FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Empirical Evaluation of Prediction Models of People Density

Iohan Xavier Sardinha Dutra Soares

Mestrado em Engenharia Informática e Computação

Supervisor: Dr. Ana Paula Rocha

Second Supervisor: Dr. Rosaldo J. F. Rossetti

July 4, 2023

Empirical Evaluation of Prediction Models of People Density

Iohan Xavier Sardinha Dutra Soares

Mestrado em Engenharia Informática e Computação

July 4, 2023

Resumo

A previsão do fluxo de pessoas aplicada às cidades inteligentes é uma tarefa que pode fornecer dados essenciais para os gestores de uma cidade utilizarem em diferentes aplicações, seja para estratégias de otimização a longo prazo, como a construção de infraestrutura, ou para operações diárias, como mapeamento da rede de tráfego, alocação de segurança e organização de eventos.

A previsão do fluxo de pessoas pode ser vista como um problema de séries temporais que analisa dados espacial e temporal em uma rede representando o número de pessoas (referido como densidade) em um espaço específico e o fluxo entre esses espaços ao longo do tempo, que é o fator que muda a densidade. Embora os algoritmos modernos de aprendizado de máquina executem com eficiência a previsão de dados espaciais e temporais, juntar os dois ainda pode ser desafiador.

A dissertação discute os fatores mais relevantes para gerar um bom modelo, os melhores métodos de aprendizado e a análise empírica dos resultados. A origem central dos dados é o histórico de dados de telecomunicações por telefone real que fornece o fluxo de pessoas entre as zonas da cidade portuguesa, juntamente com outras fontes, como dados de clima, eventos e características geográficas das áreas que podem contribuir para um modelo mais preciso.

Diferentes métodos de aprendizado de máquina e de aprendizado profundo são testados e avaliados contra o estado da arte em termos de qualidade. Graph Neural Networks são uma arquitetura de aprendizado profundo otimizada para dados estruturados em grafos, como entradas geográficas, e têm mostrado resultados promissores com esse tipo de dados em estudos recentes.

O uso de Graph Neural Networks e a associação com os dados geográficos são as principais contribuições desta dissertação. No entanto, o trabalho resultante pode ser uma referência para as cidades projetarem sua gestão especificamente para necessidades baseadas nos inputs estudados que têm a maior influência.

Abstract

Crowd flow forecasting applied to smart cities is a task that can provide essential data for a city planner to use in different applications either for long-term optimization strategies, such as building infrastructure, or for daily operations, such as traffic network mapping, security allocation, and organizing events.

Predicting crowd flow can be seen as a time series problem that analyses spatio-temporal data on a network representing the number of people (referred to as density) in a specific space and the flow between these spaces over time which is the factor that changes the density. Although modern machine learning algorithms efficiently predict spatial and temporal data, joining the two can still be challenging.

The dissertation discusses the most relevant inputs to generate a good model, the best learning methods, and the empirical analysis of the results. The central origin of input is historical real-world phone telecommunications data that provides the flow of people between the zones of the Portuguese city, alongside other sources such as data from weather, events, and geographical characteristics of the areas that can contribute to a more accurate model.

Different machine learning and deep learning methods are tested and evaluated against the state of the art for it is quality. Graph Neural Networks are an architecture of deep learning optimized for graph-structured data, such as geographical inputs, and have shown promising results with such data in recent studies.

The main contributions of this dissertation are the use of Graph Neural Networks and the association with geographical data. Nevertheless, the resulting work can be a reference for cities to project their management specifically for necessities based on the studied inputs that have the most influence.

Keywords: Crowd flow, spatio-temporal data, time series, ensemble models, machine learning

Acknowledgments

I first want to thank my supervisors, as during the period of conducting this thesis, I am certain it would have been much more difficult without their help. Professor Rosaldo Rossetti, whom I have always connected with since our conversations during AEDA practices, and who helped me get this dissertation topic. And Professor Ana Paula Rocha, who always gave me the utmost attention and support throughout the development process. Every time I was desperate, thinking I couldn't do it, she encouraged me to keep going and eventually succeed.

I also want to thank my mother, grandmother, father, sisters, and my entire family for their assistance and support throughout my academic journey. I known that even if I the whole world turned their back to me, my family would be there to help with all their love.

And thank all my friends who made this journey more enjoyable and provided me with some of the best moments of my life. Without them, our outings to chat, walk, cook, write the thesis in cafes, go to parties and bars to unwind, I'm not sure if I would have had the sanity to complete this journey.

Iohan Xavier Sardinha Dutra Soares

"Chegou a hora dessa gente bronzeada mostrar seu valor"

Assis Valente

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial Neural networks
ARIMA	Autoregressive integrated moving average
AutoML	Automatic Machine Learning
BLE	Bluetooth Low Energy
CNN	Convolutional Neural Network
CV	cross-validation
ConvLSTM	Convolutional LSTM
ETS	Error-Trend-Seasonality
GA	Genetic Algorithm
GNN	Graph Neural Network
GP	Gaussian Processes
GPDR	General Data Protection Regulation
GRU	Gated Recurrent Unit
HA	Historical Average
KNN	K-nearest neighbors
LAI	Leaf Area Index
LSTM	Long short-term memory
LSTMGC	Long short-term memory Graph Convolution
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MAXAE	Max Absolute Error
MC	Markov Chain
MLP	Multi-Layer Perceptron
MSE	Mean squared error
NN	Neural Network
OOS	out-of-sample
OSM	Open Street Maps
PCA	Principal Component Analysis
POI	Point of Interest
RMSE	Root-mean-square deviation
SEATS	Seasonal Extraction in ARIMA Time Series
STARIMA	Spatio-temporal ARIMA
STGCN	Spatio-temporal Convolutional Network
SVM	Support Vector Machines
STCGN	Spatio-temporal Graph Convolutional Network
USGS	United States Geological Survey
VAR	Vector-AutoRefressive
VARIMA	Vectorial ARIMA

Chapter 1

Introduction

In this chapter, the primary focus revolves around the description of the problem at hand, as presented in this dissertation, together with the practical applications that emanate from it. Additionally, this chapter aims to shed light on the underlying goals and motivations that have propelled the research forward throughout the entire process. Furthermore, an explicit account of the document's structure is provided, offering readers a roadmap to navigate through the various sections and components that constitute it.

1.1 Context

City management is a challenge that has existed ever since cities exist, and as cities grow in space and population, this task becomes even more complex. Modern cities are very complex systems with different regions, each with its own particularities, but all connected with the movement of people and a large variety of activities happening simultaneously. To maintain this system working at its best performance while keeping the best quality of life for the residents, the cities will often attempt to create models that explain the inner behaviors of each of its gears.

Smart Cities come as a concept in which technology can be used for the purpose of managing the city for the best benefit of all of its components. The IEEE IoT Initiative's Smart Cities Working Group defines Smart City as: "an urban area that uses technological or non-technological services or products that: enhance the social and ethical well-being of its citizens; provide quality, performance, and interactivity of urban services to reduce costs and resource consumption; and increase contact between citizens and government."[1]

Such technological services include the use of sensors that collect urban data that will be used for the models that describe and help the city. With large enough amounts of data, models can be used to predict traffic, compute optimal routes for public transport, or find the best regions to invest in habitation or commerce. This knowledge brought through the use of technology can be used by city planners, its inhabitants, and the private services that operate in the city. With it, time spent in locomotion can be reduced, shorter routes can lead to less carbon emission, and in situations like the recent COVID-19 pandemic, the flow of people may be tracked or redirected to keep track and reduce contact between potentially infected people.

1.2 Objectives

This dissertation fits in the context of Crowd Analysis in smart cities. With its results, it is expected to understand the patterns of population distribution over time in the different subdivisions of the city, what factors, and to what extent they influence these patterns. This is done through the creation of different algorithms for Crowd Density forecasting and the comparison of them. The comparison aims to find the most suited model to be used by the city in real-life applications and empirically understand the influence of external factors on the prediction. The final model is able to receive as input a moment in the future together with the detected external features for that moment, and based on data trained from the past, tell how many people will be in each subdivision at that moment.

Spatio-temporal data from a major telecommunications company for the city of Vila Nova de Famalicão, in northern Portugal, is the primary source for the models. Data from meteorological stations and satellite data is also used. The forecasting models are done based on this data, and the goals will be all relative to the city. So this dissertation aims to identify a model to predict the number of people in the zones of Famalicão, analyze the differences between the models, and the influence of the geographical and weather features.

1.3 Motivation and Contributions

Understanding city dynamics is the main motivation for this dissertation. Having a deep knowledge of how people move and its associated aspects has a scientific motivation to understand a complex behavior that is not fully understood and a practical one to help city planners and administrators. Having a crowd density forecasting model gives the possibility of use in city expansion, planning for events, zoning, and preparing for seasonal increases or decreases in population and disease control. The public could use this knowledge for planning its schedules, the industry for new developments, and the City Councils for daily and strategic management.

The contributions are to the development of smart cities that could use the model and the processes used to create it and evaluate it for a better understanding of the city dynamics to improve the city, creating solutions useful for each city's peculiarities. A pipeline for the merging treatment of geographical data to be used as input for machine learning models was created and may be used by other agents. Future research may use the techniques and knowledge about the models developed.

