Deep Learning algorithms, such as Multi-Layer Perceptron, Convolutional Neural Networks and Very Deep Convolutional Network, allowing the comparison of these two approaches, simple versus more complex algorithms. The selected models were Gradient Boosting for traffic prediction and Neural Networks for alerts and incidents prediction. Final results were discussed, improvements and recommendations were provided.

KEYWORDS

Traffic Management; Traffic Prediction; Machine Learning; Data Visualization; Intelligent Transportation Systems

Sustainable Development Goals (SDG):

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN/NN	Artificial Neural Networks / Neural Networks
BI	Business Intelligence
CNN	Convolutional Neural Network
DA	Data Analytics
DL	Deep Learning
DT	Decision Trees
ELU	Exponential Linear Unit
ITS	Intelligent Transportation Systems
ML	Machine Learning
MLP	Multi-Layer Perceptron
ReLU	Rectified linear unit
RF	Random Forests
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SHAP	Shapley Additive Explanation
softmax	Normalized Exponential function
softplus	Smooth version of ReLU function
tanh	Hyperbolic tangent function
TFP	Traffic Flow Prediction
TM	Traffic Management
VGG	Very Deep Convolutional Network

1. INTRODUCTION

1.1. MOTIVATION

Transportation is a fundamental component of modern society. It influences daily routines and affects various aspects, from productivity and general quality of life to people's health and the environment. At an era when time is one of the most important commodities, traffic is very often an obstacle to its efficient management. Time spent in traffic has increased globally in recent years (with the exception of the COVID period, when levels were lower due to mobility restrictions), with several cities exceeding 100 hours lost to traffic congestion last year[1]. In addition, traffic accidents claimed 1.3 million lives, and 20-50 million people suffered non-fatal injuries[2].

Traffic Management (TM) has gained significant importance in recent years due to the increased awareness of road safety, the negative social impact of traffic congestion and the proven reduction in economic efficiency (Fattah et al., 2022). This has occurred with the introduction of Intelligent Transportation Systems (ITS), which enable wireless communication and control of the entire transportation network, improving its safety, efficiency and convenience (Shaheen & Finson, 2016). With the growing interest in this topic, research has been developed in the areas of Data Analytics (DA) and Machine Learning (ML), impacting areas from traffic prediction (Boukerche & Wang, 2020) to route planning (Liebig et al., 2017).

In Portugal, traffic management can be improved through the introduction of sophisticated and efficient ITS. The increase in available traffic data enables the recognition of traffic patterns, the classification of traffic type, the identification of incidents and their severity (Vieira Gomes et al., 2008), the reduction of travel times and the increase of road capacity (Vanderschuren, 2003).

1.1.1. Traffic Flow, Alerts and Incidents

Traffic Flow is the movement of vehicles between places and the interactions that take place between them. It occurs when traffic demand equals or exceeds the capacity of the road network (Raheem et al., 2015). Although it is possible to quantify some of the factors that influence the “flow” of vehicles, other factors are not so easy to assess.

The characteristics of traffic are constantly evolving, leading to two types of traffic conditions: an *interrupted traffic flow*, which is controlled by external factors such as stop signs or accidents, and an *uninterrupted traffic flow*, where the flow is determined by the various interactions between vehicles and not by external elements such as traffic signals (Kumar

[1] INRIX, *2022 INRIX Global Traffic Scoreboard* available at https://inrix.com/scorecard/

[2] World Health Organization, *Road Traffic Injuries* available at https://www.who.int/health-topics/road-safety#tab=tab_1

Sharma & Swami, 2016). These categories are identified by examining traffic parameters such as speed, volume and occupancy (Cinsdikici & Memiş, 2010).

Alerts and incidents can be a variety of events, from a car stopped on the side of the road, to roadworks, fog and bad weather conditions. These events have a significant impact on traffic flow and change the interactions between vehicles on the road network.

1.1.2. Traffic Flow Prediction

As already mentioned, traffic flow is influenced by several elements, some of which are neither known "a priori" nor can be easily evaluated. However, we can distinguish the different components of traffic flow, such as the number of cars circulating, weather conditions, traffic signs and others.

The accurate estimation of traffic flow at a specific location at a specific time in the future, known as Traffic Flow Prediction (TFP) (Kashyap et al., 2022), is the most important tool for various traffic management functions, such as urban route planning (Song et al., 2015), reducing traffic congestion (Walraven et al., 2016), minimizing urban pollution/carbon emissions (Peng et al., 2020) and improving road safety (Alajali et al., 2018).

Since traffic management has been an issue since the end of the last century, research in this area was developed. There are various ML techniques that can be applied to this type of research. Some of them are simple to understand while others are more complex and require additional knowledge about prediction processes.

1.1.3. OesteCIM Challenge

By focusing on the Oeste region of Portugal, it is possible to identify and develop more appropriate and specific solutions for traffic management in this region, rather than taking a general approach and applying its conclusions to a smaller area. Furthermore, by analyzing the future results, users could benefit from the identification of critical traffic points, problematic roads and congested periods.

OesteCIM proposed the challenge of developing analytical models and visualization tools for the characterization and prediction of traffic patterns, alerts and incidents in the Oeste region through the processing and analysis of data from the WAZE application. OesteCIM is a Portuguese public entity focused on the development of a green, digital and inclusive future to improve people's quality of life.

Waze is one of Google's subsidiaries that works as a Global Positioning System (GPS) and provides satellite images on smartphones or computers via navigation software. Users share real-time information via the application by reporting the current traffic situation and road structure, such as traffic, accidents, police traps and closed roads. The data is collected, and optimal routes are designed 24 hours a day. Figure 1 shows how the app works and how it processes the data.

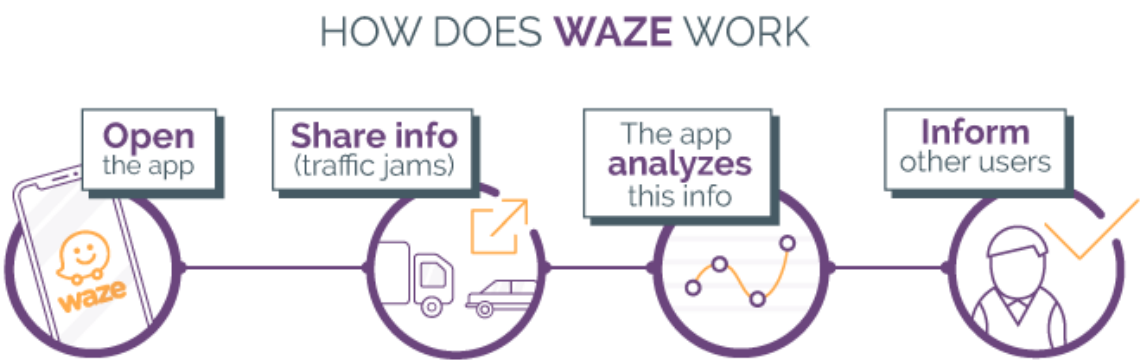

Figure 1.1 - Waze data processing pipeline. Available at https://rubygarage.org/blog/how-to-make-a-gps-app-like-waze

As with most data-related projects, the accuracy of predictive models depends heavily on the quality of the data. When working with user-supplied data, as in this project, it is important to watch out for inaccurate or incorrect information that can negatively impact the accuracy of the models. Therefore, the data must be carefully filtered to exclude irrelevant or misleading information that could prevent the model from understanding the targets.

OesteCIM seeks to gain insights into the traffic conditions provided by WAZE users. The company intends to obtain a comprehensive set of metrics and visualizations as well as a predictive model that can provide insights into future traffic conditions and incidents. This model will enable OesteCIM to know about traffic in advance, which will lead to better route planning in the future.

The development of a model for traffic prediction, based on WAZE data, will allow its incorporation in the application features for better route planning ahead of time and scheduling, dynamic traffic control, among others.

Regarding alerts prediction, this type of information is only obtained when a user reports an alert, so the type of incident is registered, but often different categories are assigned to the same event. For that purpose, a model will be developed to predict the type of alert, so the previous situation can be corrected in future scenarios.

1.2. Objectives and Methodology

The main objective of this study is to gain knowledge about traffic patterns, incidents and alerts while developing an effective and accurate predictive model that can forecast these parameters. The main objectives of this research are the following:

1. Discussion of the literature, the Machine Learning algorithms used for the predictions and the limitations encountered.
2. Elaboration and development of metrics and visual representations to characterize current traffic patterns, incidents and alerts.

3. Identification of current factors that significantly affect traffic conditions.
4. Selection and development of analytical models to predict traffic patterns, alerts and incidents.
5. Evaluation of the quality of the results obtained compared to real values.

In order to achieve the above objectives, this work followed a specific structure to ensure that each step was carried out:

1. **Theoretical Background** provides an introduction to the concepts, particularly Machine Learning, used in the development of the project and in previous research.
2. **Literature Review** presents and discusses previous literature and methods used to predict traffic flow and factors that influence traffic conditions.
3. **Methodology** explains the processes that led to the development and selection of the visualizations and final model. Presents the data preparation and model development as well as the comparison between the different models studied.
4. **Results and Discussion** reports on the results obtained and their quality.
5. **Conclusions** summarizes the research, from an initial business and data understanding to the results obtained.
6. **Limitations and recommendations for future work** presents the main obstacles and limitations encountered in the development of this research and points out interesting targets for future work.

2. THEORETICAL BACKGROUND

Before discussing previous research, it is important to understand some basic ML concepts to better understand what was carried out in the literature and what will be developed in the future phases of this project.

2.1. MACHINE LEARNING

ML is a branch of Artificial Intelligence (AI) that attempts to imitate the way humans learn. Data and algorithms are used to develop this learning process. ML algorithms can be divided into different categories, including supervised, unsupervised and reinforcement learning. Of these categories, supervised learning is the most important for this study.

Supervised learning is a category of algorithms that receive data with the desired outcome and learn from it by developing a function that correctly reproduces the output of unlabeled data, i.e., data without the desired outcome. By receiving training examples, the algorithm runs through several tests on the training data and finds the best way to use the set of variables to predict the unseen data.

When working with this type of algorithm, it is important to pay close attention to the variable we want to predict, typically denoted as y. If y is classified as discrete, i.e., it can be counted and has a limited number of values, then we are dealing with a classification problem. On the other hand, if y is continuous, we are dealing with a regression problem. It is important to understand what type of problem we are dealing with, as this will help determine the appropriate methodology and evaluation metrics to be used later in the project.

One of the most popular ML algorithms is the linear model, which includes Linear Regression (for regression) and Logistic Regression (for classification). These models establish a relationship between the dependent variable, denoted as y, and the set of independent variables, denoted as $x = \{x_1, x_2, \dots, x_n\}$. Linear Regression creates this relationship by fitting the best straight line to the observed data, using the following equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon \quad (2.1)$$

where β_0 represents the intercept, i.e., the point where the line crosses the y axis of the graph, the other βs represent the average amount the dependent variable increases when the corresponding independent variable increases by one standard deviation while the other independent variables are held constant, and ε is the error term, which accounts for what is not being accounted for in the equation, such as other variables.

Logistic Regression uses a logistic function to compute probabilities, mathematically referred to as the sigmoid function. By applying equation 2.2, this algorithm predicts whether an observation belongs to one category or another.

$$p = \frac{1}{1 + e^{-y}} \tag{2.2}$$

where p is the probability and y refers to equation 2.1. This algorithm uses the results of linear regression, as its equation is used to calculate the probabilities. We can conclude that the algorithm transforms its input into a nonlinear form, meaning that the line drawn on a graph is not straight, ensuring that the outputs fall within the interval [0,1] and guaranteeing that non-valid probability values are not returned.

In 1763, Thomas Bayes proposed Bayes's theorem, a method to compute the probability of a hypothesis knowing some evidence. In ML, the model associated with this theorem is called naïve Bayes. This theorem leads to the calculation of the conditional probability of y given x, where the category assigned to y is the one with the higher probability. For that, it is necessary to compute the joint probability of x and y, $P(x, y)$. Equation 2.3 presents the calculation performed to obtain the conditional probabilities:

$$P(y|x) = \frac{P(y) * P(x|y)}{P(x)} \tag{2.3}$$

where $P(y)$ is the probability of y belonging to a specific class, $P(x|y)$ represents the likelihood of observing the sample x if the hypothesis holds (i.e., if y belongs to the assigned class), and $P(x)$ represents the marginal probability that sample x is observed under any circumstances. Due to its simplicity, this model often produces satisfactory results. However, due to its assumption of class conditional independence, it sometimes faces loss of accuracy.

Decision Trees can work as both a classifier and a predictor, and they focus on a recursive partition of the instance space (Rokach & Maimon, 2006). Their primary goal is to discriminate instances between different classes and obtain leaves that are as unique as possible, i.e., each leaf having only instances of a particular class. Due to their interpretability, ability to deal with different data types, insensitivity to scales and automatic feature importance definition, Decision Trees have become widely adopted and efficient predictive models for several research fields.

Despite its popularity and advantages, this kind of model often requires a discrete target. Additionally, since the trees are constructed based on the supplied data, even slight variations in it can lead to vastly different results, and replication may occur in existing sub-trees. The definition of the desired tree width and variables to query is the principal step in developing a tree. There are several approaches to choose the attribute to use for partitioning, such as entropy (with information gain) and gini index, both focused on maximizing the benefit of choosing a variable for partitioning.

In addition to individual models, there are also ensemble ones, which are the aggregation of multiple algorithms in order to obtain better predictions than single models. Bootstrap

Aggregation, as known as Bagging, is used to improve stability and accuracy of ML algorithms. It develops multiple subsets (bootstrap samples) of the original dataset through random sampling with replacement. Then, a base model is trained on each bootstrap sample independently. After training the multiple base models, bagging combines their predictions through averaging (for regression) or voting (for classification). The predictions of the different models are aggregated to produce a final output, reducing the variance and improving the generalization of predictions.

Another important ensemble model is Random Forests. This model is an extension of the bagging technique, which creates an ensemble of DTs, where each tree is trained on a random subset of the training data and a random subset of features. Initially a random subset of the data is selected with replacement and a subset of features is also obtained. Each DT is trained independently on its respective data sample and random subset of features. After the training is performed, the predictions are obtained for each tree, and the class that receives the most votes is chosen as the final prediction for a specific observation.

The last ensemble model worth mentioning is Gradient Boosting. Contrarily to Bootstrap Aggregation and Random Forests, this model builds an ensemble of weak learners sequentially, where each learner corrects the errors made by the previous one. Instead of training the new model with original target values, it is trained with the residuals, focusing on the areas where the ensemble is performing poorly.

Another famous algorithm is the Artificial Neural Network (ANN), modeled to mimic the way biological neurons communicate with each other. It replicates a network of interconnected neurons (Mijwil et al., 2019), transferring information within the chain in order to learn the best approach to solve a specific problem. Over the years this type of model was widely developed, and new types have appeared, such as Multi-Layer Perceptron, Convolutional Neural Networks and Recurrent Neural Networks.

2.1.1. Multi-Layer Perceptron

Developed from the simplest ANN, which is composed only of an input layer with the corresponding weights w_i, a bias parameter w_0 and an activation function $a(z)$. The single perceptron (proposed by Rosenblatt in 1958) receives the inputs and computes the product between them and their corresponding weights, summing up the values with the bias parameter. The obtained value goes through the activation function and the output is obtained.

There are several activation functions that can be applied, being the most famous and widely adopted the tanh, $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$, the sigmoid function, $f = 1/(1 + e^x)$, and the threshold function, $f(x) = \begin{cases} 0 \; if \; x < 0 \\ 1 \; if \; x \geq 0 \end{cases}$.

In the Multi-Layer Perceptron there is more than one layer where transformations occur. It is composed of input and output layers, and between them can exist one or more hidden layers. In the input layer the input values are inserted into the network, but no activation function is applied, nor any kind of processing is performed. In the hidden layers the values are classified, and a bridge is built between the input and output layers. Finally, the output layer has the same functionality as the hidden layer and the final outputs go outside the network.

In the Perceptron model, the product between the input and the weights w_i is computed. Then, the obtained values are summed up together with the bias parameter b_i, and an activation function is applied for every node of the following hidden layer (Singh & Banerjee, 2019). Once every calculation is performed, the output is passed to the next layer, hidden or output ones. The computations between the hidden layer and the next layer are the same as described when data passes from the input layer to the first hidden layer. The values obtained at the output layer are then retrieved as the final results.

It is possible to train a neural network for it to adjust its weights w_i and the bias parameter b_i. Through the incorporation of Backpropagation algorithm (Linnainmaa, 1976; Rumelhart et al., 1986), which computes the gradient (derivative) of the sigmoid function (equations 2.4 and 2.5) and updates the weights of the network, the model improves and achieves more exact predictions.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

$$\frac{\partial}{\partial x}\,\sigma(x) = \sigma(1 - \sigma) \tag{2.5}$$

The error function compares the predicted values $\hat{y}$ with the real ones y, and the new weights intend to minimize the error by subtracting to the value of the current weight the derivative of the error function. This process, denominated as Gradient Descent algorithm (Cauchy, 2009), works as an optimization approach to minimize the prediction error of ANN.

2.1.2. Convolutional Neural Network

CNN (LeCun et al., 1989) are one of the variants of traditional ANN, where the biggest difference relies on its suitability for pattern recognition within image and videos. For that reason, image-specific features are encoded in its architecture, improving its appropriateness for image and video-focused tasks, but reducing the parameters required to build up the model (O'Shea & Nash, 2015).

These networks are composed of three types of layers: convolutional layers, pooling layers and fully-connected layers. The convolutional layer is the core component of the CNN architecture and is responsible for feature extraction, a combination of convolution operation with activation function. Given the input data, denominated as tensor, and a kernel (small matrix of weights), an element-wise multiplication between the elements is performed at

each location of the tensor and summed to achieve the output value in that corresponding position, denominated as feature map (Yamashita et al., 2018).