1.4 Dissertation Structure

This first chapter shows the context in which this dissertation is inserted, describes the goals aimed during its development, then details the motivations that led to it and the contributions that are expected for it to leave. The next chapter, chapter 2, is named State of the Art, in it first a background is given, explaining the main concepts necessary to understand the work done, then a state-of-the-art analysis is done, to display the current most modern methods in crowd analysis and forecasting, together with a gap analysis for understanding where should this work focus to bring innovation. In chapter 3, the problem is explained in detail, as well as the data used, then the solution is dissected, and each model is described with its benefits and disadvantages. chapter 4 shows the final results, and compares them with each other, in order to find the best model and explain the reasons behind it. Finally, in chapter 5, the last conclusions are made, about the whole work, and the contributions that could be made in future work are outlined. And the Appendix A contains extra information such as graphs and tables that help understand the work developed and was not fit for the main text.

Chapter 2

State of the Art

This chapter outlines the necessary background for understanding the topics addressed in the current dissertation, first by explaining the problems, methods, and technologies involved and then by analyzing the state of the art for solving such problems. It is primarily divided into the description of Prediction Models, which is the main task to be developed; the description of Time Series, which is the mathematical model that best describes the problem; and Crowd Analysis, which is the specification of the task as a known and studied problem with previous solutions.

2.1 Background

2.1.1 Prediction Models

Prediction models are models, or algorithms, that intend to predict unknown values based on related data. In most cases, and as is the focus of this thesis, the known values are values observed in the past, and the unknown values are future values; this determines a forecast of events.

Historically, models that describe the world have been created based on math, physics, or a series of steps, from which observing the behavior of related objects and variables, the unknown variable could be predicted. Physics formulas are a great example; knowing the current speed and direction of an object, the physical model gives equations to predict its future position. Complex behaviors such as weather cannot be explained by single equations and require full large-scale statistical models for forecasting.[2]

Complex events that had many influencing variables were very hard to model and calculate by hand since there were many factors to take into consideration and many relations that may not be detected or difficult to formulate. However, the use of computers enabled models to become ever more complex and less linear, finding patterns that are not as easily detectable by a human. Modern models can vary a lot, the old statistical models are still used, but now can be approximated more quickly and easily by computers; and more modern data-focused models are also used, where the computer, through a series of steps, can find patterns without minimal or none previous knowledge.

Regression Models resort to the traditional statistical and mathematical processes for prediction. Regression models consist of finding a function that maps a set of independent variables (inputs, also named data points) into a dependent variable (output or response) that will be a prediction. This function is not considered an authentic representation of the data but an approximation with a variance with superimposed random errors.[3] The task of creating a regression model, then, consist in finding first the function that defines a curve similar to the data distribution, then finding the parameters for this function that best approximate the data. The functions can be of different degrees, linear or nonlinear. This task does not need computers and can be done by traditional mathematical means. Still, computers give the ability to test many values in a short time and compute the coefficients much quicker and precisely than any human. Regression models are opposed by classification models that do not deal with continuous values, finding a relation between the inputs and classes instead.

The most common regression models are linear regression, polynomial regression and logistic regression, and ridge regression. Linear regression assumes a linear relationship between the independent variables and the dependent variable. It estimates the parameters of a linear equation that best fits the observed data points. The model assumes that the errors are normally distributed and that there is no multicollinearity among the independent variables, which means that the variables have a low degree of correlation which would make it hard to determine the impact of each one on the model if it was the case. Linear regression provides the strength, direction, and significance of the relationship between variables, allowing for predictions and hypothesis testing. Polynomial regression extends linear regression by including higher-order terms. Logistic regression models the probability of a binary outcome. And ridge regression introduces regularization to mitigate overfitting.

Machine Learning is a vast category that defines many processes, being the common characteristic that they are data-centric. The data is given to a program that will train with it, trying many times to approximate a model that fits the data [4]. Regression models can be considered machine learning since the computer will find the approximated regression function based on the input data. However, the act of approximating the model to the data may happen in ways that are not directly represented by traditional functions, such as with decision trees. And also using very complex high-dimensional functions which are not easy to visualize as regressions. Machine Learning can take longer to train the less linear the data relations are depending on the model. If the data can be easily represented by simple functions, it should converge and train quickly, while data with complex relations takes much longer.

The methods from which the data is approximated define the many existing different machine learning models. The most common to describe are:

- Support vector machines, which are supervised learning models used for classification and regression analysis by creating hyperplanes or sets of hyperplanes in high-dimensional spaces to separate data points into different classes or predict continuous values

- Tree-based models, which are decision tree algorithms that use a hierarchical structure of branching decisions based on features to make predictions or classify data

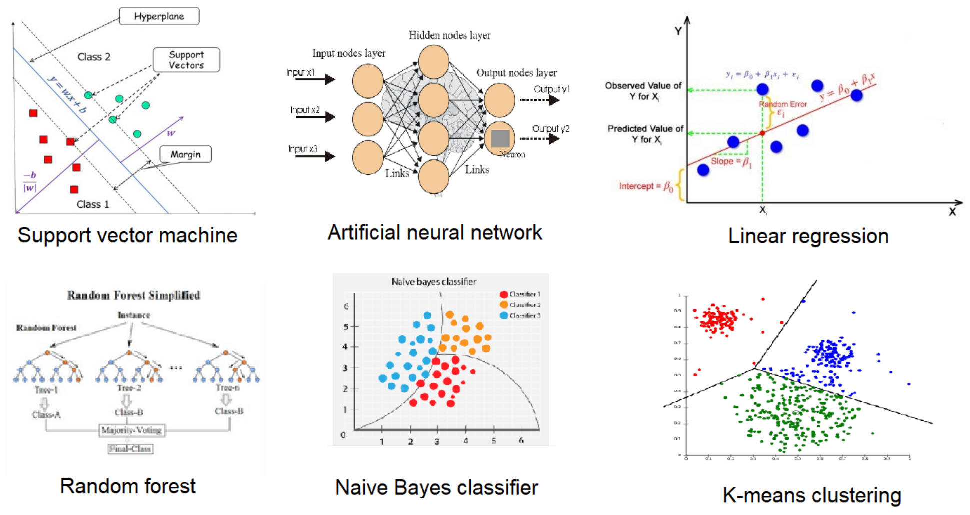


Figure 2.1: Some machine learning methods side by side

- Artificial neural networks, which are interconnected networks of artificial neurons inspired by the biological neural networks in the human brain that can learn and make predictions by adjusting the weights and biases of the connections between neurons.

The Figure 2.1 exemplifies the architecture of some of the described models. SVM, Naive Bayes, and K-means show the data and the lines that it learns to divide them in a two-dimensional example. Linear Regressions show the approximated linear function between the data points. Random Forest shows an example decision tree. And Artificial Neural Network shows the neuron architecture with one input layer, one hidden layer, and one output layer.

2.1.2 Time Series, Spatial and Spatio-Temporal Data

This section describes the spatio-temporal problems and data, which is the main kind of data approached in this work, by first explaining two other kinds that compose it: Time-Series(temporal) and Spatial.

Time Series

Time Series are data collections bounded to moments in time, usually of constant distance and organized sequentially[5]. This characteristic makes it a perfect candidate for analyzing sensor data, which is, in the context of smart cities, the main origin of information. Sensors will get the value of its observed phenomenon in constant, non-continuous, sequential time intervals. For this reason, Time Series are used for studying the evolution of weather, stock prices, the number of cars going through a road, and the number of people in a region, over a period of time.

Examples around the world include the Zurich street lighting project that controls the intensity of street lights around the city based on traffic flow observed behavior and enabled energy savings of up to 70%[6]; Seoul monitors traffic flow, speed, and air quality from which can run analysis as a basis for its services[7]. And Copenhagen uses collected data from many sensors to redirect traffic in real-time[8]. These examples use data collected from sensors that change over time and could be modeled and understood as time series.

Every sequence of data points that evolve over time will have relations between these points, from which a lot of information of the time series can be extracted and will help define how to predict its future values properly. Autocorrelation is the similarity between a time series and a lagged version of itself [9]. This measure is used to find trends and seasonality. For example, values may repeat every 24 hours, showing a daily periodicity in the data. A large degree of autocorrelation may be a problem when modeling a time series because it may lead to overfitting. A trend is a change in the observed value in a moment of time if it's growing, shrinking, or maintaining. Seasonality refers to periodic changes, such as having peaks of people in a region every summer.

Stationarity is a characteristic of a time series that determines that its statistical properties do not change over time, so its mean and variance are constant. Many statistical methods for time series analysis assume that the data is stationary. A common way of trying to make a non-stationary series stationary is by differentiating it, that is, subtracting its values from previous lagged ones. By taking the difference between consecutive observations, the trend component present in the data can be eliminated. Furthermore, differencing can also help in stabilizing the series variance, making it more amenable to modeling and analysis.[10]

The process of extracting these components of a time series is called decomposition. It consists of identifying values that together will define the series and reveal important information if analyzed separately. Decompositions can be additive or multiplicative, which literally means that the final value of the observed variable can be reconstructed by either adding or multiplying the decomposed components. Common decompositions are STL which decomposes the series into seasonality, trend, and residual; SEATS (Seasonal Extraction in ARIMA Time Series) which is a decomposition technique that combines ARIMA(AutoRegressive Integrated Moving Average) modeling with seasonal adjustment to extract seasonal, trend, and residual components; and X11, a seasonal adjustment method developed by the U.S. Census Bureau that decomposes a time series into seasonal, trend, and irregular components based on moving averages and seasonal filters. Residual refers to the rest, the part of the value that is not explained by the other components. It's important to notice that the residual should be random since if there's a clear pattern on it, then this pattern is not being captured by one component as it should.[5]

Prediction can be made with a decomposition by predicting each component and combining them to get the original time series data. This prediction can be made by any means. Usually, a regression is a good candidate, but each component can use a different method. Additive decompositions will sum the components to get to the original value, while multiplicative will multiply them. Equation 2.1 presents an additive decomposition where y is the actual value of the observed variable for the time t and STL are the Seasonal Trend and residual components of an STL decomposition. While Equation 2.2 shows the same but for a multiplicative decomposition.

$$y_t = S_t + T_t + L_t \tag{2.1}$$

$$y_t = S_t \cdot CTt \cdot L_t \tag{2.2}$$

There are many ways of forecasting a time series, the most common being: regression, modeling, and machine learning methods. Since Time Series data usually come from observations, time is a discrete variable in opposition to its natural continuity. Thus when forecasting using regression, the model goal is finding the underlying formula that, when dealing with time as continuous, would have the closest values to the original ones on the observed time moments.

Modeling in the context of time series is similar to the process of decomposition. It consists of creating a mathematical model that, through a series of variables, can approximate the series. It differs from regression by treating the evolution of the series as a step-by-step process, even though these steps can be infinitely reduced to make a continuous process. And also by using the different properties of the series instead of a function of the observed variable. One of the most common models used is ARIMA and its variations. ARIMA stands for AutoRegressive Integrated Moving Average and is a combination of two different models: autoregressive and moving average. Autoregressive models use the previous moments in time on the observed data to make a linear combination and predict the next value, similar to a regression. And moving average models uses a linear combination of the previous errors. Variations of ARIMA include VARIMA(Vectorial ARIMA), which is a generalization of the model using vectors and is used to model more than one observed variable, and VARMAX, which is also vectorial and takes into consideration exogenous variables.[11]

An autoregressive model of order p, named AR(p), with ε as the error, is represented by the Equation 2.3:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t \tag{2.3}$$

And a moving average model named MA(p) of order q is represented by Equation 2.4:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \tag{2.4}$$

Finally, an ARIMA model ARIMA(p,d,q), where d is the order of the degree of data differentiation the Equation 2.5:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{2.5}$$

Machine Learning methods vary a lot since it does not refer to the final created model, but the method of creating it focused on the data. Nevertheless, neural networks, especially using LSTM layers, are often used to predict time series due to their memory-based approach. LSTM stands for Long Short-Term Memory, which is a type of recurrent neural network (RNN) architecture that can effectively capture and utilize temporal dependencies

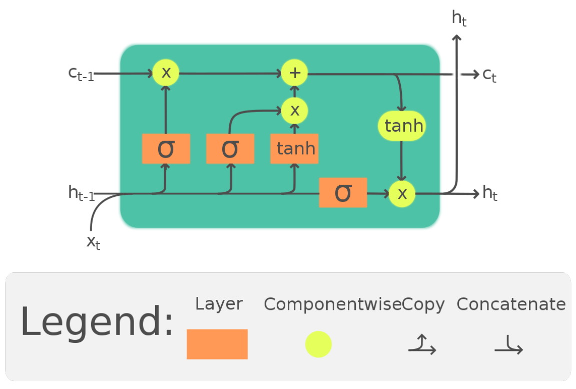


Figure 2.2: LSTM cell architecture

LSTM networks are a type of RNN architecture designed to address the vanishing gradient problem, which is a challenge in training traditional RNNs. The vanishing gradient problem occurs when the gradients used to update the network's weights during training become extremely small, leading to slow convergence or even preventing learning altogether. LSTM networks introduce memory cells, which allow them to capture and retain information over long sequences. These memory cells consist of a memory cell state and three gates: the input gate, forget gate and the output gate. The gates control the flow of information into and out of the memory cell, enabling the LSTM network to selectively remember or forget information. A representation of the cell architecture is present in Figure 2.2. The advantage of LSTM networks lies in their ability to capture long-term dependencies in sequential data. They can remember important information from earlier time steps, allowing them to make more accurate predictions or classifications. This makes them particularly useful for time series analysis, where past observations can influence future patterns[12].

Spatial Data

Spatial data refers to information where the spatial relation between the observed objects is important, and when talking about purely spatial data, time should not be a factor[13]. So all observations should be either taken at the same time or not vary between the collection time, for different places in space, or the observed. Spatial data is usually represented in two ways, using coordinates or connections. Data with coordinates will be a collection of spatial points with a specific position and the value of the observed data. While connections will create a graph, so for each data point, there will be information on the other data points connected to it.

Coordinate-based representation allows for precise localization of spatial points. Each data point is associated with a specific position on the Earth's surface, defined by latitude and longitude or any other coordinate system. This enables accurate spatial analysis and visualization, as the exact location of each observation is known. It also enables measurements of distances between points, calculation of areas and volumes, and determination of proximity between different spatial entities. These capabilities are particularly useful in tasks such as spatial clustering, spatial interpolation, and spatial modeling.

Graph-based representation captures the relationships or connections between different data points. This enables the exploration of spatial patterns and dependencies among the observations. By examining the connections between data points, one can uncover spatial interactions, identify clusters or communities of related objects, and analyze the flow or diffusion of attributes across the graph. Graphs also provide a framework for network analysis and graph algorithms. They allow for the application of graph-based techniques such as centrality analysis, community detection, and pathfinding, which can reveal valuable insights into the spatial structure and connectivity of the data.

The most common kind of spatial data is geographical data, and satellites are a big origin for collecting it. Elevation, vegetation, and temperature are examples, and these are usually represented with latitude and longitude coordinates. But other data such as road maps and airport connections, are usually represented with graphs since the connections between them are more important than the location precision.

Since purely spatial data is static in time, there's not much meaning in forecasting it, so it's more used for analysis or as extra data for timed data. This analysis can be for example, splitting the data into regions of connected data with similar values or finding the best routes in a graph. In reality, most spatial data evolve over time, and the observations can be made repeatedly to keep track of the changes.

Spatio-Temporal Data

Although most statistical analysis simplify data to be either spatial or temporal since it's easier to model and predict information that way, in the real world, most data is dependent on both space and time, which defines spatio-temporal data. In a city, all its moving parts constitute spatio-temporal data because there is relevance on which paths are being taken, what places are being passed, and what moments in time each entity passes through each point. This describes everything that moves inside a city, such as transportation, cars or public, people moving, mail delivery, and can also represent weather and other nonhuman phenomena.

Modeling spatio-temporal data usually consists of adding spatial dependencies to time-series data. For that reason, many of the previously described methods reappear as this kind of data with some tweaks to detect additional relations. Nevertheless, there are also methods modeling purely spatio-temporal techniques do exist; they are just less common. As an example, ARIMA has its spatio-temporal counterpart: Spatio-Temporal Autoregressive Integrated Moving Average (STARIMA). STARIMA considers the spatial relationships between neighboring locations and incorporates these relationships into the modeling process. STARIMA models capture the dependencies between observations at different locations and can be extended to incorporate temporal dependencies as well[14]. LSTM networks can receive spatial time as input and should be able to figure its relations without any major change compared to a network of the same architecture that only looks at temporal data.

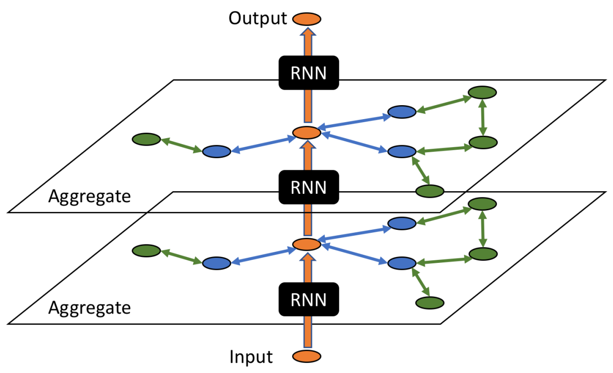


Figure 2.3: Recurrent Graph Neural Network

Gaussian Processes(GP) are regression models that can handle both spatial and temporal data. They can capture the correlation structure in the data and make predictions based on observed values at specific spatial and temporal coordinates. GPs have been successfully used in environmental monitoring, climate modeling, and disease spread prediction. Ensemble methods, such as Random Forests or Gradient Boosting, can be extended to spatio-temporal data by incorporating spatial and temporal features. Ensemble methods combine multiple models to improve prediction accuracy and can effectively handle complex relationships between spatial and temporal variables [15].

Finally, Graph Neural Networks(GNN) are a class of deep learning models designed to handle data structured as graphs, being then especially suited to spatio-temporal. GNNs operate by propagating information across the graph, allowing each node to learn representations based on its own features and those of its neighboring nodes. This propagation mechanism enables GNNs to capture complex dependencies and patterns in the data. Through multiple iterations of information exchange, GNNs refine node representations, encoding both local and global graph structures. They can be extended to handle spatio-temporal data by incorporating both spatial and temporal dimensions. And since they are neural networks, they can incorporate all sorts of layers and internal architectures such as RNN, as shown in Figure 2.3, as well as implement LSTM cells. By considering both the structural connections and the evolution of nodes over time, GNNs provide a powerful framework for spatio-temporal modeling and forecasting [16].