The calculation explained above can be expressed mathematically as follows:

$$h^k = f(W^k * x + b^k) \tag{2.6}$$

where k is the number of feature maps to create, h^k is a new variable (the future feature map), f is a non-linear function, W^k are the weights of the kernel, x is the input of each layer, and b^k is the bias term. In order to obtain an arbitrary number of feature maps, the process must be repeated with multiple kernels, obtaining maps representing different characteristics of the tensors.

Then, the feature maps go to the pooling layer, which is responsible for sub-sampling them, transforming the original features maps into smaller ones. The most frequently adopted approach is the Max-pooling, which retrieves only the maximum value $\hat{h} = \max\{h\}$ of each feature map. It is important to refer that sub-sampling does not preserve the position of the information, so it should only be applied when spatial information is not as important (Albawi et al., 2017).

The result obtained in the previous layer is fed into the fully connected layer, where the dot product between the kernel and the input is computed to obtain the final result. The process is remarkably similar to the one of a Perceptron or a Multi-layer Perceptron, where an activation function is applied in the output layer allowing the classification of the input over determined classes.

The most commonly applied activation function is Rectified Linear Unit (ReLU), due to its simplicity of calculating the partial derivative, its lower training time and the fact that it does not allow gradients to disappear (Indolia et al., 2018). In the fully-connected layer, the activation function is applied to the sum of the results obtained in the dot products mentioned above.

2.1.2.1. Very Deep Convolutional Network

VGG model presents a standard deep CNN architecture with multiple layers. Its two variations, VGG-16 and VGG-19 are composed of 16 and 19 convolutional and fully-connected layers, respectively, that is why their denominated “deep”.

VGG’s convolutional layers use small convolutional filters with 3x3 size, and the convolution stride is fixed as 1 pixel and padding to preserve spatial resolution. The idea behind these layers is to stack multiple small filters to obtain a receptive field equivalent to larger filters.

The pooling layers use ReLU as activation function and filters of 2x2 with a stride of 2 in order to reduce the spatial dimensions of the feature maps and control overfitting. Finally, there are

usually 2 or 3 fully-connected layers followed by a softmax layer that outputs the class probabilities.

Since VGG are deeper than common CNN, they are able to better capture more complex patterns. Also, the use of small convolutional filters makes the network design straightforward and modular, but despite their simplicity, VGG networks have shown competitive performance on image recognition benchmarks.

Even though the presented advantages, these networks have a large number of parameters which makes them computationally expensive in both terms of memory and computational power. Also, due to its depth and number of parameters, they require significant time and computational resources to train.

2.1.3. Performance Metrics

After the development of the models its necessary to assess them, which is done through the estimation of the prediction error on new data (data not yet seen by the model). As the models differ regarding the type of problem faced, the performance metrics are also different for classification and regression approaches.

2.1.3.1. Classification Problems

- **Confusion Matrix**: It is related to the accuracy of the model, displaying the number of true positives, true negatives, false positives and false negatives. It leads to the identification of misclassifications allowing the discovery of model's primary errors. Table 2.1 represents this.

Table 2.1 - Confusion matrix

		Real Values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN
		P	N

  - **True Positive (TP)**: The predicted value matches the real value. Predicted class is positive and real value was also positive.
  - **False Negative (FN)**: False prediction, also known as type II error. Predicted class is negative but real value was positive.
  - **False Positive (FP)**: False prediction, also known as type I error. Predicted class is positive but real value was negative.
  - **True Negative (TN)**: Predicted value matches the real value. Predicted class is negative and real value was also negative.

- **Accuracy**: Proportion of events correctly classified among all results. Should be used when both positive and negative have the same importance and the dataset is balanced.

$$Accuracy = \frac{TP + TN}{P + N} \tag{2.7}$$

- **Precision**: Proportion of correctly classified positive events among all results. Should be used when the cost of false positive is high, the model should focus on predicting positives correctly.

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

- **Recall (Sensitivity)**: Proportion of events identified as positive on all positive events. Should be used when the cost of false negatives is too high.

$$Recall = \frac{TP}{TP + FN} \tag{2.9}$$

- **Specificity**: Proportion of events identified as negative on all negative events. Similar to Recall but applied to the negatives.

$$Specificity = \frac{TN}{FP + TN} \tag{2.10}$$

- **F1 Score**: Precision and Recall can be used together in a single metric. It should be used when balance between Precision and Recall is needed and when there is uneven class distribution.

$$F1\ Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{2.11}$$

3. LITERATURE REVIEW

3.1. TRAFFIC MANAGEMENT

Traffic congestion in Portugal, especially in Lisbon, occurs because of economic activity, road accidents, maintenance work, road features and its poor condition, and the number of vehicles circulating[3]. Other macro-level factors should be considered, such as land-use planning and car sharing habits (Vencataya et al., 2018). In Figure 2 are listed the major factors, both at micro and macro-levels, which affect traffic flow and lead to the drawbacks of traffic congestion.

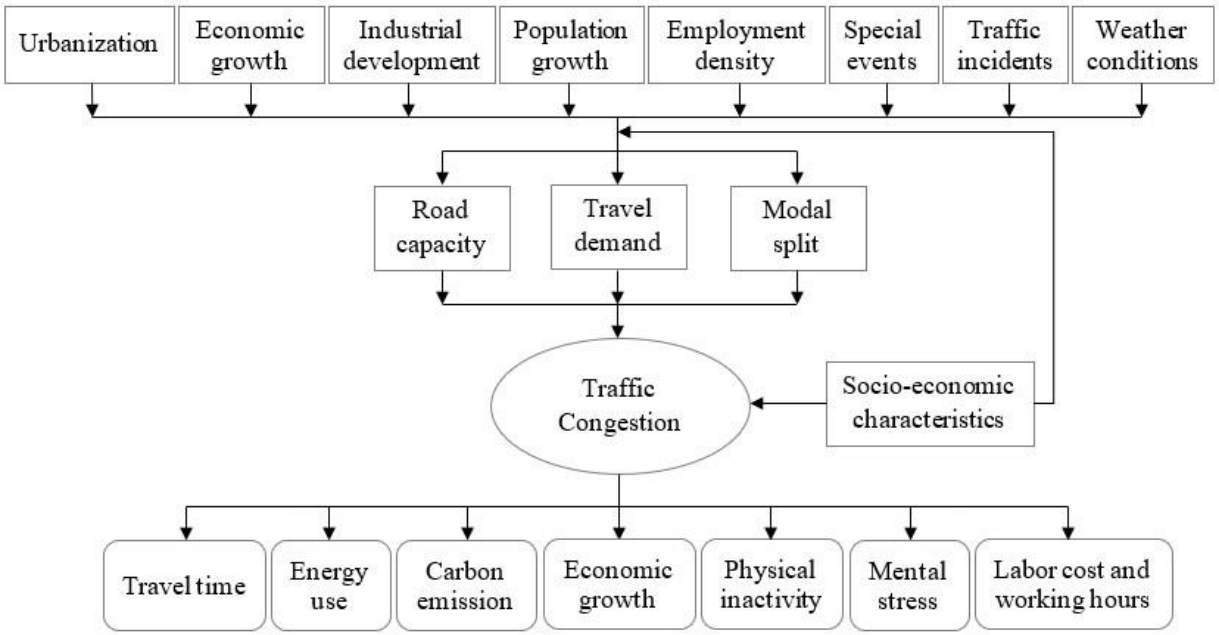

Figure 3.1 - Factors of Traffic Congestion and its Drawbacks (Rahman et al., 2022)

Of the factors presented in Figure 2, only a few were often adopted in previous research. Next is presented a small explanation of those factors and references of their incorporation:

- **Special events** are associated with activities and local events, such as sports competitions or concerts. These factors are not related to traffic, but they usually imply higher travel demand and/or route adjustments due to closed roads (Taghipour et al., 2020).
- **Traffic incidents** refer to unexpected and non-recurrent traffic events, such as accidents, road maintenance, etc., that impact traffic flow and might lead to congestion (Taghipour et al., 2020; Yao & Qian, 2021).
- **Weather conditions** go from temperature and humidity to wind and precipitation. Meteorological elements can negatively affect drivers ability, leading to potential accidents and congestion (Taghipour et al., 2020; Yao & Qian, 2021).

[3] Seguropordias, *O congestionamento nas estradas de Lisboa* available at https://seguropordias.pt/blog/tr%C3%A2nsito-lisboa-portugal

There are other factors that impact traffic, such as Road condition, Road characteristics, Drivers ability, among others. It is important to notice that the majority of the factors presented above are also key features impacting alerts and incidents. Regarding one specific incident, categorized as road accidents, 94% occur due to human error[4] and depend on drivers characteristics: Skill level, experience, risk behaviors, etc. (Rolison et al., 2018).

3.2. ALERTS, INCIDENTS AND TRAFFIC CHARACTERIZATION

Events characterization is majorly done by visualizations and Key Performance Indicators (KPI). In this section, previous work relating to this analysis of traffic and incidents is presented focusing on WAZE and similar applications. Most traffic visualizations have a temporal component, allowing to understand traffic patterns and incidents over time and its seasonality.

Buchmüller et al. (2019) proposed a new visualization, MotionRugs, able to display changes in spatiotemporal data of multiple entities based on space partitioning techniques. The representation is made in a one-dimensional ordered slice for each moment separately. It consists of a temporal heatmap, encoding spatio-temporal behavior of multiple entities at abstracted spatial dimensions.

Sun et al. (2017) developed a new visualization technique, Route-Zooming, able to integrate spatio-temporal information into maps for direct analysis of spatio-temporal data. In the visualization there is a direct overlaying of time series visualizations on road maps, obtaining a spatio-temporal combination. It allows not only the comparison of traffic in both road directions, but also of different time periods such as weekdays, months or even specific moments.

Guo et al. (2011) proposed a new visual analytics system, Triple Perspective Visual Trajectory Analytics (TripVista). With this system, it is possible to analyze trajectories from different angles, including spatial, temporal and multi-dimensional perspectives. It allows the visualization of microscopic traffic trajectory data at a road intersection.

3.3. PREDICTIONS

In order to perform traffic and alerts prediction, the input is required to contain a traffic variable, as flow or speed, represented in a spatio-temporal manner, so the output is a forecast of future conditions (Shaygan et al., 2022). The incidents/alerts forecasting problem is also addressed as a spatio-temporal problem, but the forecast is only performed to foresee future events, instead of future conditions.

[4] Wilson Kehoe Winingham, LLC, *Common Causes of Car Accidents* available at https://www.wkw.com/auto-accidents/blog/10-common-causes-traffic-accidents/

Despite only recently being considered, it is important to notice that accurate traffic prediction should account for not only temporal changes, but also the structure of the road network, spatial parameters. There are two categories of methods traditionally used in traffic forecasting: traditional ML and DL. The first category covers several models from approaches that are computationally simpler and more appropriate with small datasets, to methods that provide better interpretability and higher precision in identifying complex non-linear relations in the data.

On the other hand, DL models present even more complex structures, with increased computational skills, leading to better accuracy and precision in forecasting than some of the most used traditional methods. As mentioned before, the capability to capture spatio-temporal relationships is crucial in traffic forecasting, so the development of hybrid solutions was highly adopted in recent years (Gu et al., 2020; Hajirahimi & Khashei, 2019).

In the next sections, a literature review is presented for the two categories, exploring their contributions to the topic, main discoveries and obtained results. It focuses more on traffic flow prediction, being the more complex and researched topic, while traffic occurrence, alerts and incidents prediction can still use similar methodologies.

3.3.1. Traditional Machine Learning methods

The simpler models of this category, denominated as statistical methods, apply mathematical methods and statistical assumptions to make predictions from the provided data. The majority of these models work as a regression, frequently applied to short-term traffic prediction due to its simple interpretability and its computational requirements.

Regarding non-linear models, there are other approaches frequently adopted, such as KNN (Harrou et al., 2020; Zhang et al., 2013), Historical Average (HA), Vector Autoregressive (VAR) models, Bayesian networks (BN), among others. These presented algorithms are considered to be basic and off easy interpretation but are frequently outperformed by other more complex models.

One of the most well-known autoregressive models is the Autoregressive Integrated Moving Average (ARIMA) and its variants, approaches that are widely adopted in traffic prediction problems (Li et al., 2012; Lippi et al., 2013; Moreira-Matias et al., 2013; Shekhar & Williams, 2007; Wagner-Muns et al., 2018; Williams & Hoel, 2003). The most applied variants of ARIMA present in literature are Seasonal ARIMA (SARIMA) (Box et al., 2015), ARIMA with explanatory variable (ARIMAX) (Box & Tiao, 1975), Kohonen ARIMA (KARIMA) (Dervoort et al., 1996)and Vector ARIMA (Sims, 1980), revealing an improvement in accuracy and quality of traffic predictions when compared to basic models.

Even though these models are able to produce good forecasting, they are not able to account for the spatio-temporal relationship due to its exclusive focus on time series data. To overcome this obstacle, some researchers developed innovative extended time-series

approaches, then able to account for both spatial and temporal fields, such as ST-ARIMA (Min et al., 2009, 2010), with promising results (outperforming other approaches for short-term traffic prediction), and VARMA (Min & Wynter, 2011), also with results beyond expectations.

Some of the major disadvantages of this type of model are associated with inadequacy to account for the spatial dependencies, its preference for short-term predictions and smaller datasets, and its inefficiency to process complex time series data. Even though some variants of basic models were developed, these disadvantages were not completely surpassed.

Even with the disadvantages, some hybrid models were developed associating most basic models with DL methods that can solve these drawbacks. For example, the incorporation of RNN-based models allows the processing of long-term dependencies, while the usage of Bayesian models allow the determination of feature importance at different times, cutting back low-importance variables affecting models prediction capabilities, preventing overfitting.

Contrarily to previously presented methods, there are more complex models that are able to learn without being explicitly programmed, being able to identify more complex relationships and better adapt to changing conditions over time (Yang et al., 2020).

In this type of model, it is possible to identify two categories: Feature-based models and Gaussian models. The first category focuses on identifying and optimizing key features of input data and has been explored in recent research with the incorporation of new techniques (Guan et al., 2018; Kowalek et al., 2019; Li et al., 2019; Lin et al., 2018; Tang et al., 2019), with different conclusions but overall accurate predictions for the presented modifications. In spite of producing general good predictions, these models present a major disadvantage: since they constantly try to optimize the results and feature incorporation, the results are inconsistent, leading to different predictions within several applications.

Regarding Gaussian models, they work as probabilistic methods, mimicking normally distributed subpopulations within the overall population. Recent research has proven that Gaussian methods are accurate and practical for traffic forecasting (Kowalek et al., 2019; Matthews et al., 2018; Salinas et al., 2019), being able capture spatio-temporal relationships. The major drawback of these methods, and when compared with feature-based ones, is the higher computational complexity and bigger data storage ($O(n^3)$) (Datta et al., 2016), which leads to scalability problems when dealing with large datasets.

3.3.2. Deep Learning

With the development of modern technologies in recent years, AI also experienced a huge improvement, especially in terms of DL approaches. Recent research revealed its good ability to predict traffic and capture non-linearity (Lai et al., 2017; Shih et al., 2018). Even though DL methods require more computational effort and storage requirements, they frequently outperform most popular traditional approaches.

Oliveira et al. (2016) revealed the benefits of Rprop, a resilient backpropagation algorithm that dynamically updates the learning rate of each neuron connection, in traffic prediction through the comparison of the new model with MLP, Recurrent Neural Networks (RNN) and stacked autoencoder (SAE), revealing to achieve better results than the other tested models. This technique was developed due to the challenges of dealing with gradients, such as its updating and transmission, which can lead to poor results when training on large networks.

Additionally, Song et al. (2017) compared MLP-based models with CNN-based ones, concluding that the second approach revealed better performance, due to its ability of capturing local dependencies and being less sensitive to noise in data. The study revealed strong results, with performance measures showing precise predictions.

Restricted Boltzmann Machine (RBM) is also a reliable approach for traffic prediction. Ma et al. (2015) applied stacked RBM (RNN-RBM), to retrieve spatial and temporal correlations from traffic data, achieving good understanding of characteristics of the data and interactions between the variables. Tests were made with different speed thresholds, achieving strong results with lower values.

More recently, Bao et al. (2021) proposed a new variant for RBM, an improved Deep Boltzmann Machine, integrated with Support Vector Regression (SVR), to draw complex relationships between different components of traffic system. Experimental results showed that this method outperforms other models such as ARIMA time series and Neural Networks in traffic prediction.

Convolutional Neural Networks are shown to be highly effective retrieving localized spatial dependencies from multivariate time series data. Wu et al. (2018) proposed a hybrid model combining CNN/RNN and Long Short-Term Memory (LSTM) with an attention-model (DNN-BTF), allowing the retrieval of both spatial, by CNN model, and temporal trends, by LSTM model. The attention model has as its primary objective the calculation of correlation scores between previous traffic values and its future ones. The developed model revealed encouraging prediction accuracy.

Liu et al. (2019) proposed an innovative approach, DeepRTP, a modified CNN able to remove the pooling layer. Contrarily to traditional CNN, this novel approach performs 2D convolution on a 2D matrix representation of traffic data while pre-processing the data into different time units. With the new representation, inter-regional relationships are better retained during training. This new model incorporates CNN with a Residual Neural Network.

In the early stage of data pre-processing, a new variable is computed (Traffic State Index) for all regions and then data is converted into a two-dimensional matrix. Then, the resulting matrices are categorized according to their temporal characteristics (hourly, daily or weekly), and applied to different model components (all with same structure). The final predictions are obtained through the application of a tanh function to the grouped results obtained in the models. The experimental results obtained in the study revealed that the new approach

significantly outperforms other methods (HisAve, HisAve-w, ARIMA, SARIMA, VAR) by achieving higher prediction accuracy.

Luo et al. (2019) designed an optimized LSTM model by integrating KNN, developing a new approach KNN-LSTM. This method novelty lies on the implementation of a KNN with weighted rank-exponent method integrated with 2-layer LSTM, leading to improved performance when compared to traditional and DL approaches.