2.1.3 Crowd Analysis

Crowd Analysis is a spatio-temporal task relative to cities, much present in smart cities scenarios that study the movement of large numbers of people in an urban scenario[17]. This can consider many factors, such as the regions from which people are moving through, the number of people per square meter, and the speed of movement. It's a particularly hard behavior to model since it has humans at its center, and they may move in unpredictable ways. Different problems can be identified inside Crowd Analysis: Crowd Estimation, Crowd Density, or Crowd Flow are the most commonly tackled.

- *Crowd Estimation* is the task of trying to define the correct number of people in a crowd. Usually, it has a video of people moving as input and consists of counting and estimating the correct amount, but it may have other sensors as input. Nevertheless, it focuses on estimating the correct number of people from a noisy input where the people can be roughly identified, but the correct amount is hard to find.
- *Crowd Density* Forecasting predicts the number of people in a place from historical data. It can be purely temporal by having models for each studied place but also have real-time input. It can also be spatio-temporal by having a single model that captures the relations and predict the number of people in many predefined regions at the same time.
- *Crowd Flow* is the measure that determines the change in density. Based on that, predicting crowd flow is used when it is important to understand and predict the number of people and the movement between regions, not only in each region at that time. It's not uncommon for Crowd Flow models to predict density since one can be calculated from the other, but the opposite is not true. Density is not enough to compute flow.

In crowd forecasting, either flow or density, the data can be presented in many different ways. It is prevalent to divide a map into equally sized shapes for analysis. Depending on the size of these divisions, the resolution may yield interesting results that can go as far as street level. Another way of dividing the map is by significant areas with no regular shapes, possibly areas with similar characteristics, administrative areas, or determined by the sensors. Irregular areas may be inferred by regular shapes, with a good resolution clustering this area of interest before doing the flow analysis may produce better models. The inverse is not possible, though. Having irregular shapes, it can be assumed that they have a regular density inside them, but the information would be lost when extrapolating it to regular-sized divisions. Irregular shapes may lead to fewer connections and a better graph representation that may be better when using graphs for solving the problem.

Crowd Analysis is always a Spatio-temporal problem since the density and flow compare more different regions in space, which can have connections between them, and this changes over time. Some ways of retrieving crowd data are using communication networks, like public Wi-Fi that pedestrians connect to, mobile telephone data or Bluetooth; from apps that collect GPS and other information; registers from physical places like metro stations; and using cameras that can count the number of people that pass by. Each method has its peculiarities and precision, and multiple methods can be used together, compared, and merged to create more accurate data. One practical everyday use of Crowd Analysis is Google Maps busy areas functionality which highlights areas with high density of people and show the estimation of the number of people over the day, as shown in Figure 2.4

This thesis will focus on Crowd Density Forecasting, which can use many of the previously discussed methods. The biggest decision for choosing a method is the inclusion of the spatial component. It is not uncommon in previous attempts at modeling Crowd Density in cities to have

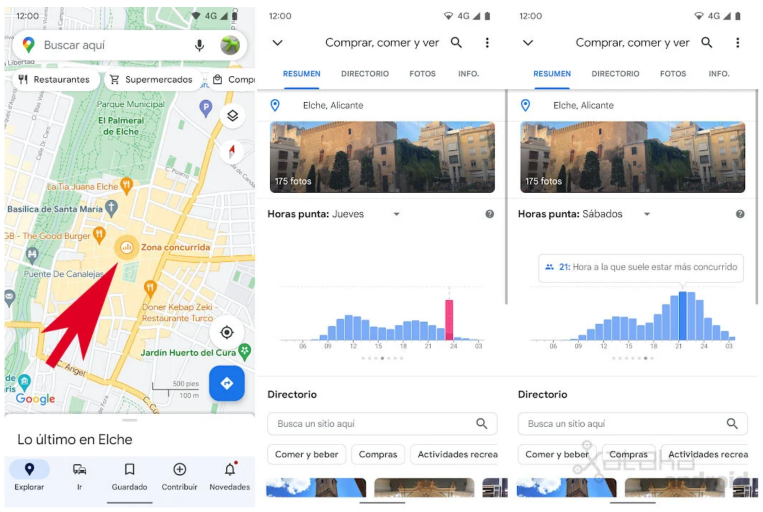


Figure 2.4: Google Maps Busy Areas functionality

a single model for each region. This would favor time-series models such as ARIMA. Not having the spatial component may result in the loss of information and create a less accurate model, so LSTM networks and GNNs could be used. This comparison between including or not the spatial component is discussed in more detail in section 2.3 and chapter 3.

2.2 Empirical Evaluation

In the context of Crowd Density Forecasting, Empirical Evaluation consists of evaluating each model so that the factor that differentiates one from the other is clear and the only changing variable. By evaluating the differences in the results of each experiment, the factors that influence the final results should become clear and bounded in the reality of the experiments, thus empirically based. From the empirical observations, conclusions can be taken about the phenomena and extrapolated to the reality of the modeled original data.

Even with good precision in the data collection, it is always necessary to interpret the data and the predictions correctly to understand it in all of its details. Some factors that need special attention are the proportion of the actual population and the population detected by the means of collection. For example, a Wi-Fi hotspot will only detect a part of the smart device users, so it would be necessary to factor in the percentage of device users that will not connect, of people that do not have smart devices, the connection of more than one device by person and these proportions to the overall population.

Census data is an essential data source for knowledge about the population and place that can result in a higher quality model. But also for validation, once the ratio between the real population and observed data is known, the census data can be compared. In a good crowd density forecasting model, the number of people in a region at a time that is known that most people are at home, such as late at night, should match approximately with the population.

Error metrics are the primary way of comparing models, with many different metrics being used, each metric with its own benefits related to different models and data. The most common metrics in the literature are MAPE (Mean Absolute Percentage Error), MAE (Mean Absolute Error), MAXAE (Maximum Absolute Error), MSE (Mean Squared Error), and RMSE (Root Mean Squared Error). These metrics differ in aggregating and summarizing errors between actual and predicted values. They can be more or less appropriate depending on the particular forecasting problem. [18]

- MAPE: measures the average percentage error between actual and predicted values. It is used in cases where the magnitude of the error is more important than its direction. It provides a clear and intuitive measure of the average error in percentage terms, making it easy to understand and interpret. However, it can be misleading in cases where the actual values are minimal since even minor absolute errors will result in significant percentage errors.

- MAE: is the average absolute error between actual and predicted values. It measures the overall average deviation of the forecasts from the actual values. It is simple to calculate and interpret and provides a robust measure of the average error, regardless of the direction of the error. On the other hand, it does not provide information on the magnitude of the error in percentage terms.

- MAXAE: calculates the maximum absolute error between the actual and predicted values. It measures the worst-case deviation of the forecasts from the actual values. It indicates the most significant error in the prediction, which helps identify outliers or extreme cases. Its weakness is that it is sensitive to a single outlier and may not represent the overall accuracy.

- MSE: measures the average squared error between the actual and predicted values. It measures the overall average deviation of the forecasts from the actual values, with a higher weight given to more significant errors. It provides a measure of the forecast's overall accuracy that considers the magnitude of the error. However, it is pretty sensitive to outliers.

- RMSE: is the square root of the MSE, and it measures the overall average deviation of the forecasts from the actual values in the same units as the actual values. It provides a measure of the forecast's overall accuracy that considers the magnitude of the error and is easy to interpret intuitively. Its weakness is that it can be sensitive to outliers and can be challenging to compare different forecasting problems with varying measurement scales.

In spatio-temporal forecasting, the accuracy metrics can be more complex due to the data's temporal and spatial dependencies. A model that predicts many zones simultaneously may be very accurate for some zones and not that much for others. For that, it is important that be careful when using these metrics to extract the correct information from the empirical data. In the same way, an average error may not reflect the model at all in its particularities. The objective of the error comparison should always be taken into consideration, not only the values themselves. If the

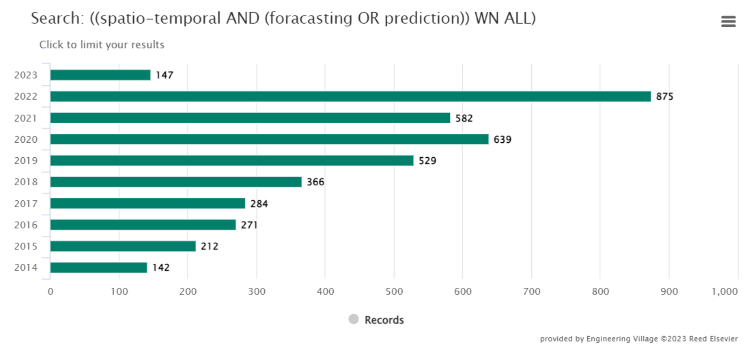


Figure 2.5: Publications by year for Spatio-temporal forecasting

values differ a lot from one model to another, it can mean that it is better, but if the difference is small, one may be better at predicting all regions in space, and the other may be very good in a single region which brings the values down.

The empirical evaluation should use the metrics of the model to find the best one and as evidence to make bigger conclusions about the data. This means observing that a variable makes the model better or worse and looking at the factors that could explain the influence of that variable. For example, regions that constantly show higher error metrics may have some factors in common that may explain why they are harder to model than others.

2.3 Related Work

Crowd Density Forecasting is a problem that can be tackled using the prediction algorithms discussed above for either time-series or spatio-temporal data. Many approaches have been used in the past for the matter, with different results for each case. The topic of this section is the presentation and discussion of related work using the mentioned algorithms for solving such problem and their specificity, first, by presenting works that present insights into the challenges related to Spatio-temporal forecasting. Then discussing works related to crowd forecasting.

2.3.1 Spatio-Temporal Forecasting

Spatio-temporal Forecasting is a task not only related to smart cities, with a great variety of applications with increasing interest in recent years. The recent advances in neural networks helped to make research in the field grow since some architectures can detect spatial and temporal dimensions better than previous methods, which is the hardest task in this kind of forecasting. When searching in the Engineering Village Platform for 'Spatio-temporal AND (forecasting OR prediction),' it is possible to find 6248 records, the oldest being from 1977, which are mostly statistical analyses of natural phenomena such as weather and earthquakes. But the number of publications has been much greater in recent years, mostly involving neural networks and other machine learning approaches. Figure 2.5 shows the number of publications found by year.

Traffic forecasting is the most common use of spatio-temporal forecasting, being the most present in the most recent results on the search engine. Nevertheless, two other topics appear in reasonable numbers: Weather events forecastings, such as solar irradiation, wind patterns, and natural disasters; The other one is pandemic spreading, mostly focused on the recent COVID-19 pandemic. Through a quick analysis of the results, it is possible to identify the most common and trending methods for forecasting, from which can be highlighted: Spatio-temporal AutoRegressive Models and similar Time-Series methods adapted to handle spatial relations; and Neural Networks, especially Graph Neural Networks with varying architectures, Convolutional Networks, LSTMs, and Gated Recurrent Units.

Graph Neural Networks are a particularly growing method for spatio-temporal data due to the graph structure shown in the search results as discussed in [19]. GNN was first proposed in 2009, and their interest has only grown. It has been used extensively for traffic flow prediction, such as in [20], [21], [22] and [23]. However, it is used in other spatio-temporal contexts, such as crowd flow in [24], [25], and [26]; but also for the other problems already mentioned such as weather.

When dealing with spatio-temporal data for forecasting, some precautions are needed when processing it due to its temporal and spatial dependencies. [27] exposes that it is expected that when dealing with this kind of data, the models produced are either not spatial or temporal aware because of the nature of many algorithms to treat the dependencies as constant and focus on the other. It points out that most statistical models are not, by default, ready to deal with spatial data. Hence, the solution is to create many models for the different spatial parts or to average them out, thus losing essential features for forecasting. While many deep learning models that are commonly used have the same parameters for the time series of different locations, assuming that they are similar. Because of that, it proposes a data-focused method for spatio-temporal awareness using attention to turn models that miss this relation.

Additionally, [28] focuses on how data must be split into test and train data to accurately evaluate machine learning performance and show a comparison between methods. Spatio-temporal data cannot be divided without considering the relations between neighboring points in time and space. The removal of these points can lead to a significant loss of information. Usually, splitting time can be done by deciding a moment to separate training from test data, while splitting spatial data has more freedom. Still, points may need other influence points to show relations. The two main methods that exist for splitting are out-of-sample (OOS) and cross-validation (CV) strategies. OSS procedures include Time-wise holdout involving choosing a single split-point, and time-wise Monte Carlo involves randomly generating several split-points. These divisions may vary between iterations. CV strategies involve dividing the data into folds and using each fold as a test set while the rest of the data is used as the training set. Standard cross-validation randomly assigns instances to folds, ignoring time and space dependencies. Time-sliced CV considers time-slices but ignores the spatial dimension, while spatial block CV considers spatial blocks but ignores the temporal dimension. Spatio-temporal block CV takes into account both spatial and temporal dimensions. The study recommends avoiding standard CV and holdout methods to consider space and time correctly. The rest of the methods resulted in similar errors between them, without a preferred one for all cases.

The spatial subdivision is another split important to be careful of when dealing with spatial data. Human divisions may not always be the best way of dividing geographical areas for machines to model, especially when dealing with graphs, where the shape of each studied region is not considered; the aggregation or division of nodes may produce different accuracy models. [25] compares the use of irregular subdivisions in opposition to grid-based data and proposes a CNN solution for it. Although convolutional networks may seem ideal to grid-like data since they can be directly translated into matrices which is where the convolutions can be applied, the proposed solution using CNN achieves better results than the ones it was compared against.

[29] proposes a way of subdividing an area into subregions based on busy areas from GPS. It uses DBSCAN, a clustering method that can identify the busier core locations than others that will be the centers of the subdivisions, then compare them to their neighbors to cluster them into bigger nodes. Ultimately, the map is subdivided based on the circular areas around each core location aggregated to its neighboring less active areas, generating an irregularly shaped map of regions. Since the generated subdivisions are constructed based on the density of people, it should be a good way of clustering areas for forecasting tasks.

Although spatio-temporal forecasting has similar characteristics for all sorts of problems related to this kind of data, each specific application may have its peculiarities. [30] compares approaches for this kind of forecasting in the context of smart cities. It divides smart cities' most common forecasting applications into three: renewable energy forecasting, load demand forecasting approaches, and Traffic characteristics forecasting approaches. About the latter, it shows that spatio-temporal approaches outperform regular ones. Much of the literature is based o Auto-Regressive models such as ARMA and ARIMA. Still, machine learning methods appear too, such as ANN, SVM, Fuzzy Logic, GA, Bayesian Networks, MC, and kNN, and have shown a superior ability to make predictions compared to statistical counterparts. It also points out a lack of methods to predict more uncommon behaviors, such as the ones on weekends, with many studies using only weekdays to create the models since it is more constant and easier to extract trends.

As the most studied spatio-temporal forecasting problem, traffic flow and density are similar to crowd density, the problem tackled in this dissertation. While both crowd flow/density forecasting and traffic flow involve predicting the movement of entities, they differ in terms of the entities being studied and the characteristics of the environment. Crowd density forecasting primarily concerns human behavior, considering factors such as pedestrian flow, crowd dynamics, and individual interactions. It aims to estimate people's density, distribution, and movement patterns within a specific area. On the other hand, traffic flow analysis focuses on vehicular movement, considering factors like traffic congestion, signal timings, and road capacity. The methods and models used in crowd density forecasting may incorporate social dynamics, crowd psychology, and individual behavior. At the same time, traffic flow analysis typically relies on traffic engineering principles, traffic flow models, and transportation infrastructure design. Both fields contribute to urban planning, public safety, and resource allocation, but they address distinct aspects of movement and congestion within different domains.

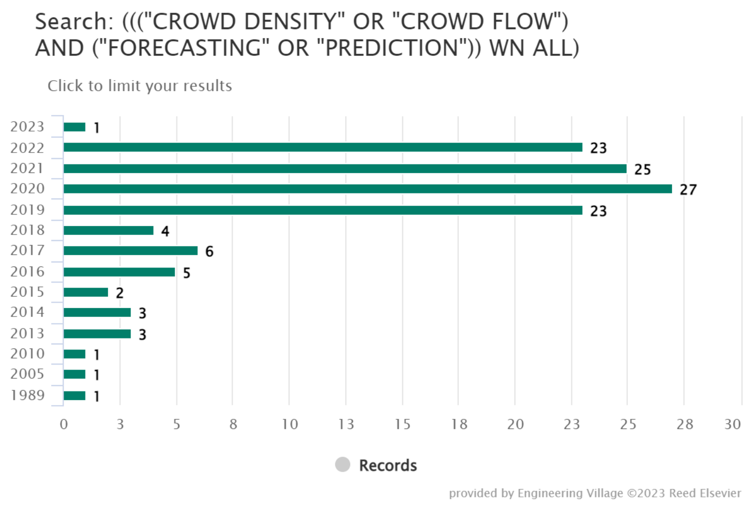


Figure 2.6: Publications by year for Crowd Forecasting

2.3.2 Crowd Forecasting

When focusing on only Crowd Forecasting, finding articles is a harder task, and the search results are much more scarce compared to traffic flow and spatio-temporal forecasting as a single search. Searchin in Engineering Village, with keywords "Crowd Flow," "Crowd Density," "Prediction," and "Forecasting" as *("CROWD DENSITY" OR "CROWD FLOW") AND ("FORECASTING" OR "PREDICTION")* it returns 125 results most from 2019 onward with a few older exceptions. Many of these results are about crow density estimation, though. The oldest one in the search with these keywords is about crowd estimation from images, which are distances from the subject. The first one that is really about crowd forecasting is [31] from 2016, which uses mobile data from smartphones to predict the number of people in critical areas to warn of large gatherings. This was done by using a Markov model to predict the future position of people based on their current positions and count the number of people that will move to a region. Figure 2.6 shows the number of publications found when searching. It is important to note that this graph is from the beginning of 2023, thus the single publication for the year.

Ideally, crowd forecasting should always be a spatio-temporal task, either for crowd flow or density. Nevertheless, many times the spatial dimension is not considered. This could be for some different reasons, first for making a simpler model that may be trained more easily, and the other reason is due to technical limitations, such as models that run on in-loco hardware and may not have the computational power of a big model or even the communication with other sensors for exchanging the necessary information. These kinds of models usually consist of time series formed from individual sensors scattered through the city that collect the number of people passing by. [32], for example, counts the number of people connecting to different Wi-Fi to collect data and properly treats it not to collect wrong information, such as counting people twice. And trains an LSTM NN that can forecast the density in future moments.