Alternatively, Zhao et al. (2020) proposed a RNN hybrid model through the combination of Gated Recurrent Unit (GRU) with Graph Convolutional Networks (GCN), creating T-GCN architecture. The GCN model is responsible for capturing the topological structure of the road network for the modelling of spatial dependencies, while GRU model captures the changes in data over time to model the temporal dependencies. Experimental results reveal that T-GCN can outperform predictions of other traditional methods.

Wu et al. (2020) proposed the development of an attention mechanism integrated with a bi-directional LSTM architecture, ATT-LSTM. The attention mechanism dynamically learns the importance of each adjacent time series on the predicted series, retrieving the most important spatio-temporal dependencies. At the end, the output provided is processed through a normalized exponential function (softmax) and the predictions are obtained. When compared with other DL methods, the proposed one is able to improve computational efficiency, having lower prediction error, and consequently higher accuracy.

More recently, (Shi et al., 2021) experimented on integrated attention mechanism with a modified LSTM with recurrent skip connections, named APTN. The input is processed through a Recurrent Skip Neural Network and then spatial relationships go through an encoder. In the decoder, the temporal attention mechanism is applied to capture the dependencies from encoder hidden states across all time steps. When evaluated, the proposed model produced better results than most baseline models, mentioned above.

Finally, Zheng et al. (2019) proposed a multi-attention network (GMAN), designed as the integration of several attention mechanisms into an encoder-decoder architecture, being responsible for modelling the relationships between future and previous time steps. The attention mechanisms act between the encoder and the decoder, converting the historic variables to generate future representations.

Both the encoder and the decoder contain L ST-Attention blocks with residual connections, where each block is composed by spatial and temporal attention mechanisms. This proposed method also produced very strong results when tested on real data and it is able to make predictions into the far future.

Many techniques have been proposed in last years to address traffic prediction and different performance evaluation criteria have been selected. Next, a summary table is presented with the different models and metrics adopted by previous mentioned DL research.

Table 3.1 - Comparison of previous Literature on models and performance metrics adopted

Author	Title	Year	Models	Metrics
Ma et al.	Large-Scale Transportation Network Congestion Evolution Prediction Using Deep Learning Theory	2015	RNN-RBM BPNN SVM	Accuracy Sensitivity Specificity
Oliveira et al.	Computer network traffic prediction: a comparison between traditional and deep learning neural networks	2016	MLP-BP MLP-RP RNN SAE	MSE NRMSE
Song et al.	Traffic speed prediction under weekday using convolutional neural networks concepts	2017	CNN MLP	MAE RER
Wu et al.	A hybrid deep learning based traffic flow prediction method and its understanding	2018	DNN-BTF LASSO BPNN SAE DeepST StoS	MAE MRE RMSE
Liu et al.	DeepRTP: A Deep Spatio-Temporal Residual Network for Regional Traffic Prediction	2019	HisAve ARIMA SARIMA VAR DeepRTP	RMSE
Luo et al.	Spatiotemporal traffic flow prediction with KNN and LSTM	2019	ARIMA SVR WNN DBN-SVR LSTM KNN-LSTM	RMSE Predicting Accuracy
Zheng et al.	GMAN: A Graph Multi-Attention Network for Traffic Prediction	2019	ARIMA SVR FNN FC-LSTM STGCN DCRNN Graph WaveNet GMAN	MAE RMSE MAPE

Author	Title	Year	Models	Metrics
Zhao et al.	T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction	2020	HA ARIMA SVR GCN GRU T-GCN	MAE RMSE Accuracy R^2 Var
Wu at al.	A combined deep learning method with attention-based LSTM model for short-term traffic speed	2020	ATT-LSTM LSTM RNN LSTM-CNN CNN ANN	MAE MSE RMSE
Bao et al.		2021	ARIMA NN DBN	Accuracy MAPE
Shi et al.	A Spatial-Temporal Attention Approach for Traffic Prediction	2021	HA ARIMA VAR LSTM DA-RNN GeoMAN STGCN DCRNN Graph WaveNet ASTGCN APTN	RMSE MAE MedAE MASE MAPE

4. METHODOLOGY

In this section will be presented the Methodology adopted for the development of this work, starting with its theoretical explanation and followed by the description of the steps adopted for this specific work.

This thesis follows Cross Industry Standard Process for Data Mining (CRISP-DM) (Chapman et al., 2000), a widely adopted model that serves as base for data science processes, which is divided into 6 main phases organized in a cyclical manner, allowing moving back and forth between them, where the outcome of each one determines the next one. Figure 3 presents a visual representation of CRISP-DM.

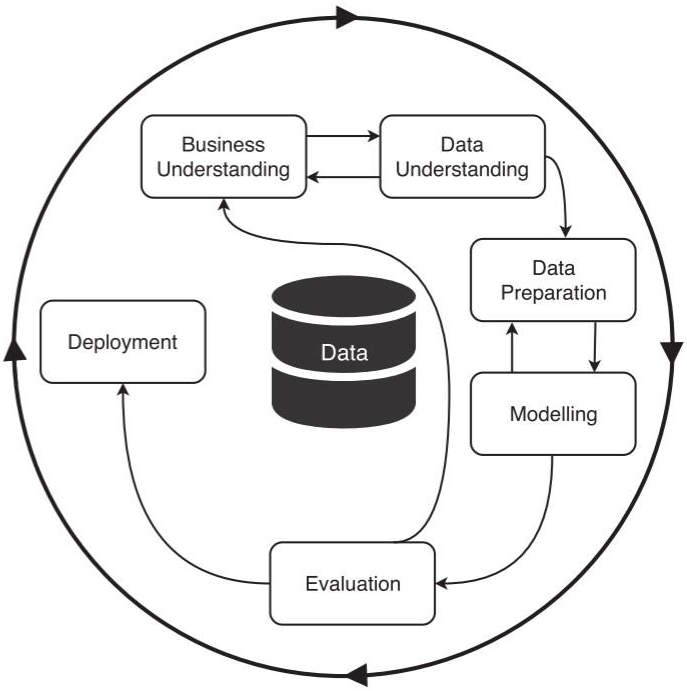

Figure 4.1 - CRISP-DM process model (Martinez-Plumed et al., 2021).

The first phase, **Business Understanding**, was majorly performed in Theoretical Background and Literature Review, but other aspects will be approached in this section. In the second phase, **Data Understanding**, statistical measures and visual representations were developed, easing the identification of useful patterns and relationships.

Data Preparation was the next phase, where data was transformed to better fit the models. This phase was the most time consuming one, accounting for about 80% of the project. In the fourth phase, **Modelling**, several models were developed and assessed.

In the fifth step, **Evaluation**, the results were evaluated, and the final model chosen. Since this work was developed for academic purposes only, **Deployment** phase was only composed of an analysis and discussion of the results obtained.

While the majority of the CRISP-DM phases will be performed in Methodology section of this work, the last one (Evaluation and Deployment) will be developed during Results and Discussion section.

4.1. BUSINESS UNDERSTANDING

This phase was executed in Theoretical Background and Literature Review sections of this thesis. Main objectives and company background were presented in Introduction. Data mining goals and the success criteria should also be presented in this phase.

Goal: develop machine learning models able to predict both future traffic and alerts, based on real data, from Waze app provided by OesteCIM, from previous events. It should be able to identify which factors positively or negatively impact traffic and lead to a specific alert to occur, and the importance of each one for the prediction.

Success criteria: in classification problem, the success criteria focused primarily on F1-score, to which the objective is to obtain a value as high as possible, i.e., as close to 1 as possible.

4.2. DATA UNDERSTANDING

4.2.1. Data sources

Two datasets were provided by OesteCIM regarding traffic and alerts, and two others were obtained from online sources, one regarding weather conditions and the other of Covid-19 time interval. The traffic dataset contains information regarding traffic characteristics, period and region of occurrence, and details about the user report. The alerts dataset is quite similar, also containing information about the period and region of incident occurrence, and the type of alert reported by the user. The weather dataset contains information like weather type, precipitation, temperature, among others.

Table 4.1 - Traffic dataset variables description

Variable	Description	Variable type
creationdate	Date when report was created	Time
creator	Report's creator identification	Numerical
editdate	Date when report was edited	Time
reportlevel	Traffic congestion level (1=free flow, 5=blocked)	Categorical
street	Street where traffic occurred	Categorical
city	City where traffic occurred	Categorical
delay	Delay of jam compared to free flow speed, in seconds (-1 means blocked road)	Numerical
length	Jam length, in meters	Numerical
startnode	Nearest junction/street/city to jam start	Categorical

Variable	Description	Variable type
endnode	Nearest junction/street/city to jam end	Categorical
description	Jam description	Categorical
shape	Shape of the jam	Categorical
parish	Parish where traffic occurred	Categorical
county	County where traffic occurred	Categorical
district	District where traffic occurred	Categorical
dicofre	Parish's unique identified	Numerical
latitude	Latitude of the place where traffic occurred	Numerical
longitude	Longitude of the place where traffic occurred	Numerical
speedKMH	Current average speed on jammed segments, in kilometers per hour	Numerical

After an initial analysis of the provided traffic data (Table 4.1), the features 'creator', 'editdate', 'description' and 'creationdate', were removed before any further steps, since they did not contain any information, only missing values. After this step, the dataset was reduced to 31 columns, 1 target and 30 independent features. A new variable was created to store the minute of the traffic occurrence, since the other date details were already stored on its own.

Some inconsistencies were found in 'parish' and 'dicofre'. Regarding the first variable, there were two different ways of referencing to the same parish, one in full "União das freguesias…" while the other with abbreviations "U.F. …". To keep the consistency, the first option was replaced with the second, so every parish was only presented in one way. Regarding the second variable, parishes Ventosa and Vimeiro had two codes associated, so the one with fewer records was replaced by the other.

In 'city', 'startnode' and 'endnode' some differences in punctuation were found, the usage of parentheses, commas and square brackets to separate information. The first one was the most widely used in the dataset, so the others were replaced by it. The additional information presented after the punctuation was not removed because there were several places with the same name, but that didn't refer to the same location, for interpretation and analysis purposes, the choice was made to keep the information within parenthesis.

In these features, some places also had two different representations, so they were joined to maintain consistency. Some values didn't fit what was expected for these variables, so they were replaced as null values for future imputation.

At the end, duplicate records were removed by 'uuid' column, since it should represent a unique jam ID, i.e., it should be unique for every record in the dataset, which wasn't the case. Two new variables were created, one for the minute of the alert occurrence and another for the day when the occurrence took place.

Table 4.2 - Alerts dataset variables description

Variable	Description	Variable type
Date	Date when report was published	Time
Road/Street	Street where event occurred	Categorical
City	City where event occurred	Categorical
Type pt	Event type, in Portuguese	Categorical
Subtype pt	Event subtype, depends on Tipo, in Portuguese	Categorical
dicofre	Parish's unique identified	Numerical
parish	Parish where event occurred	Categorical
county	County where event occurred	Categorical
district	District where event occurred	Categorical
latitude	Latitude of the place where event occurred	Numerical
longitude	Longitude of the place where event occurred	Numerical

In Table 4.2, is possible to see some features of alerts dataset. In it inconsistencies regarding parishes names and codes were also found and were solved the same way as for the traffic dataset. Other inconsistencies were found in 'Type' and 'Subtype' variables, both in English and Portuguese variants. For the first variable, there were two *Road closed* nomenclatures, which were then merged together. For the alert subtype, some of them appeared twice, one with capital letters. As done for the other variable, the two nomenclatures were joined.

Table 4.3 - Weather dataset variables description

Variable	Description	Variable type
Location	Location: Concelho	Categorical
Time	Date and time	Time
temperature_2m (Â°C)	Air temperature at 2 meters above ground, in Celsius	Numerical
relative_humidity_2m (%)	Relative humidity at 2 meters above ground, in percentage	Numerical
dew_point_2m (Â°C)	Dew point temperature at 2 meters above ground, in Celsius	Numerical
precipitation (mm)	Total precipitation (rain, showers and snow), in millimeters	Numerical
rain (mm)	Only liquid precipitation including local showers and rain from large scale systems, in millimeters	Numerical
snowfall (cm)	Snowfall amount, in centimeters	Numerical

Variable	Description	Variable type
weather_code (wmo code)	Weather condition as a numeric code, (WMO weather interpretation codes[5])	Categorical
cloud_cover (%)	Total cloud cover as an area fraction	Numerical
cloud_cover_low (%)	Low level clouds and fog up to 2 km altitude, in percentage	Numerical
wind_speed_10m (km/h)	Wind speed at 10 meters above ground, in kilometers per hour	Numerical
terrestrial_radiation_instant (W/mÂ²)	Solar radiation at the instant time, in Watt per square meter	Numerical

This dataset (Table 4.3) was obtained from an outside source, Open-Meteo[6], with records from December 2018 to July 2023, for all 14 counties present in the traffic and alerts datasets. The data presents an hourly range for better association with previous datasets.

Regarding the data preprocessing tasks, the one performed was the addition of new variables since no incoherencies and missing values were found. The created features relate to the date, storing the day of the month, month, year and hour of the registered weather conditions.

The covid dataset is only composed of two columns: 'Date' and 'Covid'. The first one contains dates from December 2018 to July 2023, with a daily range. The second is a binary variable that informs if a specific date corresponds or not to covid period. This dataset was manually created, using the same timestamp from traffic and alerts datasets, and the covid period was retrieved from online sources (eportugal).

4.2.2. Exploratory data analysis

After importing the data and dealing with its major inconsistencies, it is analyzed for an initial understanding and familiarization with its main characteristics, relationships and patterns. Next, the major insights obtained from Exploratory Data Analysis (EDA) are presented.

From an initial analysis of the data, it is possible to conclude that traffic with 21% to 40% of free low speed is the most common category (31.46%), while blocked road accounts for only 7.32% of all traffic occurrences reported. The low percentage of the category with higher proportion of free flow speed translates that the majority of reported traffic impacts significantly populational mobility dynamics. This can be seen in Figure 4.

[5] National Centers for Environmental Information, *WMO CODE TABLE 4677* available at https://www.nodc.noaa.gov/archive/arc0021/0002199/1.1/data/0-data/HTML/WMO-CODE/WMO4677.HTM

[6] Open-Meteo, *Historical Weather API* available at https://open-meteo.com/en/docs/historical-weather-api

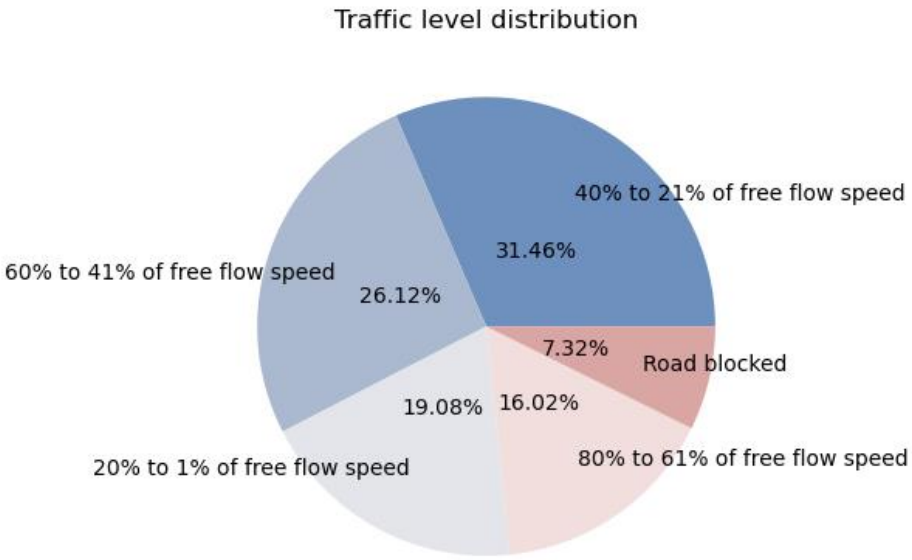

Figure 4.2 - Pie Chart of traffic level

Regarding alerts type *weather hazard* represents almost half of the reports (45.44%), while *accidents* represent only 3.18% of alerts. It would be expected that *jam* would account for a higher percentage of alerts reported, since it is one of the most recurrent road events. On the contrary, *road closed* revealed a very significant proportion, what was not expected. This can be seen in Figure 5.

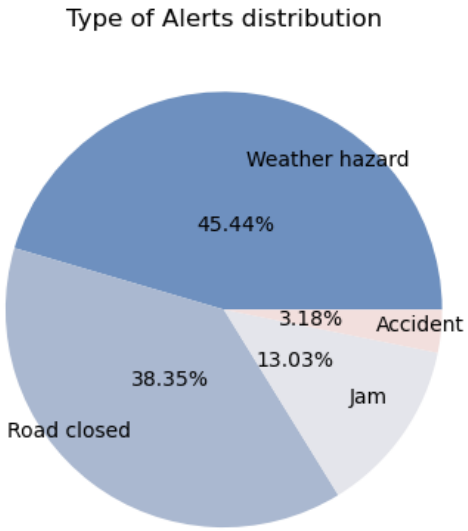

Figure 4.3 - Pie Chart of type of alert

Another important insight that can be taken from the data is the targets evolution during the years. By the analysis of Figure 6 is possible to conclude that traffic level shows a smaller number of reports in 2023, what is justified by the fact that the data stops in July of that year, but even so more records were expected, since more than half of the year is present in the dataset. From 2020 to 2023 less records were expected, especially of categories with more altered traffic flow, when compared with previous years, since covid-19 mobility restrictions were implemented.

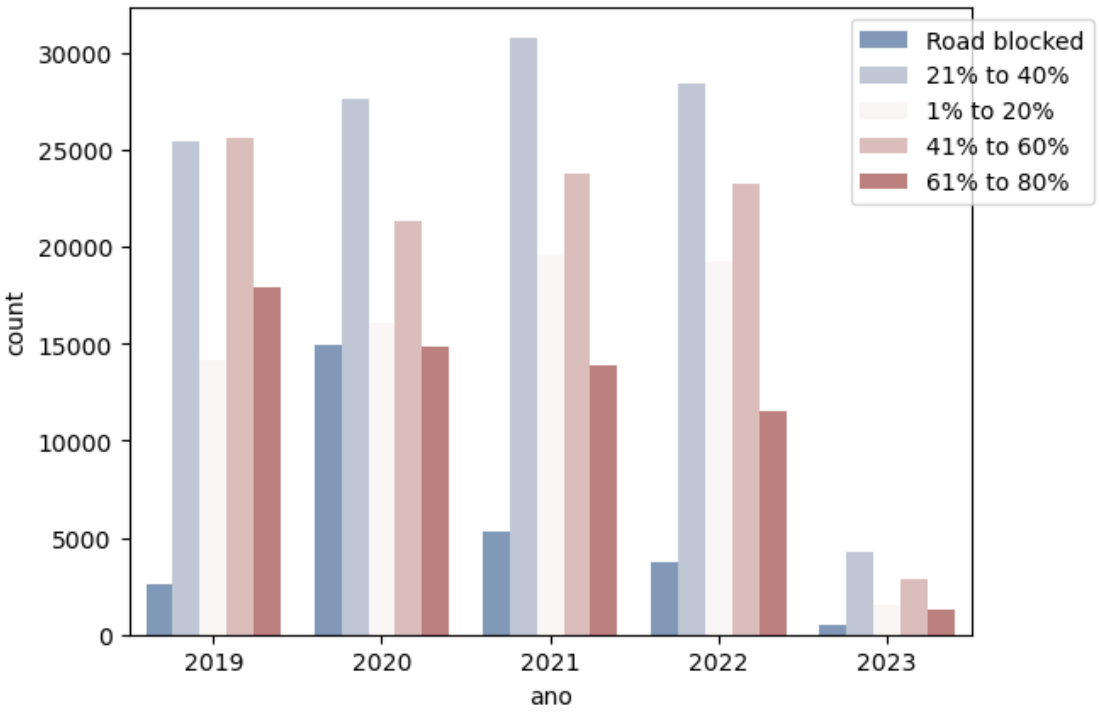

Figure 4.4 – Count plot of traffic level over the years

For the alerts data, Figure 7 shows that the reports of closed roads increased very significantly in 2022 and 2023, more than doubling the value of previous years. In 2018 the number of reports was very low, since dataset only contains values for December of that year. The most frequent category in the majority of the time period is *weather hazard*, confirming what was observed in the pie plot.

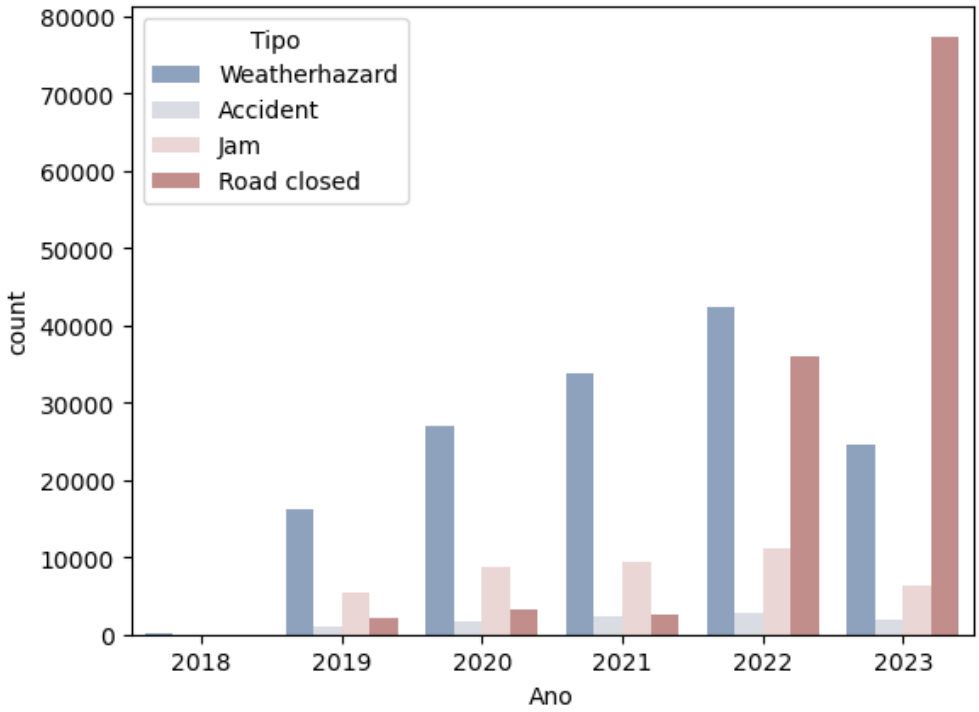

Figure 4.5 – Count plot of type of alert over the years

As mentioned in Literature Review, when developing traffic and alerts visualizations, is important to have a spatio-temporal perspective. The two maps presented allow the identification of regions where traffic and alerts tend to occur more frequently, and also the identification of records that might present wrong information regarding location features.

In Figure 8, traffic occurs uniformly in Oeste region, but it possible to identify some areas where traffic events concentration is bigger than in other places. Regarding alerts occurrence by type it is notorious the difference between regions near main cities and less populated ones. In this map some reports have coordinates of an ocean location, which is wrong information, and will be dealt with in future steps.

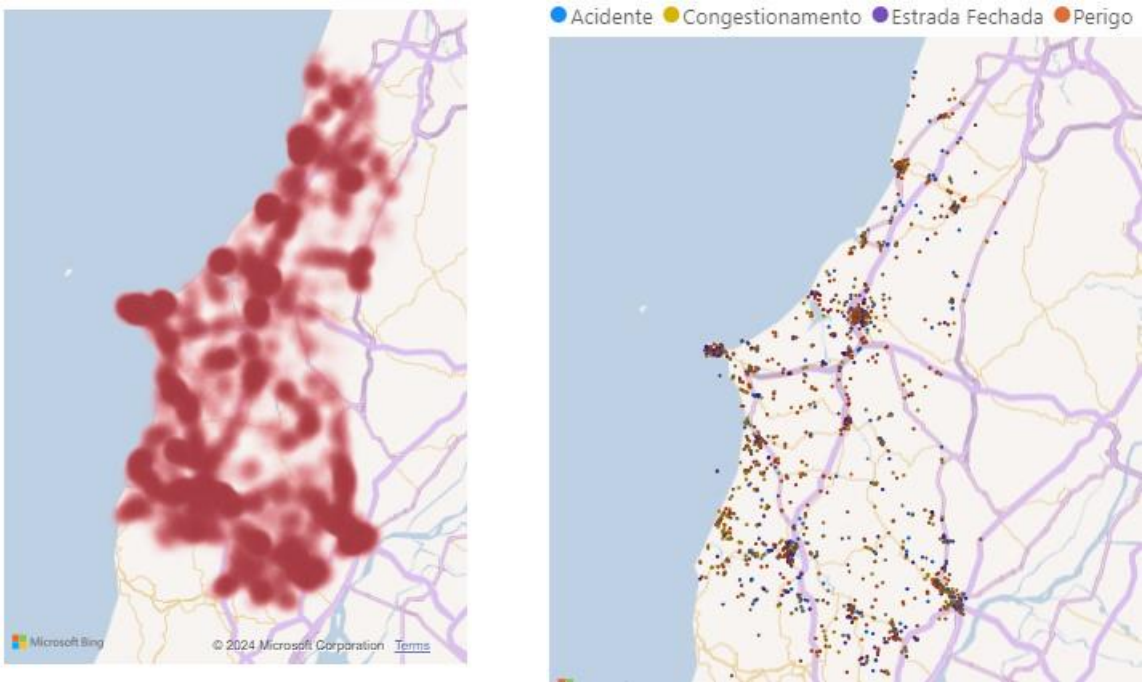

Figure 4.6 - Maps of traffic (left) and alerts (right) occurrence

4.3. DATA PREPARATION

After the analysis of the data and its understanding, data transformation is crucial for the development of effective and accurate models. In this step, raw data is modified into a format easily understood and used by machine learning algorithms. Several steps are performed until the data can be applied to the models, such as dealing with missing values and outliers, the creation and selection of features, the encoding of categorical variables and the data normalization for every feature to have the same range, so same initial importance to the model.

4.3.1. Data Partition

In order to later access the models performance, to perform model validation, it is necessary to split the original dataset into two or three subsets: Train, Validation and Test. The train data is used to create and develop the models. The validation data is then shown to the model, during its development and assessment in order to ensure that the model is able to predict unseen data as accurate as already seen one.

The test dataset is another way to assess the model, another subset of unseen data to further test model's ability to predict new data, but instead of being applied during model development and assessment, as the validation set, it is only applied after the model is chosen and completely developed. The different techniques applied are explained next.

4.3.1.1. Train/Test Split

It is the simplest method for model validation. The data is slipped into train and validation, based on a proportion, and then the model is trained with the train dataset. Lastly, the model is evaluated and the hyperparameters are tuned using the validation dataset. The process can be repeated multiple times and once it is completed the final model is assessed with the test dataset.

Even though this is a very basic method, it can outperform more complex data partition techniques. For that reason, this technique was adopted to split the original dataset into training and test before performing data preparation. This was performed in order to prevent information leakage from the testing set into the training set.

This technique was also adopted for the train/validation slip for the DL models, since it revealed more reliable and accurate final models, when compared to other techniques tested, for example K-fold. The split was performed to the train dataset account for 70% of the data.

4.3.1.2. K-Fold Cross Validation

In this technique, the dataset is divided into k subsets and consequently the model is trained and evaluated k times, using a different fold as the validation set each time. The model is trained using k-1 subsets of the original data, and then it is assessed using the remaining fold that was not used in the training process. This process is repeated k times, so every subset is used for validation. At the end, the final result is the average performance score from all repetitions.

There are other variations of this technique, such as the stratified k-fold, which ensures that each fold has the same proportion of observations for each target class as the original dataset, and the repeated k-fold, where the k-fold process explained is repeated n-times.

This technique and its variations were adopted for the ML models since its application led to higher quality predictions, revealing smaller difference between prediction quality for train

and validation datasets, reducing the overfitting or underfitting that might occur. The number of splits adopted was 5 and the number of repetitions was 3.

The next Data Preparation steps were only initially applied to the train dataset. Incoherencies and errors found in validation and tests datasets were solved using the same techniques and models developed for the train dataset.

4.3.2. Data Imputation (Missing Values)

Missing values can have a huge impact on models quality and predictive effectiveness, so it is extremely important to handle them accordingly. Next will be presented the approach applied in this project for traffic and alerts datasets, explaining the algorithm behind it and how the imputation is performed. Regarding the weather and covid datasets, they did not present any type of missing values, so they won't be mentioned further.

In the traffic dataset only, categorical variables presented missing values. In order to apply an imputer to them, its encoding was necessary. For that purpose, Ordinal Encoder was applied to features 'street', 'city', 'sartnode', 'endnode', 'parish, 'county, 'district' and 'dicofre' for them to only contain numerical values. The purpose of this step is only to transform the data so it can be applied to KNN imputer (Batista et al., 2002), which requires data to be numeric, so this encoder was only chosen due to its simplicity and fast application.

Ordinal Encoder receives an array of integers or strings, converting the features to ordinal integers, resulting in a column of integers ranging from 0 to number categories-1 for each feature provided.

Since all features that contained missing values were location related ones, it made sense to also integrate latitude and longitude features for the imputation, so they can be also considered when determining the value to be inserted. The imputer chosen was, as mentioned previously, KNN imputer, which is a method based on k-Nearest Neighbours.

For each sample, the missing value is imputed using the mean value from the n closest neighbours found in the training set. This imputer retains the most data when compared with other techniques, such as removing rows or columns with missing values. Also, it preserves the underlying relationships of the data, considering feature similarities between the data points. In order to impute the missing values, this method estimates the values based on characteristics of similar records, i.e., based on the characteristics of the nearest neighbours in a n-dimensional space.

After the imputation was preformed, the encoding was reversed, having again the categorical variables containing strings instead of only numerical values. This step was only performed so, further in this project, other types of encoding can be applied and tested for better results.

For the alerts dataset the procedure was very similar, applying both Ordinal Encoder and KNN imputer to the categorical features 'Subtype', 'Road/Street', 'City', 'Subtype pt', 'dicofre',

'parish', 'county' and 'district'. The biggest difference in this dataset was the fact that it also presented missing values in numerical variables 'latitude' and 'longitude'. Despite that, they were added to the array of categorical variables and used in the imputation process. Also, 'Tipo' and 'Tipo pt' were also used for the imputation since they are also categorical variables that are highly correlated with the subtype features.

4.3.3. Outlier detection

Another important step in data preprocessing is the identification and outlier processing. An outlier is a data point that is distant from the remaining ones and behaves differently from what is expected. These data values can have a huge impact on data statistics and machine learning techniques, since they can represent incorrect data and skew data from a normal distribution. There are several methods that can be applied to outlier detection going from Interquartile Range (IQR), and visualization methods such as boxplots and histograms.

In this project, 4 different approaches were used: boxplots, manual filtering, IQR method and z-score. Next, the procedure applied to traffic and alerts dataset is explained. It is worth mentioning that the weather dataset also presented outliers, but after further research and ponderation, the decision of not removing them was made, since they represent true values and was believed that its removal would lead to important information loss.

It is important to notice that, for quality purposes, no more than 5% of the initial dataset must be removed in this phase, in order to ensure to not fall into important information loss.

Boxplots were initially used to access which numerical features presented outliers, leading to the conclusion that 6 presented them: 'speed', 'delay', 'length', 'latitude', 'longitude' and 'speedKMH'.

The manual filtering method was performed by analyzing the boxplot of each feature with outliers and then choosing the interval to which the values must be inserted into. This method suggested removing 0.16% of the original data, what does not represent a big amount.

The IQR method starts with the computation of the inter quantile range, i.e., the difference between the 75^{th} percentile (q3) and the 25^{th} percentile (q1), and then the lower and upper bound are calculated.

$$\begin{aligned} Lower\ Bound &= q1 - 1.5 * IQR \\ Upper\ Bound &= q3 + 1.5 * IQR \end{aligned} \tag{4.1}$$

However, the multiplication of IQR by 1.5 would lead to the removal of a big percentage of data, so this value was replaced by 3, getting the major outliers, instead of the minor ones. After the computations are performed, only the data points with values between the lower and upper bounds are maintained, while the others are then considered outliers and removed

from the dataset. For this technique, the percentage of data to be removed was 10.51%, which surpasses the threshold.

The final method applied, z-score, indicates how many standard deviations a data point is above or below the mean of the dataset. The higher the score, the more distant the point is from the others. To compute the z-score, the following equation is applied:

$$z\text{-}score = \frac{x-\mu}{\sigma} \quad (4.2)$$

where x is the current value, μ represents the mean of the feature to where the current value belongs to, and σ is the standard deviation of that same feature.

Finally, in this method, a threshold must be defined to determine how extreme a data point has to be for it to be considered an outlier. The usual adopted threshold is 3, but for this project and this dataset, it had to be settled as 5, so the amount of removed data would not surpass 5%.

After comparing the three methods presented, the one chosen was z-score, where observations with more than 5 standard deviations from the mean were discarded, leading to a removal of 1.98% of original data.

For the alerts dataset, from the analysis of the boxplots, only two features presented outliers: 'latitude' and 'longitude'. Regarding the manual filtering, the upper and lower interval bounds were defined by looking at the boxplots of each feature. After its application 9.79% of the original data would be removed.

In the IQR range method, the interquartile range was also multiplied by 3 to obtain the major outliers. From this method, 19.30% of the observations would be removed, which represents too much information loss.

The final method, z-score, had to be settled with a threshold of 3.2 to ensure that a significant number of outliers were removed, but without removing more than 5% of the original observations.

After the analysis and comparison of the three methods, the one chosen for this dataset was also z-score, where the values with more than 3.2 standard deviations above or below the mean were discarded, leading to a removal of 1.09% of the original observations.

The next phases of Data Preparation will be performed on the final datasets, the ones already combined together and to be applied to the predictive models. Still two datasets will be used, one to predict the traffic and another to predict the alerts and incidents.

4.3.4. Feature Engineering

This phase is where new features, which aren't already in the dataset, are generated. Feature Engineering eases ML process and can lead to increased algorithm's predictive power since important features might be created. For both main datasets, traffic and alerts, new features were created, and will be explained next.

For the main purpose of this work, traffic prediction, three new features were created: 'Part of day', 'Season' and 'alert'. During the day, traffic congestion occurs at different levels, especially due to the number of people travelling at a specific time. Usually, the most congested periods of the day are early morning, before working hours, and in the afternoon, after working hours.

For that reason, it would be important to understand at which time of day the traffic occurs most of the time. Since that feature was not directly present in the dataset, it was created. It was generated following the expression below, dividing the day according to working hours:

$$\text{If 'hour'} \in [0;5[\cup [21;23] : \text{'}Part\ of\ day\text{'} = \text{'}night\text{'}$$
$$\text{If '}hour\text{'} \in [5;12[: \text{'}Part\ of\ day\text{'} = \text{'}morning\text{'}$$
$$\text{If '}hour\text{'} \in [12;17[: \text{'}Part\ of\ day\text{'} = \text{'}afternoon\text{'}$$
$$\text{If '}hour\text{'} \in [17;21[: \text{'}Part\ of\ day\text{'} = \text{'}evening\text{'}$$

Traffic is also usually impacted by the season of the year, since in Winter the weather usually leads to worst driving conditions, impacting traffic speed. For that purpose, a feature containing that information was created using the following expression:

$$\text{If 'month'} \in [1;3] : \text{'}Season\text{'} = \text{'Winter'}$$
$$\text{If 'month'} \in [4;6] : \text{'Season'} = \text{'}Spring\text{'}$$
$$\text{If '}month\text{'} \in [7;9] : \text{'}Season\text{'} = \text{'}Summer\text{'}$$
$$\text{If '}month\text{'} \in [10;12] : \text{'}Season\text{'} = \text{'}Autumn\text{'}$$

Another feature that was added to this dataset was regarding the association of an alert to the events. For that purpose, the alerts dataset was merged to the traffic one based on the location and time features: 'street', 'city', 'parish', 'dicofre', 'county', 'district', 'year', 'month', 'day_ext' and 'hour'. It is important to notice that before joining the two datasets, it was important to make sure that the merging features had the same type and equal values represented in the same way (especially in the location features). Also, some of the traffic events had more than one alert associated, so the duplicates were removed by 'uuid' variable, since the important is if any alert occurred at the same time or not.