Another example is [33], which uses a random forest regressor, using data spatio-temporal data collected from Bluetooth low energy(BLE) at street level together with weather, calendar, and recent trends in the city of Tokyo. It focused on social distancing since it was developed during the pandemic of COVID-19. Some BLE devices were installed around shopping districts in Tokyo, collecting data from how many devices passed by over time. A random forest regression model was trained for each of the nine locations with the best data since some locations were removed due to malfunctions in the hardware that caused missing and noisy values. The main goal of using a random forest is to let the algorithm identify the most critical features among those that were given as input. The average accuracy was 84%. It was compared to an LSTM model having very similar results.

The data's origin significantly impacts the extra features that can be extracted from it and the best ways to handle it. Mobile singular data, as seen in [34], that is, there are individual entries for each user, makes it possible to extract information about the user behavior and the zones based on the time it spends in each zone or recurring patterns between users in the same zone, which can be very useful for a model. However, this is not always possible due to different regulations for each country and the nature of some sensors. As pointed out by [35], European Union GPDR(General Data Protection Regulation) inforces that mobile data must be aggregated.

Using tokens that count devices passing through, like Bluetooth low energy used by [33], can lead to difficulty in capturing the spatial relations if the tokens are not well positioned or if there are not as many. It is also fairly standard across the literature to use traffic data to represent the crowd as in [24], most articles compare the proposed methods to older ones for traffic. This relation is not always direct tough, and it is easier to extrapolate density and not necessarily flow. In fact, [24] hints about the importance of detecting indirect connections when creating a graph, like the ones that represent subway stations. Such ways of people getting from one node to another may not be extractable from traffic data.

The kind of model trained enables different behaviors in its final application. Statistical models, generally traditional regressions or AutoRegressive, need to be trained every time new data is collected and usually lose accuracy the further from the trained data it gets. On the other hand, neural networks, especially recurrent ones, and LSTM, usually are trained to predict the next moment in time based on a window of previous moments. This does not mean that NN would not profit from being retrained with new data but enables trained models to work in real-time more easily with data being collected when the forecasting is being done. That is the case of [36], which collects data from GPS to create an RNN with GRU focused on predicting individual movement sequences. It divides the space into irregularly shaped cells and trains the model to predict the next cell based on the sequence of previous cells that it has been in the previous time steps. It is used to predict in real-time the next position of a person in an event for controlling the formation and the movement of crowds.

How the data is organized spatially also plays an important role, mainly in the models applied. When there is information for equally sized grids, the data is analogous to images, with each pixel being an area and the color representing the density. This comes in opposition to graph-structured, where the relation between the areas needs to be explicitly represented. The similarity to images suggests the use of convolutional algorithms that have been used with much success for finding patterns in matrix-like structures.

[34] proposes an algorithm with a density grid as input, analogous to videos, with the images representing the density for each pixel as a location and evolving over time. The model utilizes a pyramid ConvLSTM and an attention mechanism to learn how each hidden state generated by the pyramid ConvLSTMs will match the next prediction frame. In addition, the model does not use additional data sources such as Point-of-Interest (POI) data and related processing modules. Instead, it takes the entire spatial domain as the feature rather than a certain mesh grid or a couple of mesh grids. It evaluates the model using MSE, RMSE, MAE, and MAPE for a dataset of GPS logs from Japan for both Tokyo and Osaka. The logs are aggregated into the 4D data to be input into the algorithm. It was compared against other models from previous papers, with exceeding results.

On the other hand, [25] proposes using graphs. It uses mobile location data from a mobile communications company in Suzhou, China. Since the data had the information time and location for individual uses, though anonymized, the activity type was extracted from the location behavior of the user. So for each area, there was not only the density count but the count of people doing different kinds of activities for each moment in time. The activities inferred were home activity, work, and other, based on the location each user was recurrently at night and during working hours in the week. The most innovative contribution was using irregularly shaped areas, which is uncommon in other attempts. It proposes using a graph convolutional network composed of three blocks, an attention-based feature fusion block, a spatial-temporal convolution block, and the output block. There were 23 areas in the study, with the moments in time and activities. Additionally, as input was given, the day of the week, the information if the day was a holiday and the weather of the moment. The model was validated against six other methods: Historical average (HA), SARIMA, Vector-AutoRegressive(VAR), KNN, LSTM, Spatio-temporal Graph Convolutional Network (STGCN), a deep learning framework that predicts traffic flow on road networks. Between which STCGN had the best results. But compared to the proposed method was worse,

And [24] presents a system to transform datasets with regular grid data, such as the ones in [34] and [37], into graphs to use graph neural networks. To do so, it uses the Euclidean distance and the Pearson correlation between the distances of two cells to create a KNN directed graph. This is done to consider some geographical features of a city that cause large areas with zero counts of people on a grid dataset, such as rivers. Once the graph is constructed, it uses a graph neural network with LSTM and self-attention mechanisms to predict the flow. It was tested using two datasets, TaxiCQ and BikeNYC, of taxi movement in Chongqing, China, and bike rental in New York. And compared with many consolidated autoregressive models and state-of-the-art neural network architectures proposed by other articles, obtaining the minimum RMSE and MAE for the taxi dataset and the minimum RMSE and second minimum by one decimal point in the bike dataset.

It is noticeable that even though most of the literature proposes machine and deep learning methods for forecasting crowds, they are always evaluated against statistical methods that were more common in the past. [35] compares statistical models and deep learning models for solving crowd density. It uses an aggregated mobile location dataset, similar to the one used in this thesis, but with regular square divisions of the city of Modena in Italy. It creates one model per cell in the grid for all forecasting methods. Then the models are trained and evaluated using MAPE, RMSE, and MAXAE. The methods tested are, for time series statistical models: ETS, AR, ARIMA, and Prophet, and the deep learning models LSTM, CNN, MLP, and CNN-LSTM. Between those, the deep learning models had a better performance overall. The best deep learning method was cnn-lstm, and the best statistical method was ARIMA. ARIMA showed better results than CNN-LSTM in MAPE, and RMSE metrics, even though the other statistical methods did not against the simpler deep learning. But CNN-LSTM outperformed ARIMA in MAXAE.

Additionally, the areas were separated into high and low variability cells, where high variability means that the zone has more chance of having abrupt changes in density while low variability follows simpler patterns. This was used to check the accuracy of deep learning and statistical models when dealing with more complex patterns. Though ARIMA performed better, the deep learning methods had a more considerable advantage in predicting irregular patterns, with cnn-lstm still being the best.

Although forecasting flow and density are similar, constructing one from the other is not always possible. It is impossible to know the density from the pure flow since there may be a continuous density in an area unaffected by the flow. Extracting the flow from density is possible but not always straightforward. Movements through a region faster than the time scale may suggest a connection between unconnected locations. The number of neighbors for each area will make the extraction increasingly complex as it grows. For that, [38] proposes a deep learning method for extracting this information with a significant improvement over the tested baselines. This can be used in scenarios where an algorithm may benefit from having the historical flow data for future forecasting, but only density is provided.

The training time to train different models is an important measure that varies greatly from model to model and is not always taken into consideration in most studies. [37] proposes a deep learning convolutional neural network approach focused on optimizing time efficiency while maintaining high accuracy. The model implements a k-NN layer which was responsible for the biggest reduction in training time when compared to similar models without this layer.

2.4 Gap Analysis

A gap analysis table(Table 2.1) is built based on revising the cited articles and their characteristics. The columns present these characteristics, while each row represents a study and its features. The Algorithm column refers to the algorithm used to generate the forecasting model; the origin relates to the data origin, the collection device/method; the Time span is the time between the first and last entry in the database; the Zone refers to how the studied area was divided, either as a graph or a grid, some graphs have a fixed number of nodes others are that are generated algorithmically vary, the one that says only '9 locations' means that there were nine models generated for each of the locations; the additional parameters are the other features considered when making the model other than density or flow historical data; finally, the type refers to what is the output of the model trained, that can be the crowd density or flow.

Paper	Algorithm	Origin	Time span	Zone	Additional Parameters	Type
[25]	GNN	Mobile	3 Months	23 Zone Graph	Weather, calendar and activity type	Density
[33]	RFR	BLE	7 Months	9 locations	Weather and calendar	Density
[34]	ConvLSTM	GPS	3 Months	Grid	None	Density & Flow
[24]	GNN	GPS	6 Months	Graph	None	Flow
[39]	GNN	Government	13 days	Graph	Zone type	Density
[37]	NN	Mobile	3 Months	Grid	None	Density & Flow
[40]	MFD	Mobile	2.5 Months	Grid	Zone Type	Density
[41]	GNN	Mobile	3 Months	Graph	None	Flow
[32]	LSTM NN	Wi-Fi	8 days	2 Locations	None	Density
[42]	GNN	Mobile	3 Months	Graph	None	Flow
[43]	CNN	GPS	3 Months*	Grid	Weather and Calendar	Flow
[36]	RNN-GRU	GPS	18 hours	Irregular Shaped	None	Cell Sequence
[44]	CNN-GRU	App	26 days/42 days	Grid	None	Density
[45]	GNN-CNN	App	1 week/6 Months/2 years	Graph	None	Flow

Table 2.1: Gap Analysis Table

*Many datasets were used and compared. Three months is an average.