Not all traffic events reported had an alert associated, so some rows presented missing values for the features 'Tipo pt', 'Subtipo pt' and 'reliability', the ones obtained from alerts dataset. This was expected and wanted, since the new variable should reference if an alert is associated or not to the traffic reported. The feature was developed following the next expression:

$$\text{If } `Type' \text{ is not Null} : \ `alert' = 1$$
$$\text{If } `Type' \text{ is Null} : `alert' = 0$$

After the creation of the two new features, the weather and covid datasets were also added to the data. The first one was merged by 'county', 'year', 'month', 'day_ext' and 'hour' while the second one was only merged by the temporal variables. The feature 'Weather code', obtained from the weather dataset, was transformed to store the weather type instead of codes.

The new features were added to the data in order to understand if they have any impact on how the model is able to predict traffic and if they improve its performance. At the end duplicated columns were removed from the data since they did not bring any additional value to the future models. Finally, after all new features were added and preprocessed, the traffic data ended up with 38 columns and 363112 records.

The process performed for alerts dataset was very similar to the one presented for the traffic prediction. The major difference is regarding the new features created, where only 'Part of day' and 'Season' were added, since here all records represent a reported alert. At the end the dataset was composed of 28 columns and 313279 records.

4.3.5. Categorical variables encoding

Most machine learning algorithms require numerical data and are not prepared to deal with features containing strings. For that reason, the categorical features of the datasets that contained strings had to be put through an encoding process, in order to transform them into numerical values.

The majority of the categorical features were nominal, meaning that their values are not ordered, but some of them presented very high cardinalities, making it not fit to some of encoding methods. The few ordinal features present in the data are not classified as intervals since the magnitude between values is not the same, so Ordinal and Label encoding were excluded from possible solutions.

One-hot encoder is a widely adopted technique applied to nominal features. It transforms each category into a new binary column, generating as many columns as categories of each original feature. Despite that it creates equally important new features, high cardinality can lead to the development of a significant number of new columns, leading to the curse of dimensionality. Since the dataset contains features with high cardinality, the adoption of the method would lead to an exaggerated number of columns in the final dataset, so it was only applied to the features with cardinality lower than 5.

Target encoder is another technique that was tested during the development of this work. It is a Bayesian technique that uses the target variable to encode the categorical features of the data. It basically checks the target values associated with each category and computes the mean of the target for each category. It can be adapted to different problem types, which in this case is multiclass classification.

The target feature, in traffic and alerts dataset, is multiclass categorical with strings as values. Since, as mentioned before, most machine learning models require data to be numerical, this variable had to be encoded. For the ML algorithms tested, Ordinal encoder was used, a simple technique that converts features values into ordinal integers, containing values from 0 to the number of categories-1 per feature. For the DL algorithms, one-hot encoder was applied to the target features.

4.3.6. Data Normalization

Before deciding which feature to include or not in the final dataset that is going to be applied to the prediction models, is necessary to scale them. This step is crucial since features with bigger value ranges would have a greater impact on the models than the ones with smaller value range, leading to biased results. For that reason, a scaling technique is applied to the features to ensure they have the same range, each one is equally important and are easier to process by ML algorithms.

The technique applied in this work was normalization, through Min-Max scaler algorithm, where each feature has a minimum value of 0 and a maximum value of 1. This transformation is performed through the application of the following equation:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{4.3}$$

This approach is widely adopted when data's distribution is not known or is not Normal distributed.

4.3.7. Feature Selection

The principal objective of this phase is to remove redundant features from the dataset, in order to build simpler models whose predictive processes can be easily understood, improve models performance and reduce the computational costs associated to modelling.

There are three major types of feature selection techniques: filter, wrapper and embedded methods. The first category uses statistical techniques to evaluate the relationships between each feature and the target. In wrapper methods a predictive model is applied to evaluate the combination of features and a score is assigned to that same combination based on model's accuracy. Embedded methods are able to identify which features contribute more to the accuracy of the model during its creation.

In this work several techniques are applied: Spearman correlation, Chi-square test, ANOVA, Recursive Feature Elimination (RFE), Least Absolute Shrinkage and Selection Operator (LASSO), Random Forest feature importance based on mean decrease in impurity and Decision Trees for Feature Importance. Next the techniques are briefly explained, and the results obtained for each one are presented.

4.3.7.1. Filter Methods

Firstly, the variance of each numerical feature was checked to verify if they represent relevant information. 'Snowfall' feature showed to have the same value for each observation, having a variance of 0, so it was immediately removed from the dataset, since it did not bring any relevant information to future models.

Then, spearman correlation was used to access the relationships between numerical features and the target. This correlation coefficient is a nonparametric measure that reflects the statistical dependence between the rankings of two variables. Next are presented the results of high correlated features:

- length and reportlevel (target) have a correlation of -0.79 – traffic prediction
- Temperature and Humidity have a correlation of -0.51 – both predictions
- Temperature and Dew have a correlation of 0.62 – both predictions
- Temperature and Solar radiation have a correlation of 0.61 – both predictions
- Humidity and Solar radiation have a correlation of -0.56 – both predictions
- Humidity and Cloud cover have a correlation of 0.45 – both predictions
- Precipitation and Rain have a correlation of 1.0 – both predictions
- Cloud cover and Cloud cover low have a correlation of 0.84 – both predictions

The majority of the features with high correlations are weather related. Temperature and Humidity present a negative correlation, since the higher the temperature, usually, less humidity is in the air. Also, temperature is positively correlated with solar radiation since the higher the radiation emitted by the sun that reaches the surface of the Earth, the higher will be the temperature. Precipitation and rain have a correlation of 1, since the first feature is computed by summing rain and snowfall.

For the categorical variables, the Chi-Square test was applied to determine whether there is any kind of association between categorical features and the target, if they are independent or related. Features that present a significant dependency on the target are considered important while the others might be removed from the analysis. In this project, this technique considered all categorical features to be important, for both traffic and alerts predictions.

Still in Filter methods, ANOVA technique was applied to numerical features. It is considered a statistical method, which is used to check if the means of two or more groups are significantly different from each other. When applied to feature selection, this technique returns the k most relevant features, the ones with higher values for the statistical test. In this work, this

method suggested the usage of 6 numerical features, for both traffic and alerts predictions, as shown in Table 4.4.

Table 4.4 - ANOVA selected features

Traffic prediction	Alerts prediction
latitude	Hour
longitude	Minute
hour	Temperature
Humidity	Wind speed
Cloud cover	Solar radiation
Solar radiation	Cloud cover low

4.3.7.2. Wrapper Methods

In this project, only RFE was applied. The algorithm starts by fitting the predictive model to all predictors and then each one is ranked based on its importance to the model. At each iteration of feature selection, the top ranked predictors are retained, the model is refit, and the performance is assessed. The group of predictors with the best performance at the end is used to fit the final model.

After looking for the best optimal solution of this technique, it concluded that, for traffic prediction, 6 features might be retained, and, for alerts prediction, 8 features should be kept. The following table shows the features returned as important to this technique, as presented in Table 4.5.

Table 4.5 - RFE selected features

Traffic prediction	Alerts prediction
latitude	Hour
hour	Temperature
Temperature	Humidity
Humidity	Dew
Dew	Precipitation
Rain	Rain
	Wind speed
	Solar radiation

4.3.7.3. Embedded Methods

The first applied technique was LASSO, which is a regularized linear regression that includes an L1 penalty. This method regularizes the model parameters by shrinking the regression coefficients. A penalty is added to the residual sum of squares, which is posteriorly multiplied by the regulation parameter λ. For traffic prediction, 7 features were kept, and for alerts prediction 5 features were picked to be retained, as shown in Table 4.6.

Table 4.6 - LASSO selected features

Traffic prediction	Alerts prediction
latitude	Hour
hour	Temperature
Humidity	Humidity
Dew	Dew
Precipitation	Solar radiation
Wind speed	
Solar radiation	

The next methods were applied to categorical features, while the ones presented previously (for wrapper methods and the first in embedded methods) were applied to the numerical ones. The first was Random Forest feature importance based on mean decrease in impurity. The more a feature decreases the impurity, the more important the feature is. For alerts prediction, 21 features were determined to be important, while for traffic prediction, 32 features were kept for this technique, as can be seen in Table 4.7.

Table 4.7 - Random Forest Feature Importance based on mean decrease in impurity selected features

Traffic prediction	Alerts prediction
month (5 features)	Day (4 features)
weekday (5 features)	Year (4 features)
day_ext (5 features)	Road/Street (4 features)
Year (5 features)	city (4 features)
Weather code (5 features)	parish (4 features)
street (5 features)	Part of day_night
Part of day_evening	
Part of day_morning	

The last method applied was Decision Trees for Feature Importance. As for the previous technique, the model calculates the feature importance, so the best performing features are as close to the root of the tree as possible. The results were quite similar to the ones obtained with Random Forest, only a few features were labeled differently, not modifying the final decisions, presented in Table 4.8.

Table 4.8 - Decision Tress for Feature Importance selected features

Traffic prediction	**Alerts prediction**
month (5 features)	Nº Month (4 features)
weekday (5 features)	Day (4 features)
day_ext (5 features)	Road/Street (4 features)
Year (5 features)	City (4 features)
Weather code (5 features)	Part of day_night
street (5 features)	
Part of day_evening	
Part of day_morning	

After applying the feature selection techniques, was decided to remove from traffic prediction the features containing information of that specific traffic event and that cannot be obtained without it occurring, as for example, 'delay', 'shape', 'length' and 'speedKMH'. For alerts prediction 'reliability' was also removed since it is only known after the incident occurred.

After the application and analysis of all applied feature selection techniques, the traffic prediction dataset was kept with 28 features, while the alerts prediction dataset had 21 features after dropping non important ones. The features were selected by summarizing the results of the tested feature selection methods and then keeping the ones that were referred to be important for the majority of the methods.

4.4. MODELLING

After data preparation, the datasets are fit to be used to create the predictive models. Several ones will be developed, and the best one will be selected based on performance metrics. Several procedures were used in order to find the better model, by testing different class imbalance handling methods and checking different validation processes.

As mentioned before, this work faces multiclass classification problems, so the targets assume more than 2 classes, the level of traffic and the type of alert. The next step was to develop the predictive models, perform hyperparameter tuning and test the models against the validation

and test datasets. Next, both ML and DL approaches adopted will be explained, but firstly techniques to deal with imbalanced data will be presented.

4.4.1. Class imbalance

Real-word machine learning problems generate, the majority of the time, imbalanced class distribution. Classes don't present an equal distribution, ones presenting a higher number of records than the others. This is faced as a problem since many ML algorithms assume an equal distribution for each class, ignoring the ones with smaller observations. Having a balanced dataset makes the model training process much easier preventing bias towards classes with bigger number of observations.

To deal with this issue, in this project Synthetic Minority Over-sampling Technique (SMOTE) combined with Tomek Links was used. The first technique relates to oversampling, duplicating random examples from the minority classes. This is performed by selecting an example from the minority class and by looking for the k nearest neighbors of that example and choosing one at random. Then, a synthetic example is created at a randomly selected point between the two examples in the feature space.

The second technique refers to undersampling and, instead of looking for the k nearest neighbors, this algorithm selects pairs of observations that are at the same time each other's nearest neighbor but belong to different classes. The observation that belongs to the bigger class is removed.

This technique was adopted, firstly due to better results obtained when compared with other imbalanced data techniques, but also due to the fact that, despite significantly increasing the number of records, it is able to restrict that increase and better differentiate the classes. The next tables present the different classes distribution before and after the application of SMOTE-Tomek.

Table 4.9 - Number of observations per target class before and after applying SMOTE-Tomek for traffic dataset

Target Class	Number of observations Before SMOTE-Tomek	Number of observations After SMOTE-Tomek
1 (80% to 61% of free flow speed)	38937	81384
2 (60% to 41% of free flow speed)	66446	67929
3 (40% to 21% of free flow speed)	81384	65682
4 (20% to 1% of free flow speed)	48429	72712
5 (road blocked)	18982	81314

Table 4.10 - Number of observations per target class before and after applying SMOTE-Tomek for alerts dataset

Target Class	Number of observations Before SMOTE-Tomek	Number of observations After SMOTE-Tomek
0 (Accident)	6976	99729
1 (Jam)	28528	99366
2 (Road closed)	84062	99705
3 (Weather hazard)	99729	99369

4.4.2. Models

Several algorithms were tested, ML and DL ones, but only a few were selected as potential final models. The models chosen did not differ regarding the type of prediction, traffic or alerts. Regarding the ML models, the more accurate ones were Neural Networks, Decision Trees and some ensemble models (Bagging, Random Forests and Gradient Boosting). Stacking was also implemented but did not provide better results than the ones obtained with individual models.

In DL models, surprisingly, less complex models were able to outperform complex ones, such as LSTM and GRU. Multilayer Perceptron and VGG were algorithms able to better predict both traffic and alerts.

The models were initially settled with default parameters, in order to test if their most basic versions were able to make predictions with sufficient efficiency. Some variations within the parameters were applied to some of the models that weren't chosen, just to ensure that no mistake was made when discarding some of the initially tested models. After that, hyperparameter tuning was performed.

4.4.3. Hyperparameter Tuning

Different parameter settings can lead to different model performance, so, in the development of a prediction problem, hyperparameter tuning is performed to find the best possible combination of hyperparameters, i.e., the combinations that creates the model with best performance possible but at the same time reduces overfitting or underfitting.

This process can be very time consuming and require high computational effort, so over time several different techniques were developed to deal with this phase. All models mentioned in subsection 4.4.2 were used for hyperparameter tuning, since its performances could vary significantly with different parameters settings.

For the ML algorithms, the technique applied was GridSearchCV. This method performs an exhaustive search over the parameter grid provided, trying every combination and optimizing them by cross-validation grid-search over the parameter grid.

For the DL algorithms, another technique was applied, since we were dealing with a bigger search space, and the adoption of the same technique would reveal to be time consuming and require too much computational effort. For that reason, RandomSearch and Hyperband were used. The first technique randomly selects combinations of hyperparameters to evaluate.

The second one is an adaptive algorithm that allocated resources to different hyperparameter configurations based on their performance. It starts by running multiple configurations for a small number of epochs and eliminates the ones that have worse performance. Next a table is presented with the different hyperparameters that were tested for each ML and DL models.

Table 4.11 - Hyperparameter search space for GridSearchCV (ML models) and Hyperband (DL models)

Model	Hyperparameter	Values
Neural Networks	hidden_layer_sizes	{(10), (10,10), (10,10,10)}
	activation	{tanh, logistic}
	solver	{sgd, adam}
	learning_rate_init	{0.001, 0.01}
	learning_rate	{constant}, {invscaling}, {adaptive}
Decision Trees	criterion	{gini, entropy}
	splitter	{best, random}
	max_depth	{1, 2, 3, 4, 5, 10, 15, 20, 25, 30}
Bagging	n_estimators	{2, 5, 10, 20, 30, 50, 100, 150, 200}
	max_features	{0.1, 0.2, 0.4, 0.6, 0.8, 1.0}
Random Forest	n_estimators	{10, 20, 50, 100, 200, 300}
	max_features	{sqrt, log2}
Gradient Boosting	n_estimators	{10, 20, 50, 100, 200, 300}
	max_features	{2, sqrt, log2}
Multilayer Perceptron	number of hidden layers	{min_value:1, max_value:5, step:1}
	number of hidden units per layer	{min_value:1, max_value:10, step:1}
	activation function of hidden layers	{relu, sigmoid, softmax, softplus, softsign, tanh, selu, elu, exponential, leaky_relu, relu6, silu, hard_silu, gelu, hard_sigmoid, linear, mish, log_softmax}
	learning rate	{0.1, 0.01, 0.001}

Model	Hyperparameter	Values
CNN	number of convolutional layers	{min_value:1, max_value:5, step:1}
	activation function of convolutional layers	{relu, sigmoid, softmax, softplus, softsign, tanh, selu, elu, exponential, leaky_relu, relu6, silu, hard_silu, gelu, hard_sigmoid, linear, mish, log_softmax}
	number of hidden layers	{min_value:1, max_value:5, step:1}
	number of hidden units per layers	{min_value:1, max_value:10, step:1}
	activation function of hidden layers	{relu, sigmoid, softmax, softplus, softsign, tanh, selu, elu, exponential, leaky_relu, relu6, silu, hard_silu, gelu, hard_sigmoid, linear, mish, log_softmax}
	learning rate	{0.1, 0.01, 0.001}
VGG	number of convolutional layers	{min_value:1, max_value:5, step:1}
	activation function of convolutional layers	{relu, sigmoid, softmax, softplus, softsign, tanh, selu, elu, exponential, leaky_relu, relu6, silu, hard_silu, gelu, hard_sigmoid, linear, mish, log_softmax}
	number of hidden layers	{min_value:1, max_value:5, step:1}
	number of hidden units per layers	{min_value:1, max_value:10, step:1}
	activation function of hidden layers	{relu, sigmoid, softmax, softplus, softsign, tanh, selu, elu, exponential, leaky_relu, relu6, silu, hard_silu, gelu, hard_sigmoid, linear, mish, log_softmax}
	learning rate	{0.1, 0.01, 0.001}
	filters	{32,64,128}

5. RESULTS AND DISCUSSION

In this section of the work, the results of the developed models are presented as well as a discussion regarding their performance and suitability for the project. While presenting performance metrics of the best models, an explanation is also provided, understanding the importance of each feature in the predictions.

Previously in this document was mentioned that the f1-score was the performance metric used to select the best model. Although it was the most observed and analyzed metric, other previously mentioned metrics were also analyzed while selecting the best model.

5.1. TRAFFIC PREDICTION

Starting with traffic prediction, the ML models tested were Neural Networks, Decision Trees, Bagging, Random Forests and Gradient Boosting. All models were tested in their based format, only creating the instance without specifying additional parameters, with some variations on those same parameters and applying GridSearchCV to obtain the optimized models.