The table enables some observations about how different models behave and their properties. Due to the nature of the data, the articles in which there are many models, one for each location, cannot use graph-based approaches since they are space agnostic. There is a direct correlation between the use of convolutions and grid data. Indeed just one of the models that used convolutions does not have grid data. As explained before, this is due to the direct translation between grid data and numerical matrices, for which CNNs are much used.

Although it is possible to create some logical association between the prediction of either flow or density for each data origin and kind of algorithm, none can be seen in the table. GNN could be assumed to be better for computing flow since the graph structure is explicit, while traditional NN would treat it as a simple value, but this relation is not present in the table. However, when looking at the data origin, even though there is no direct relation overall, there are data collection methods that may lack the precision to capture the flow and may count only the density, such as Bluetooth tokens.

It is possible to see that most datasets do not cover periods of much longer than half a year. And that additional features are usually restricted to weather, calendar, and zone type. No study was found that compared the impacts of different additional features on different forecasting models. And none that adds geographical features such as elevation and green coverage, thus leading to the main focus of this dissertation.

Chapter 3

Problem and Solution

This chapter provides a comprehensive and detailed exploration of the crowd density forecasting problem. It dives into the intricacies of the data, discusses its unique characteristics, and elucidates the collection and treatment processes employed to use it. Furthermore, it offers a thorough examination of the methodology utilized in the creation of each model, offering valuable insights into their respective advantages and the challenges encountered along the way.

3.1 Problem Description

The problem to be tracked consists of two parts, each may be divided into several tasks. First, to develop machine learning models for crowd density forecasting, using as a case study predetermined regions of the city of Vila Nova de Famalicão in North Portugal. And secondly, to empirically evaluate these models to find the best model for predicting the density of people for future moments, as well as detecting the level of influence of external features in the prediction, namely geographical and weather features.

The city of Famalicão is a mid-sized city in the Ave sub-region of Portugal, between the second and third bigger cities in the country, Porto and Braga, respectively. As of 2021, it was the twentieth biggest city by population, in the country[46]. Being close to two bigger cities, it has a big part of its population commuting daily to them, having access to main highways and a train station that connect it to the rest of the country. The city also has its industries, mainly textile, automotive, agro-food, and metalworking, and for commuting inside the city, it offers 44 bus lines and a great offer of 27km of bike lanes for the citizens to move internally[47].

The municipality's administrative boundaries are much larger than the urbanized area, giving it a big, green, but unpopulated area. Also, seven parks are present around the city, and citizens have a good infrastructure to utilize the places the city has to offer, such as bike routes. And many activities that the citizens can do in their free time and may influence their movement patterns: six museums and many historical and cultural sites such as churches, bridges, and aqueducts.

Since 2019 the city has had a Smart City project, in which it collects data with the help of companies to provide public access to information about the city and help and encourage the

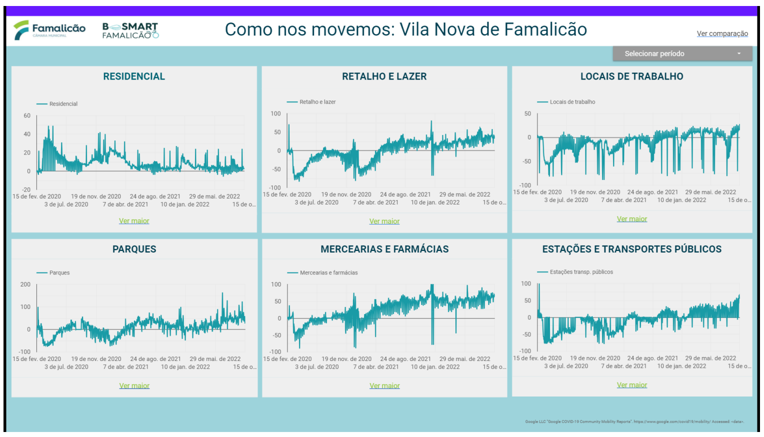


Figure 3.1: 'How do we move' Famalicão B-Smart page

development of smart projects[48]. The portal B-Smart[49] has twenty-one categories from which can be accessed information about the city infrastructure: housing, transport, health, economy, tourism, and culture. It also provides insights into city analytics in mobility, with data and graphs about the inhabitant's from which an excerpt can is shown in Figure 3.1.

This dissertation aims to provide a funcrionality in the field of smart cities: the crowd density forecasting. With that goal in mind, four machine learning models are created and compared, one using Autoregressive algorithms, one recurrent neural network with LSTM layers, a graph neural network also with LSTM layers, and one using AutoML. Each one will be described in more detail in the following sections. These four models were chosen due to past research showing them to be very effective. The models are trained on the data and tested on a part of it to measure its ability to predict a future date by using unknown data. The model should be able to find the density of people for the following moments after the ones it was trained over, with good accuracy overall. And between all the models, the one with the best performance in predicting future densities should be selected for use.

The empirical evaluation is done on the fact that the best model performs best. By understanding which characteristics it has to better suited to the crowd density forecasting problem. The models are used to evaluate the influence of the features present as input empirically, but checking the impact on the resulting model when including or not each feature. Finally, the results are extrapolated using the city knowledge, such as what real value would the predicted value represents, based on the proportions of users and citizens, and what the included features mean in the city context, for example, if green areas are good for the model, how does it change forecast in places close to a park. The next sections discuss the processes to tackle the problem. First, a description of the data, followed by the operations done over it: preparation, analysis, and clustering.

3.1.1 Data Description

The main data is from a major Portuguese telecommunications company from which two datasets were given to train the machine learning models.

First, a people count dataset, where the city of Famalicão is divided into a hundred and sixty-nine statistical subdivisions, and contains the number of connected smartphones to the company network over five periods of the day: from midnight to six in the morning, from six to noon, from twelve to two in the afternoon, from two to seven, and from seven to midnight. The data differentiates the count for national, international, and all people based on the origin of the SIM card connected. And the registries are from May 2021 to April 2022. So each dataset row contains date, zone, and counts (one for each time period for national, international, and all SIM cards). An excerpt from the database can be seen in Table 3.1

cod_time_day	sccode	val_all	val_all_00_06	val_all_06_12	val_all_12_14	val_all_14_19	al_all_19_00	val_est	val_est_00_06	val_est_06_12	val_est_12_14	val_est_14_19	val_est_19_00	val_nac	val_nac_00_06	val_nac_06_12	val_nac_12_14	val_nac_14_19	val_nac_19_00
2021-05-03	SC031224004	82	0	35	17	43	13	0	0	0	0	0	0	82	0	35	17	43	13
2021-05-03	SC031247003	130	9	56	39	74	48	0	0	0	0	0	0	130	9	56	39	74	48
2021-05-03	SC031248006	104	9	48	35	61	35	0	0	0	0	0	0	104	9	48	35	61	35

Table 3.1: Example from Crowd Count dataset

In the table, *code_time_day* refers to the day of the entry; *sscode* is the code of the statistical zone; *v_all* is the crowd density for the whole day in that region; *val_all_00_06*, *val_all_06_12*, *val_all_12_14*, *val_all_14_19* and *al_all_19_00* are the densities for the region in the interval's of time described before; *val_est* is the density of foreign SIM cards in that day; *val_est_00_06*, *val_est_06_12*, *val_est_12_14*, *val_est_14_19* and *val_est_19_00* are the counts for foreign SIM cards during the time intervals; *val_nac* is the density of Portuguese SIM cards during the day; and *val_nac_00_06*, *val_nac_06_12*, *val_nac_12_14*, *val_nac_14_19* and *val_nac_19_00* are the counts of national SIM cards for the time intervals.

Additionally, an origin-destination dataset(Table 3.2) was also provided. It has the number of people moving from one region to another during hourly intervals. The movements are from a single day, the 16th of April 2020. The origin-destination dataset has the counts for regions outside the city, showing the number of people leaving and entering. But since the data time range does not match the density, since it is just for a single day, it could not be used for studying the crowd flow. Also, there were too many wrong values, but they could be used to infer the connections between the zones, as will be described in detail in future sections.

Pais	Periodo_Horario	Dia	Codigo_Seccao_A	Codigo_Seccao_B	Entradas_BA	Saidas_AB	Permanencias_AA_ou_BB	Codigo_Concelho_A	Codigo_Concelho_B	Concelho_A	Concelho_B
Outros	2022-03-16 00:00:00 UTC	2022-03-16	SC030216001	SC031237001	6	8	0	CC0302	CC0312	Barcelos	Vila Nova de Famalicão
Outros	2022-03-16 00:00:00 UTC	2022-03-16	SC030219002	SC031227003	6	8	0	CC0302	CC0312	Barcelos	Vila Nova de Famalicão
Outros	2022-03-16 00:00:00 UTC	2022-03-16	SC030219002	SC031227004	6	8	0	CC0302	CC0312	Barcelos	Vila Nova de Famalicão

Table 3.2: Example from origin-destination dataset

In the table, *Pais* is the country of the count, meaning that the number stands for the number of SIM cards from that country; *Periodo_Horario* is the hour of the day for the count; *Dia* is the day; *Codigo_Seccao_A* and *Codigo_Seccao_B* are the codes for the A and B statistical regions; *Entradas_BA*, *Saidas_AB*, and *Permanencias_AA_ou_BB* are the numbers of people moving from A to B, B to A and staying in the same zone, respectively; *Codigo_Concelho_A* and *Codigo_Concelho_B* are codes representing the cities A and B; and *Concelho_A* and *Concelho_B* are the names of the cities.

The company has 29.3% of the mobile communication user base in Portugal, and from the overall population, 95% has access to mobile data [50]. Considering an equal distribution of SIM card users and users from the company, and overall statistical regions, the data represents 27.8% of the actual population, so the number should be multiplied by a factor of 3.6 times to approximate the real values of the population. The population of the city and its regions is another dataset used(Table 3.3), and it was obtained through INE using the 2021 census data [51]. The real population can be compared to the count data at night of each region, multiplied by the correcting factor, and also to the forecasted count for the region to check its quality. Usually, this is done using the values, either historical or predicted, from the nighttime, where it is assumed that most of the region's population would be at home, then the numbers should be similar.

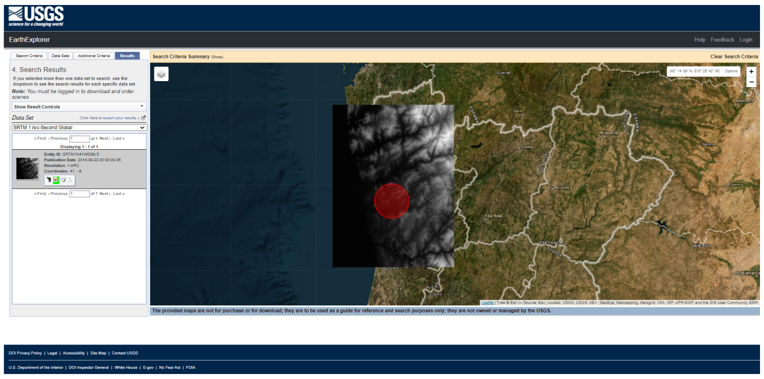
Neighborhood	Total population
Abade de Vermoim	437
Antas	6925
Avidos	1742
Bairro	3598
Bente	925
Brufe	2231
Cabeçudos	1466
Calendário	11667

Table 3.3: First rows of Famalicão Neighborhood Population Dataset

neighborhood	sccode
BAIRRO	SC031204001
BAIRRO	SC031204002
BAIRRO	SC031204003
BAIRRO	SC031204004
BAIRRO	SC031204005
BRUFE	SC031206001
BRUFE	SC031206002
BRUFE	SC031206003
CASTELÕES	SC031210001
CASTELÕES	SC031210002

Table 3.4: First rows of Neighborhood to statical zone Dataset

The regions of the city do not match the statistical regions used by the telecommunications company to divide the city. But the company provides a mapping of the statistical region code to the administrative neighborhood divisions of the city(Table 3.4) that is used to aggregate the



4. Search Results

If you selected more than one data set to search, use the dropdown to see the search results for each specific data set.

Note: *You must be logged in to download and order scenes*

SRTM 1 Arc-Second Global

Displaying 1 - 1 of 1

Entity ID: SRTM1N41W009V3
Publication Date: 2014-09-23 00:00:00-05
Resolution: 1-ARC
Coordinates: 41 , -9

The provided maps are not for purchase or for download; they are to be used as a guide for reference and search purposes only; they are not owned or managed by the USGS.

Figure 3.2: NASADEM_HGT dataset in EarthExplorer

statistical regions in different moments for many reasons, such as for the comparison with real values.

External features datasets were collected for calendar information, weather, elevation, and leaf coverage:

- The calendar information was easily found online by searching online for Portuguese national holidays, and the data was extracted from [52].

- The weather dataset is from [53] that offers historical data aggregated from weather stations, aircraft, buoy, radar, and satellite observations for any location in the world. The city of Famalicão has a weather station, so the weather data will be the same for all regions and is used because it can still be useful since it varies for the time intervals. The site offers many weather metrics that will be tested: temperature, humidity, wind, and rain.

- Both leaf coverage and elevation data were obtained from earthexplorer.usgs.gov(Figure 3.2), which is a website operated by the United States Geological Survey (USGS). Earth Explorer provides access to a wide range of geospatial data, including satellite imagery, aerial photography, and other remote sensing datasets. Users can search and download various data products for scientific, environmental, and land management purposes by selecting a region in the map and selecting one of the available datasets. Leaf coverage was extracted from the MCD15A2H v006 dataset[54] that offers a Leaf Area Index (LAI) with a 500-meter pixel size, which indicates the percentage of that area covered by vegetation. The elevation dataset was obtained from NASADEM_HGT[55], which provides a 1 arc sec binary image representing the elevation of a region of the globe. Since the geographical location of the statistical regions is unknown, the neighborhoods were used to determine each neighborhood's average leaf coverage and elevation, using a process best described in subsection 3.1.2

.

3.1.2 Data Preparation

The process of preparing the data and merging it was done in a Jupyter Notebook with a runtime of Python 3.10. The datasets were managed using the Pandas library, the most widely used library for dealing with tabular data. For dealing with the shapes of the neighborhoods and doing polygon operations, the Shapely library was used.

Before feeding to the models, the data needed to be prepared.

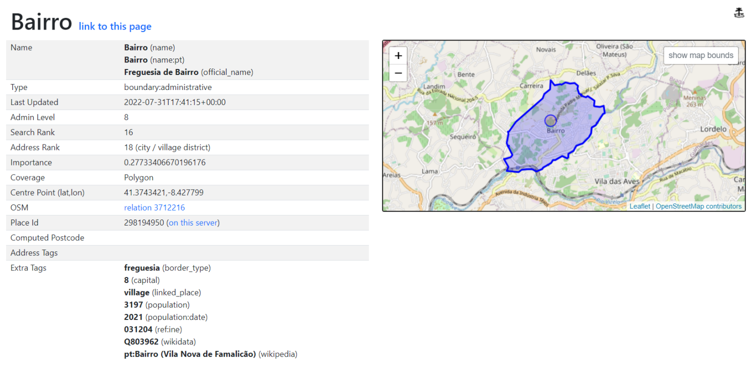
External Geographical Data

First, the external geographical features were fetched from earthExplorer in formats that were not usable in Python. The software QGIS(Quantum Geographic Information System)[56], an open-source software application designed for handling, analyzing, and visualizing geospatial data, was used for that process. QGIS supports various data formats, including vector and raster data, and allows users to perform tasks such as data acquisition, editing, geoprocessing, and map creation. With it, it was possible to add the datasets of leaf coverage and elevation as layers and see that they matched the map of the city of Famalicão. Then, using the tool raster to vector, the values were converted into a new layer with floating number values that could be used. Then finally, it was possible to select the vector layers and export them as CSV files while adding their latitude and longitude information. Ending up with a correlation between points in space and the numerical values of the features.

Neighborhood Data

Then to correlate the values with the neighborhoods, the geographical boundaries of each neighborhood are necessary. OpenStreetMap (OSM)[57] is a collaborative and freely accessible mapping project that aims to create a comprehensive and detailed map of the world. Unlike traditional mapping providers, OpenStreetMap relies on a community-driven approach, where individuals from around the globe contribute to creating and maintaining the map data. Volunteers use GPS devices, aerial imagery, and other sources to collect geographic information, including roads, buildings, landmarks, and points of interest. OSM is able to draw the boundaries of many regions, and it does so by having a polygon dataset that can be publically accessed through [58], where a polygon ID can be used to find the latitude and longitude of the points that define a region registered in OpenStreetMap(Figure 3.4). The polygon ID is accessible using Nominatims(Figure 3.3), a tool for searching OSM data using location names [59]. With both sites, the polygons that define each neighborhood of Famalicão could be found by searching for it by name in Nominatims and then by using the ID found to download the polygon information as a GeoJSON, an encoding of JSON for geographical data.

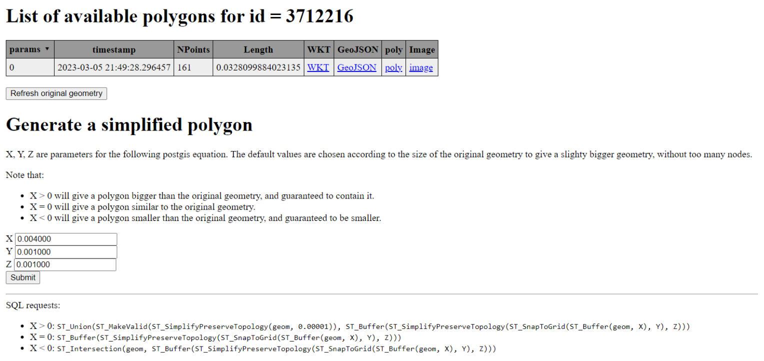
With both information about the values related to the location and the boundaries of the polygons, it was possible to write a Python code that computed the average value for each neighborhood. It loads the data that come in a JSON format and uses the shapely library to detect if a point is inside a polygon. Then it goes over all the polygons, and for each one, it goes over all the points



Bairro link to this page

Name	**Bairro** (name) **Bairro** (name:pt) **Freguesia de Bairro** (official_name)
Type	boundary:administrative
Last Updated	2022-07-31T17:41:15+00:00
Admin Level	8
Search Rank	16
Address Rank	18 (city / village district)
Importance	0.27733406