Before analyzing the performance scores of the tested models, is important to understand the characteristics of each one. Regarding the simple models of each method, the majority of the parameters were set to their default. The models returned by the hyperparameter tuning approach presented different characteristics.

The NN tuned model was returned with a hidden layer size set as (10,10), tanh as the activation function, the learning rate for the weight update set as constant, adam as the solver, and an initial learning rate set as 0.001. Regarding the DT tuned model, the criterion was set as gini, the max depth was 10 and the splitter method was set as best, choosing the best split at each node.

In ensemble models, only two parameters per method were tuned, as mentioned previously at Table 4.11. For Bagging tuned model, the number of features to each base estimator was set as 0.8 and the number of base estimators as 200. Regarding Random Forests tuned model, the number of features to consider when looking for the best split was set as log2, i.e., the logarithm of the total number of features, and the number of trees in the forest was set as 300.

Gradient Boosting tuned model was returned with 300 boosting stages to perform and the number of features to consider when looking for the best split set as log2. The MLP tuned model is composed of 1 hidden layer, elu as the activation function and a learning rate set as 0.001.

Different CNN were tested, with different numbers of convolutional layers and hidden units. A dropout rate of 0.2 was adopted for the different testing, and SGD optimizer with a learning rate set as 0.1. The approach providing better results consisted of 3 convolutional layers, with

32, 64 and 128 hidden units, respectively. Regarding its tuned version, the model returned consisted of 3 convolutional layers, with elu as the activation function for these layers, 1 pooling layer, with leaky_relu as the activation function and the SGD optimizer set with a learning rate of 0.001.

Finally, regarding VGG models, the approach was very similar to the one adopted for CNN. Also, with a dropout rate of 0.2 and a learning rate set as 0.1, the experiment leading to better performance scores consisted of 2 convolutional layers and an activation function set as sigmoid. VGG tuned model was returned presenting 2 convolutional layers, with softplus as the activation function, and 1 pooling layer. The optimizer adopted for these models was also SGD with the learning rate of 0.01.

Table 5.1 - f1-score for traffic prediction

Model	**F1 Score**	
	Train	Validation
Neural Networks	0.75	0.75
Decision Trees	0.99	0.77
Bagging	0.97	0.82
Random Forests	0.99	0.78
Gradient Boosting	0.78	0.77
MLP	0.80	0.67
CNN	0.78	0.70
VGG	0.83	0.70

By looking at the table above it is possible to conclude that some of the models were able to make high quality predictions of the training data, but when predicting the validation data, their performance was not as good, implying the presence of overfitting. In this type of situation, the best model is not only the one capable of better predicting both seen and unseen data, but also the one that predicts both with equivalent accuracy.

With that in mind, the majority of ML methods applied presented overfitting, predicting training data with much more quality than validation one, only Neural Networks and Gradient Boosting being able to predict both datasets with similar quality. Regarding DL models, all presented overfitting, so they were excluded from the list of possible final model.

Even though Neural Networks and Gradient Boosting models did not present the higher predictive quality for both train and validation datasets, they are able to reduce the difference between the quality of predictions within those datasets, reducing the presence of overfitting and underfitting.

After the analysis of all models performance, the final choice was Gradient Boosting model, even though not being the best at predicting train and/or validation data, it was the model able to predict both with higher quality and with lower presence of overfitting. Next, Table 5.2 shows the classification report of the chosen model.

Table 5.2 - Classification report on Gradient Boosting's traffic prediction

Traffic category	precision	recall	f1-score	support
1 (80% to 61% of free flow speed)	0.81	0.72	0.76	81384
2 (60% to 41% of free flow speed)	0.72	0.72	0.72	67929
3 (40% to 21% of free flow speed)	0.76	0.80	0.78	65682
4 (20% to 1% of free flow speed)	0.76	0.75	0.75	72712
5 (road blocked)	0.85	0.87	0.86	81314
accuracy			0.77	369021
macro avg	0.78	0.77	0.77	369021
weighted avg	0.78	0.77	0.78	369021

Gradient Boosting model has an overall precision and f1-score of 0.78 (78%) and recall of 0.77 (77%). Regarding the different types of traffic flow, the model is able to predict category 5 more accurately than the others, with a precision of 0.85, recall of 0.87 and f1-score of 0.86, while the category worse predicted was 2, with precision, recall and f1-score of 0.72. By taking a look at support, it is possible to conclude that the model better classifies traffic flows more frequent in the dataset.

Category 1 presents a significant difference between precision and recall scores, meaning that the model is incorrectly classifying category 1 observations as belonging to another category, but when an observation is classified in this category, 81% of the times the prediction is correct.

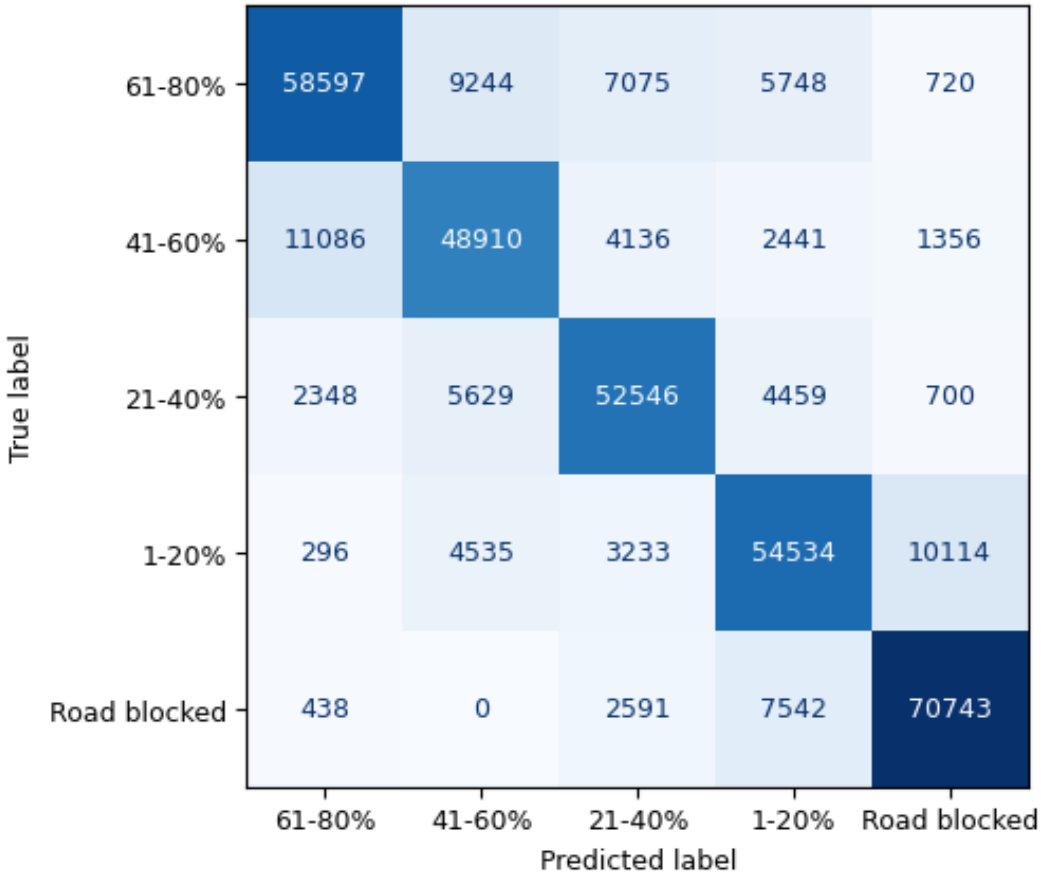

Figure 5.1 - Confusion matrix of traffic prediction

The confusion matrix for the traffic prediction, Figure 10, reinforces the results obtained with the classification report. It is possible to conclude that the more frequent mistake occurs when classifying observations related to 4*1-60% of free flow speed*, being often classified as *61-80% of free flow speed*, and vice-versa. Also, many mistakes occur when classified observations belonging to *1-20% of free flow speed*, which are often predicted as *road blocked*. But, in general when mistakes occur, the predicted category tends to correspond to the percentage of free flow immediately above or below the true category.

The interpretation of ML and DL models is often quite challenging due to their complexity. Explainable AI has been a powerful tool in this matter, easing the comprehension of the results and outputs generated by ML algorithms. In order to better understand what was developed in this study, SHapley Additive exPlanation (SHAP) was used. One visualization widely used to assess features impact on the model is beeswarm. It shows the impact that each feature has on the model in both directions (positive and negative), and also the impact by its color (with the scale on the right) and the volume of that impact (by the size, number of dots).

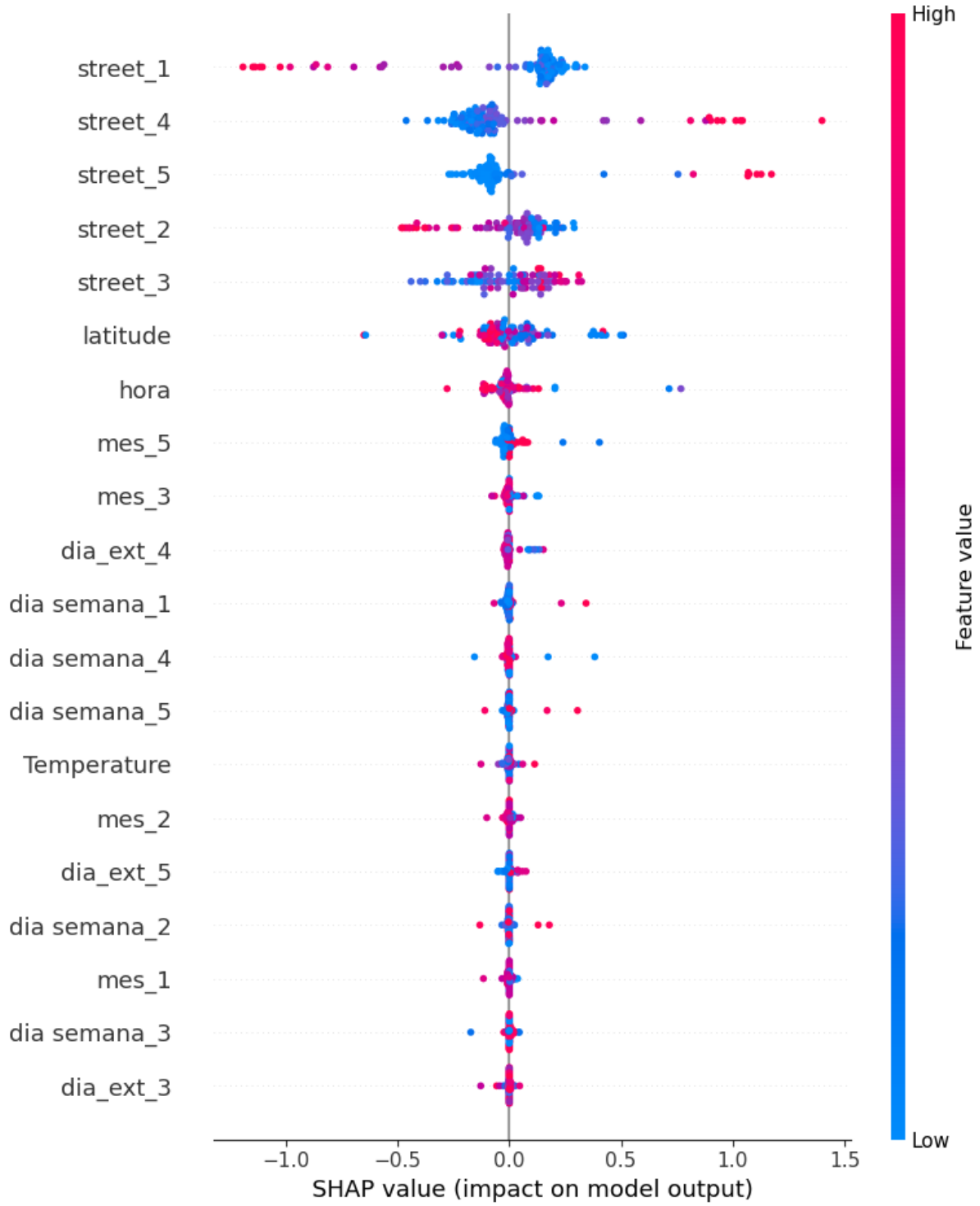

Figure 5.2 - SHAP beeswarm plot for traffic prediction

From the analysis of Figure 11, is possible to conclude that the features related to street (all 5) are the ones that most influence model predictions. Feature 'street_1' presents a negative correlation with model's output, since when it presents higher values, the prediction tends to be negatively influenced, leading to lower categories, i.e., labelled with smaller values. On the contrary, 'street_4' and 'street_5' present a positive correlation with model's prediction,

meaning that when they present higher values, the prediction is also one labelled with a higher value.

The other two street related features have a different behavior. Both features, when having higher values, can impact positively or negatively on the model's output. What differs between them is the fact that 'street_2' impact model's predictions in a positive direction when facing lower values, while, in these circumstances, 'street_3' impacts the output in a negative direction.

It is possible to see that the features related to time also have a greater impact on the model, while weather related ones don't seem to have such significant influence, only 'Temperature' appearing among the 20 most important features. It is important to mention that not all features are shown in the figure, only the ones with the highest impact on model's output.

SHAP waterfall plot is able to represent individual predictions and the impact of each feature in it. It shows how each prediction is strained in each direction as each feature impact score is applied one step at a time.

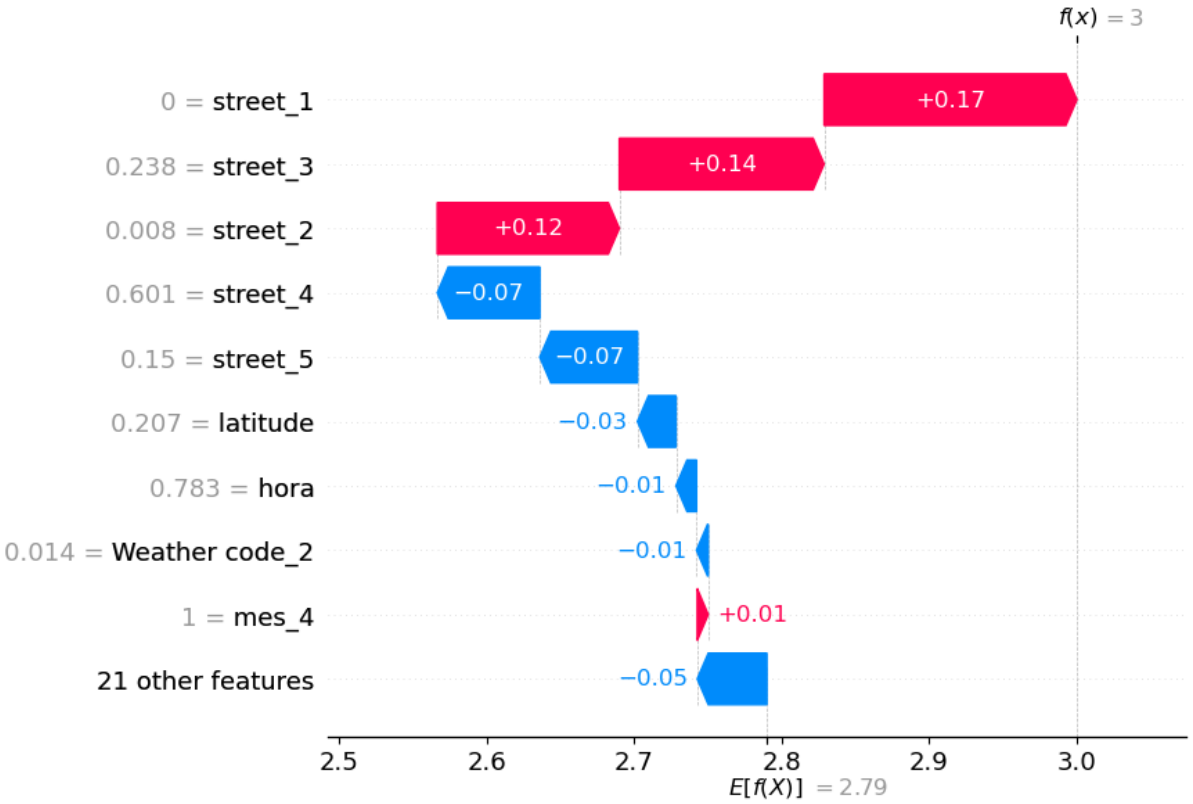

Figure 5.3 - SHAP waterfall plot for traffic prediction

As we can see in Figure 12, features 'street_1', 'street_3' and 'street_2' are the ones with greater impact on the final prediction, followed by 'street_4' and 'street_5'. The first set of features presents a positive impact on the output while the second set influences the

predictions in a negative direction. Also, the combination of all 21 features that are not represented in the figure accounts for only a variation of 0.05 in the final prediction.

Reinforcing what was explained for Figure 11, when 'steet_1' has a lower value (in this case 0), the model output is positively impacted, as can be seen in Figure 12, where an increase in 0.17 units is represented. Also, 'street_4' also solidifies what was concluded in Figure 11, since its higher value (0.601) negatively impacts the model output.

5.2. ALERTS PREDICTION

Now regarding alerts prediction, the models tested were the same ones used for traffic prediction. As performed previously, all models were tested in its simples form, with some parameter changes and with hyperparameter tuning.

Regarding the hyperparameter tuned models, it is important to access their characteristics, so they can be accurately replicated in the future.

The NN tuned model was returned with a hidden layer size set as (10,10,10), the activation function set as tanh, an adaptive learning rate, the solver set as sgd, and an initial learning rate set as 0.01. The DT tuned model was developed with the criterion set as gini, the max depth set as 30 and the splitter method was set as random.

For Bagging tuned model, the number of features to each base estimator was set as 0.6 and the number of base estimators as 200. For Random Forests tuned model, the number of features to consider when looking for the best split was set as log2, i.e., the logarithm of the total number of features, and the number of trees in the forest was set as 30.

Finally, Gradient Boosting tuned model was returned with 300 boosting stages to perform and the number of features to consider when looking for the best split set as sqrt, i.e., the square root of the total number of features.

In DL models, the experiments performed were the same ones done in traffic prediction, such as the number of hidden layers or the number of hidden units. The difference between the models of the two predictions resides majorly on the hyper tuned models, where the proprieties of each one were different from the ones obtained for the previous prediction.

For MLP tuned, the network was composed of 2 hidden layers, with softplus as the activation function and a learning rate set as 0.1. Regarding CNN tuned version, the model returned consisted of 2 convolutional layers, with softplus as the activation function for these layers, 5 pooling layers and the SGD optimizer set with a learning rate of 0.1.

Finally, the VGG tuned model was returned presenting 5 convolutional layers, with softplus as the activation function, and 5 pooling layers. The optimizer adopted for this model was also SGD, but with a learning rate of 0.01.

Table 5.3 - f1-score for alerts prediction

Model	F1 Score	
	Train	Validation
Neural Networks	0.94	0.94
Decision Trees	0.99	0.94
Bagging	0.99	0.95
Random Forests	0.99	0.96
Gradient Boosting	0.94	0.94
MLP	0.87	0.90
CNN	0.92	0.92
VGG	0.94	0.94

Similarly to what happened in traffic prediction, from the analysis of the table above, it's possible to see that ML models were able to predict alerts with more similar quality for train and validation datasets, reducing the presence of overfitting observed for traffic data. Decision Trees, Bagging and Random Forests still presented overfitting, but not a major obstacle since other models were outperforming them. From the DL methods applied, CNN and VGG were chosen as possible final models, due to their higher predictive quality and absence of overfitting/underfitting.

As performed for traffic prediction, all possible final models performances were analyzed. So, as mentioned previously, the final model was chosen between Neural Networks, Gradient Boosting, CNN and VGG. After analyzing other performance metrics, such as accuracy and recall, it was possible to conclude that Neural Networks was the most adequate model, being able to predict with more similar quality the different types of alerts, and with higher quality when compared to the other two models. Next, Table 5.4 shows the classification report of the chosen model.

Table 5.4 - Classification report on Neural Network's alerts prediction

Alert category	precision	recall	f1-score	support
0 (Accident)	0.99	0.88	0.93	99729
1 (Jam)	0.90	0.92	0.91	99366
2 (Road closed)	0.99	1.00	0.99	99705
3 (Weather hazard)	0.87	0.96	0.91	99369
accuracy			0.94	398169
macro avg	0.94	0.94	0.94	398169
weighted avg	0.94	0.94	0.94	398169

From the analysis of the table above, we can conclude that Neural Networks model has an overall precision, recall and f1-score of 0.94 (94%). It is also possible to conclude that it is able to almost perfectly predict the category *road closed*. Regarding the other types of alerts, the model has more difficulty in predicting *jam* and *weather hazard*, presenting performance scores below the ones observed for the other categories.

As experienced in traffic prediction, here *accident* presents a significant difference between precision and recall scores, meaning that the model is incorrectly classifying observations belonging to this category, but 99% of the instances classified as *accident* actually belong to this category.

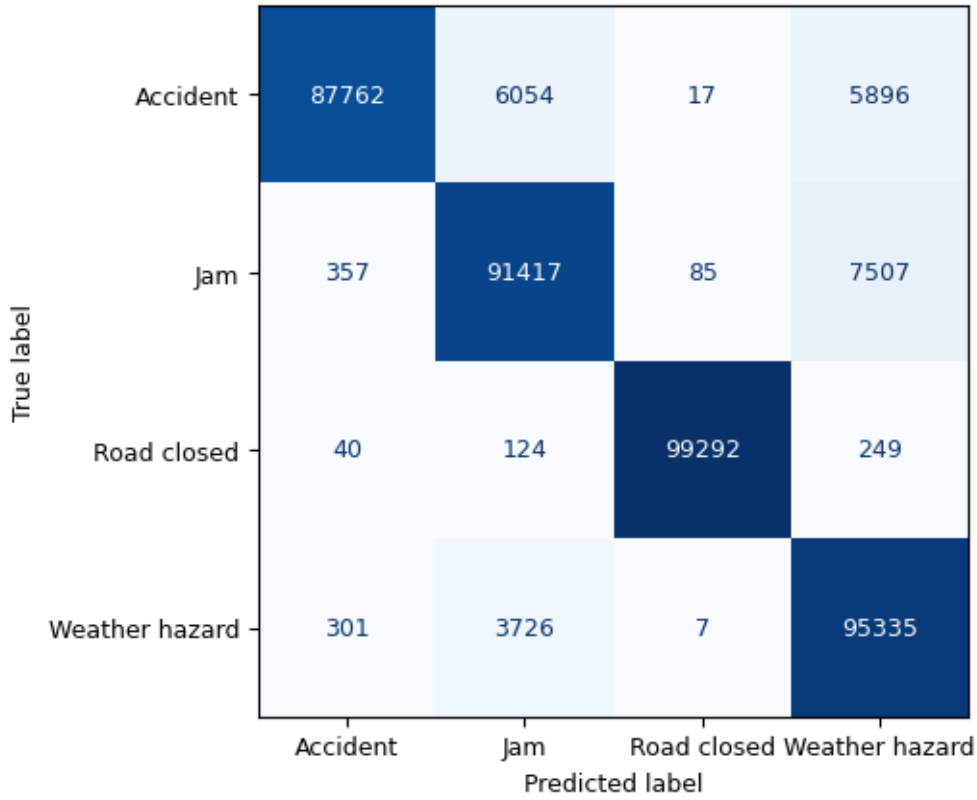

Figure 5.4 - Confusion matrix of alerts prediction

Figure 13, representing the confusion matrix for the alerts prediction, can be assessed as a visual representation of the classification report. We can conclude that category *road closed* is predicted with very high quality, where only a small number of observations were classified incorrectly.

The major obstacle faced by the model is when *weather hazard*, *jam* and *accident* are wrongly predicted, it usually happens between each other, where *weather hazard* is often classified as *jam*, and vice versa, and *accident* is often classified as both *jam* and *weather hazard*.

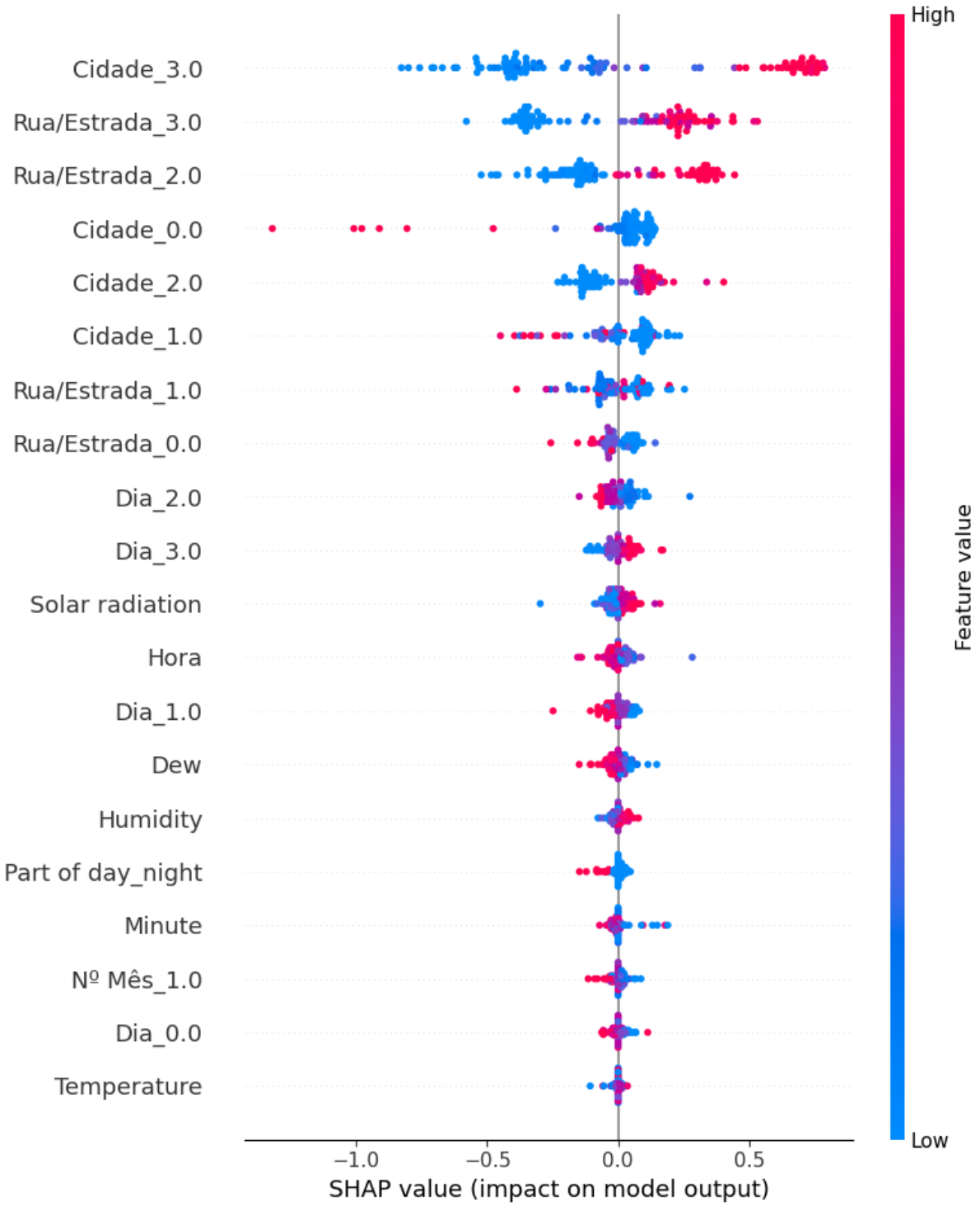

Figure 5.5 - SHAP beeswarm plot for alerts prediction

By the analysis of the above figure, is possible to conclude that the feature 'Cidade_3.0' is the one that most influences model predictions. When this feature presents low values, it has a negative impact on the prediction, while when it presents high values, its impact is positive, increasing the predicted value.

The same happens with ‘Rua/Estrada_3.0’, ‘Rua/Estrada_2.0’ and ‘Cidade_2.0’, all with significant impact on model’s output. Contrarily, ‘Cidade_0.0’ has the opposite effect on predictions, presenting a negative correlation. As mentioned for traffic prediction, not all features are represented in the figure, only the ones high higher impact on model output.

Finally, and as performed for traffic prediction, it is possible to analyze each prediction separately, visualizing which features highly impact each one, and the direction and magnitude of that same impact.

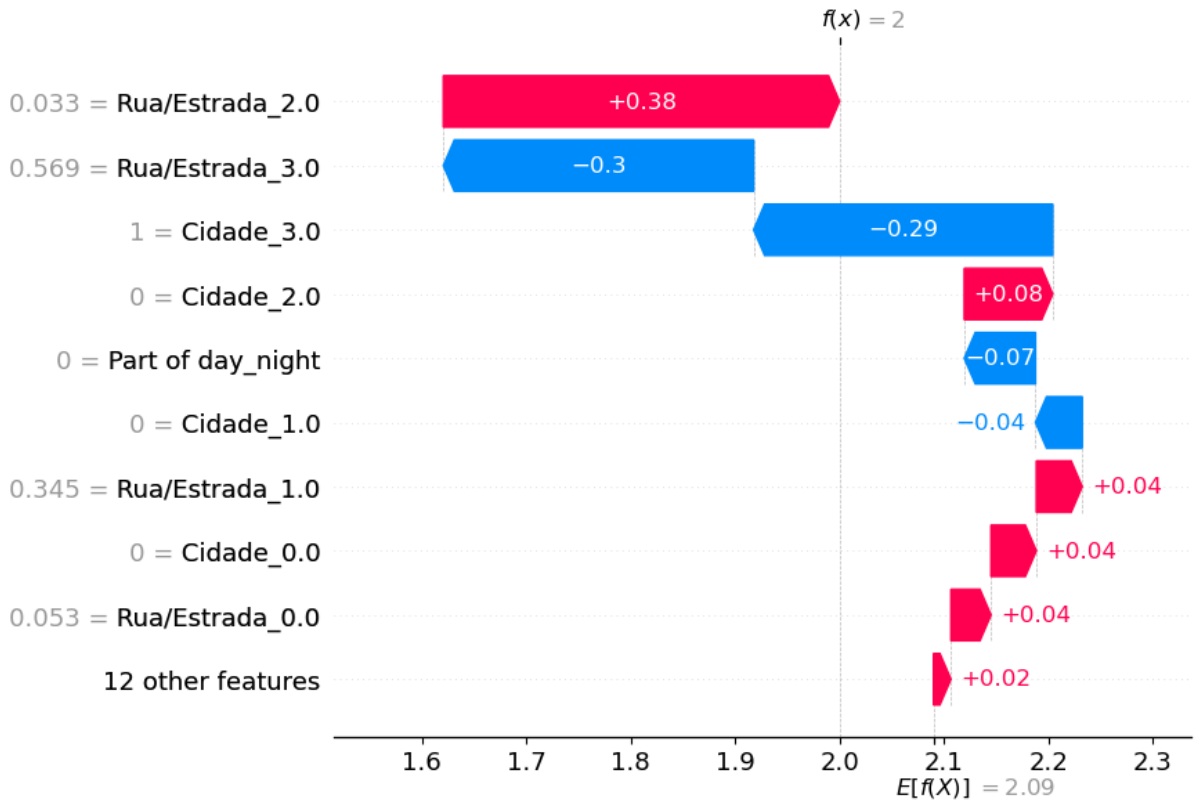

Figure 5.6 - SHAP waterfall plot for alerts prediction

In Figure 15, it is possible to see that both expected value and prediction belong to the same class, 2 (*road closed*). ‘Rua/Estrada_2.0’ is the feature that has a higher impact on the prediction, followed by ‘Rua/Estrada_3.0’ and ‘Cidade_3.0’. This is just the prediction of one observation, but still allows to take powerful insights regarding the impact of each feature on individual predictions.

As mentioned in the analysis of beeswarm plot, here we can also conclude that ‘Rua/Estrada_2.0’ is one of the features with higher impact on model’s output. Even though this remains true, for this specific observation, a lower value of this feature leads to an impact with a positive direction, which contradicts what was mentioned previously.

6. CONCLUSIONS

Upon the challenge purposed by OesteCIM to develop analytical models and visualization tools for the characterization and prediction of traffic patterns, alerts and incidents in Oeste region, several ML and DL algorithms were developed and tested using data retrieved by WAZE application.

The methodology adopted for this study was CRISP-DM, starting with an initial business understanding and followed by both data understanding and preparation. Different data sources were combined and pre-processed to prepare the datasets to be applied in modeling and evaluation phases. This was performed by inputting missing values, outlier processing, feature engineering, categorical features encoding, data normalization and feature selection.

During data preparation phase, some visual representations were developed, allowing a better contextualization with data under analysis and its spatio-temporal characteristics. Class imbalance was also an approach topic before modelling phase, as well as different data partition techniques were applied to train and evaluate the models.

Initially, different algorithms were developed and tested, from the most basic ML models to more advanced DL methods, and the best ones were optimized using GridSearchCV or Hyperband. Models with better performance were ensemble learning techniques and Neural Networks' architecture based.

Final results revealed that Gradient Boosting outperformed the majority of other tested classification algorithms for traffic prediction and Neural Networks was the best performing model for alerts prediction for the evaluation metrics selected.

Even though the models delivered good results, there were significant differences between prediction quality of the different categories for both traffic and alerts problems. Hyperparameter tuning techniques were able to improve basic models performance and returned the best models for both predictions, but also usually lead to underfitting or overfitting scenarios.

In conclusion, for traffic prediction the model chosen was Gradient Boosting with tuned hyperparameters that delivered predictions with a precision of 78.18%, a recall of 76.94%. of and an f1-score of 77.56%. Regarding alerts prediction, the chosen model Neural Networks classifier, also with hyperparameters tuned, produced results with a precision of 94.27%, a recall of 93.56%, and an f1-score of 93.91%.

For both traffic and alerts predictions location based features revealed to be the ones with higher impact on model's output, especially the ones regarding the street. Even though some rare events can occur, some roads are more prone to traffic and accidents than others, depending on their characteristics and travel demand, so it is normal that these features are the ones with higher impact.

Time related features were expected to have a higher influence on the model's output, especially part of day related features, since travel demand increases during the periods of the day when people are going to and returning from work. Weather features didn't have the impact expected, having a very small impact in the occurrence of traffic and the type of alert reported.

7. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORK

One of the limitations faced during the development of this study was related to computational needs. The dataset provided was quite extensive, making the main processes applied very time consuming and computationally expensive. This occurred specifically when performing hyperparameter tuning, missing values imputation and while performing cross-validation with repeated K-fold.

Another limitation faced was that the hyperparameter tuning was only performed on a limited set of hyperparameters and values, meaning that not all combinations and values of hyperparameters were tested, due to time and computational capacity constraints. In future work, it would be interesting to test other combinations of hyperparameters for the models presented, as well as apply it to non-discussed algorithms.

In the development of DL algorithms, the number of epochs was set as a low value due to time and computational effort constraints. Several models could have been tested, especially regarding DL approaches, where the models could be combined with other techniques to better account for both spatio-temporal dependencies.

Regarding visual representations of both traffic and alerts, only basic information was displayed. One of the original objectives of this project was also to develop more complex and useful visualizations, but due to time constraints and the complexity of the two predictive models developed, this part of the study was minimized.

Future work could pass by exploring new data sources related to traffic, for instance, events, industrial development and employment density. There are factors that impact traffic that are not being accounted for in this study that could lead to very different outcomes.

Another approach would be using speed as the target variable instead of traffic flow category, transforming the research into a regression problem. This wasn't performed in this study since the majority of its values correspondent to 0 km/h, meaning that the cars weren't moving when the report was made. This approach would allow a bigger comparison with previous research since most focus on traffic flow as a regression problem.

For alerts prediction, instead of forecasting which type of incident will occur, it should be interesting to predict if an event will occur. This was not performed in this study since the main focus was on traffic prediction, and it would be computationally complex since the dataset would be quite extensive.

A final recommendation for future work would be the development of other predictive models, for instance other DL algorithms that can be integrated with other components that make them more suited for this kind of problem or time series oriented models.

BIBLIOGRAPHICAL REFERENCES

Alajali, W., Zhou, W., & Wen, S. (2018). Traffic Flow Prediction for Road Intersection Safety. *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 812–820. https://doi.org/10.1109/SmartWorld.2018.00151

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a Convolutional Neural Network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, *2018-January*, 1–6. https://doi.org/10.1109/ICENGTECHNOL.2017.8308186

Bao, X., Jiang, D., Yang, X., & Wang, H. (2021). An improved deep belief network for traffic prediction considering weather factors. *Alexandria Engineering Journal*, *60*(1), 413–420. https://doi.org/10.1016/J.AEJ.2020.09.003

Batista, G. E., Monard, M.-C., Batista, G. E. A. P. A., & Monard, M. C. (2002). *A Study of K-Nearest Neighbour as an Imputation Method. 48*, 251–260. https://www.researchgate.net/publication/220981745

Boukerche, A., & Wang, J. (2020). Machine Learning-based traffic prediction models for Intelligent Transportation Systems. *Computer Networks*, *181*, 107530. https://doi.org/10.1016/J.COMNET.2020.107530

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control* (D. J. Balding, N. A. C. Cressie, G. M. Fitzmaurice, G. H. Givens, H. Goldstein, G. Molenberghs, D. W. Scott, A. F. M. Smith, R. S. Tsay, & S. Weisberg, Eds.; 5th ed.). John Wiley & Sons, Inc.

Box, G. E. P., & Tiao, G. C. (1975). Intervention analysis with applications to economic and environmental problems. *Journal of the American Statistical Association*, *70*(349), 70–79. https://doi.org/10.1080/01621459.1975.10480264

Buchmüller, J., Jäckle, D., Cakmak, E., Brandes, U., & Keim, D. A. (2019). MotionRugs: Visualizing Collective Trends in Space and Time. *IEEE Transactions on Visualization and Computer Graphics*, *25*(1), 76–86. https://doi.org/10.1109/TVCG.2018.2865049

Cauchy, A.-L. (2009). Méthode générale pour la résolution des systèmes d'équations simultanées. In *Oeuvres complètes* (Vol. 1, pp. 536–538). Cambridge University Press. https://doi.org/https://doi.org/10.1017/CBO9780511702396.063

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. SPSS.

Cinsdikici, M. G., & Memiş, K. (2010). Traffic flow condition classification for short sections using single microwave sensor. *Eurasip Journal on Advances in Signal Processing*, *2010*. https://doi.org/10.1155/2010/148303

Datta, A., Banerjee, S., Finley, A. O., & Gelfand, A. E. (2016). On nearest-neighbor Gaussian process models for massive spatial data. *Wiley Interdisciplinary Reviews. Computational Statistics*, *8*(5), 162. https://doi.org/10.1002/WICS.1383

Dervoort, M., Dougherty, M., & Watson, S. (1996). Combining Kohonen Maps with ARIMA Time Series Models to Forecast Traffic Flow. *Transportation Research. Part C: Emerging Technologies*, *4*(5), 307–310. https://doi.org/https://doi.org/10.1016/S0968-090X(97)82903-8

Fattah, M. A., Morshed, S. R., & Kafy, A. Al. (2022). Insights into the socio-economic impacts of traffic congestion in the port and industrial areas of Chittagong city, Bangladesh. *Transportation Engineering*, *9*, 100122. https://doi.org/10.1016/J.TRENG.2022.100122

Gu, Y., Lu, W., Xu, X., Qin, L., Shao, Z., & Zhang, H. (2020). An Improved Bayesian Combination Model for Short-Term Traffic Prediction With Deep Learning. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, *21*(3). https://doi.org/10.1109/TITS.2019.2939290

Guan, J., Wang, W., Li, W., & Zhou, S. (2018). A Unified Framework for Predicting KPIs of On-Demand Transport Services. *IEEE Access*, *6*, 32005–32014. https://doi.org/10.1109/ACCESS.2018.2846550

Guo, H., Wang, Z., Yu, B., Zhao, H., & Yuan, X. (2011). TripVista: Triple Perspective Visual Trajectory Analytics and its application on microscopic traffic data at a road intersection. *IEEE Pacific Visualization Symposium 2011, PacificVis 2011 - Proceedings*, 163–170. https://doi.org/10.1109/PACIFICVIS.2011.5742386

Hajirahimi, Z., & Khashei, M. (2019). Hybrid structures in time series modeling and forecasting: A review ☆. *Engineering Applications of Artificial Intelligence*, *86*, 83–106. https://doi.org/10.1016/j.engappai.2019.08.018

Harrou, F., Zeroual, A., & Sun, Y. (2020). Traffic congestion monitoring using an improved kNN strategy. *Measurement*, *156*. https://doi.org/10.1016/j.measurement.2020.107534

Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. *Procedia Computer Science*, *132*, 679–688. https://doi.org/10.1016/J.PROCS.2018.05.069

Kashyap, A. A., Raviraj, S., Devarakonda, A., Nayak K, S. R., Santhosh, K. V., & Bhat, S. J. (2022). Traffic flow prediction models–A review of deep learning techniques. In *Cogent*

Engineering (Vol. 9, Issue 1). Cogent OA. https://doi.org/10.1080/23311916.2021.2010510

Kowalek, P., Loch-Olszewska, H., & Szwabiński, J. (2019). Classification of diffusion modes in single-particle tracking data: Feature-based versus deep-learning approach. *Physical Review E*, *100*(3). https://doi.org/10.1103/PhysRevE.100.032410

Kumar Sharma, H., & Swami, B. L. (2016). Congestion Characteristics of Interrupted Flow for Urban Roads with Heterogeneous Traffic Structure. *MATEC Web of Conferences*. https://doi.org/10.1051/matecconf/20168101001

Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (2017). Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. *ArXiv E-Prints*. https://doi.org/https://doi.org/10.48550/arXiv.1703.07015

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*. https://doi.org/10.1162/neco.1989.1.4.541

Li, W., Cao, J., Guan, J., Zhou, S., Liang, G., So, W. K. Y., & Szczecinski, M. (2019). A General Framework for Unmet Demand Prediction in On-Demand Transport Services. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, *20*(8). https://doi.org/10.1109/TITS.2018.2873092

Li, X., Pan, G., Wu, Z., Qi, G., Li, S., Zhang, D., Zhang, W., & Wang, Z. (2012). Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science in China*, *6*(1), 111–121. https://doi.org/10.1007/S11704-011-1192-6

Liebig, T., Piatkowski, N., Bockermann, C., & Morik, K. (2017). Dynamic route planning with real-time traffic predictions. *Information Systems*, *64*, 258–265. https://doi.org/10.1016/J.IS.2016.01.007

Lin, L., Li, J., Chen, F., Ye, J., Member, S., & Huai, J. (2018). Road Traffic Speed Prediction: A Probabilistic Model Fusing Multi-Source Data. *IEEE Transactions on Knowledge and Data Engineering*. https://doi.org/10.1109/TKDE.2017.2718525

Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *BIT*, *16*, 146–160. https://doi.org/https://doi.org/10.1007/BF01931367

Lippi, M., Bertini, M., & Frasconi, P. (2013). Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, *14*(2), 871–882. https://doi.org/10.1109/TITS.2013.2247040

Liu, Z., Huang, M., Ye, Z., & Wu, K. (2019). DeepRTP: A Deep Spatio-Temporal Residual Network for Regional Traffic Prediction. *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*. https://doi.org/10.1109/MSN48538.2019.00062

Luo, X., Li, D., Yang, Y., & Zhang, S. (2019). Spatiotemporal traffic flow prediction with KNN and LSTM. *Journal of Advanced Transportation*, *2019*. https://doi.org/10.1155/2019/4145353

Ma, X., Yu, H., Wang, Y., & Wang, Y. (2015). Large-scale transportation network congestion evolution prediction using deep learning theory. *PLoS ONE*, *10*(3). https://doi.org/10.1371/JOURNAL.PONE.0119044

Matthews, A. G. de G., Hron, J., Rowland, M., Turner, R. E., & Ghahramani, Z. (2018). Gaussian Process Beaviour in Wide Deep Neural Networks. *ArXiv*, *abs/1804.11271*. https://doi.org/https://doi.org/10.48550/arXiv.1804.11271

Mijwil, M. M., Alsaadi, A., Mijwel, M. M., Esen, A., & Shamil, A. (2019). Overview of Neural Networks. *An Overview of Neural Network*. https://doi.org/10.11648/j.ajnna.20190501.12

Min, W., & Wynter, L. (2011). Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, *19*(4), 606–616. https://doi.org/10.1016/J.TRC.2010.10.002

Min, X., Hu, J., Chen, Q., Zhang, T., & Zhang, Y. (2009). Short-term traffic flow forecasting of urban network based on dynamic STARIMA model. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 461–466. https://doi.org/10.1109/ITSC.2009.5309741

Min, X., Hu, J., & Zhang, Z. (2010). Urban traffic network modeling and short-term traffic flow forecasting based on GSTARIMA model. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 1535–1540. https://doi.org/10.1109/ITSC.2010.5625123

Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., & Damas, L. (2013). Predicting Taxi-Passenger Demand Using Streaming Data Index Terms-Autoregressive integrated moving average (ARIMA), data streams, ensemble learning, Global Positioning System (GPS) data, mobility intelligence, taxi-passenger demand, time-series forecasting, time-varying Poisson models. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, *14*(3), 1393. https://doi.org/10.1109/TITS.2013.2262376

Oliveira, T. P., Barbar, J. S., & Soares, A. S. (2016). Computer network traffic prediction: a comparison between traditional and deep learning neural networks. *International Journal of Big Data Intelligence*, *3*(1), 28. https://doi.org/10.1504/IJBDI.2016.073903

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *ArXiv E-Prints*. https://doi.org/https://doi.org/10.48550/arXiv.1511.08458

Peng, T., Yang, X., Xu, Z., & Liang, Y. (2020). Constructing an environmental friendly low-carbon-emission intelligent transportation system based on big data and machine learning methods. *Sustainability (Switzerland)*, *12*(19). https://doi.org/10.3390/su12198118

Raheem, S. B., Olawoore, W. A., Olagunju, D. P., & Adeokun, E. M. (2015). The Cause, Effect and Possible Solution to Traffic Congestion on Nigeria Road (A Case Study of Basorun-Akobo Road, Oyo State). *International Journal of Engineering Science Invention*, *4*(9). https://api.semanticscholar.org/CorpusID:212503505

Rokach, L., & Maimon, O. (2006). Decision Trees. *Data Mining and Knowledge Discovery Handbook*, 165–192. https://doi.org/10.1007/0-387-25465-X_9

Rolison, J. J., Regev, S., Moutari, S., & Feeney, A. (2018). What are the factors that contribute to road accidents? An assessment of law enforcement views, ordinary drivers' opinions, and road accident records. *Accident Analysis & Prevention*, *115*. https://doi.org/https://doi.org/10.1016/j.aap.2018.02.025.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagation errors. *Nature*, *323*. https://doi.org/10.1038/323533a0

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., & Gasthaus, J. (2019). High-Dimensional Multivariate Forecasting with Low-Rank Gaussian Copula Processes. *Advances in Neural Information Processing Systems*, *32*. https://doi.org/https://doi.org/10.48550/arXiv.1910.03002

Shaheen, S. A., & Finson, R. (2016). Intelligent Transportation Systems. In *Reference Module in Earth Systems and Environmental Sciences*. Elsevier. https://doi.org/10.1016/B978-0-12-409548-9.01108-8

Shaygan, M., Meese, C., Li, W., Zhao, X. (George), & Nejad, M. (2022). Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities. *Transportation Research Part C: Emerging Technologies*, *145*, 103921. https://doi.org/10.1016/J.TRC.2022.103921

Shekhar, S., & Williams, B. M. (2007). Adaptive seasonal time series models for forecasting short-term traffic flow. *Transportation Research Record*, *2024*, 116–125. https://doi.org/10.3141/2024-14

Shi, X., Qi, H., Shen, Y., Wu, G., & Yin, B. (2021). A Spatial-Temporal Attention Approach for Traffic Prediction. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, *22*(8), 4909. https://doi.org/10.1109/TITS.2020.2983651

Shih, S.-Y., Sun, F.-K., & Lee, H. (2018). Temporal Pattern Attention for Multivariate Time Series Forecasting. *Machine Learning*, *108*. https://doi.org/https://doi.org/10.48550/arXiv.1809.04206

Sims, C. A. (1980). Macroeconomics and Reality. *Econometrica*, *48*(1), 1. https://doi.org/10.2307/1912017

Singh, J., & Banerjee, R. (2019). A Study on Single and Multi-layer Perceptron Neural Network. *3rd International Conference on Computing Methodologies and Communication (ICCMC)*. https://doi.org/10.1109/ICCMC.2019.8819775

Song, C., Lee, H., Kang, C., Lee, W., Kim, Y. B., & Cha, S. W. (2017). Traffic speed prediction under weekday using convolutional neural networks concepts. *IEEE Intelligent Vehicles Symposium, Proceedings*, 1293–1298. https://doi.org/10.1109/IVS.2017.7995890

Song, Z., Duan, H., Zhou, S., & Qiu, X. (2015). Urban Route Planning Considering Traffic Flows. *Chinese Automation Congress (CAC)*. https://doi.org/10.1109/CAC.2015.7382822

Sun, G., Liang, R., Qu, H., & Wu, Y. (2017). Embedding Spatio-Temporal Information into Maps by Route-Zooming. *IEEE Transactions on Visualization and Computer Graphics*, *23*(5), 1506–1519. https://doi.org/10.1109/TVCG.2016.2535234

Taghipour, H., Parsa, A. B., & Mohammadian, A. (Kouros). (2020). A dynamic approach to predict travel time in real time using data driven techniques and comprehensive data sources. *Transportation Engineering*, *2*. https://doi.org/10.1016/j.treng.2020.100025

Tang, L., Zhao, Y., Cabrera, J., Ma, J., & Tsui, K. L. (2019). Forecasting Short-Term Passenger Flow: An Empirical Study on Shenzhen Metro. *IEEE Transactions on Intelligent Transportation Systems*, *20*(10), 3613–3622. https://doi.org/10.1109/TITS.2018.2879497

Vanderschuren, M. J. W. A. (2003). The Benefits of Intelligent Transport Systems: Modelling the Effects of Different ITS Systems. *UPSpace*. http://hdl.handle.net/2263/7054

Vencataya, L., Pudaruth, S., Dirpal, G., & Narain, V. (2018). Assessing the Causes & Impacts of Traffic Congestion on the Society, Economy and Individual: A Case of Mauritius as an Emerging Economy. *Studies in Business and Economics*, *13*(3), 230–242. https://doi.org/10.2478/sbe-2018-0045

Vieira Gomes, S., Carvalheira, C., Cardoso, J., & Picado Santos, L. (2008). Accident prediction models in urban areas: Lisbon case study. *WIT Transactions on the Built Environment*, *101*, 619–627. https://doi.org/10.2495/UT080601

Wagner-Muns, I. M., Guardiola, I. G., Samaranayke, V. A., & Kayani, W. I. (2018). A Functional Data Analysis Approach to Traffic Volume Forecasting. *IEEE Transactions on Intelligent Transportation Systems*, *19*(3), 878–888. https://doi.org/10.1109/TITS.2017.2706143

Walraven, E., Spaan, M. T. J., & Bakker, B. (2016). Traffic flow optimization: A reinforcement learning approach. *Engineering Applications of Artificial Intelligence*, *52*, 203–212. https://doi.org/10.1016/j.engappai.2016.01.001

Williams, B. M., & Hoel, L. A. (2003). Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of Transportation Engineering*, *129*(6), 664–672. https://doi.org/10.1061/(ASCE)0733-947X(2003)129:6(664)

Wu, P., Huang, Z., Pian, Y., Xu, L., Li, J., & Chen, K. (2020). A combined deep learning method with attention-based LSTM model for short-term traffic speed forecasting. *Journal of Advanced Transportation*, *2020*. https://doi.org/10.1155/2020/8863724

Wu, Y., Tan, H., Qin, L., & Jiang, Z. (2018). A hybrid deep learning based traffic flow prediction method and its understanding. *Transportation Research Part C: Emerging Technologies*, *90*. https://doi.org/https://doi.org/10.1016/j.trc.2018.03.001

Yamashita, R., Nishio, M., Kinh, R., Do, G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, *9*. https://doi.org/10.1007/s13244-018-0639-9

Yang, X., Zou, Y., Tang, J., Liang, J., & Ijaz, M. (2020). Evaluation of Short-Term Freeway Speed Prediction Based on Periodic Analysis Using Statistical Models and Machine Learning Models. *Journal of Advanced Transportation*, *2020*. https://doi.org/10.1155/2020/9628957

Yao, W., & Qian, S. (2021). From Twitter to traffic predictor: Next-day morning traffic prediction using social media data. *Transportation Research Part C: Emerging Technologies*, *124*. https://doi.org/10.1016/j.trc.2020.102938

Zhang, L., Liu, Q., Yang, W., Wei, N., & Dong, D. (2013). An Improved K-nearest Neighbor Model for Short-term Traffic Flow Prediction. *Procedia-Social and Behavioral Sciences*, *96*, 653–662. https://doi.org/10.1016/j.sbspro.2013.08.076

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., & Li, H. (2020). T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, *21*(9). https://doi.org/10.1109/TITS.2019.2935152

Zheng, C., Fan, X., Wang, C., & Qi, J. (2019). GMAN: A Graph Multi-Attention Network for Traffic Prediction. *ArXiv*, *abs/1911.08415*. https://doi.org/https://doi.org/10.48550/arXiv.1911.08415

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa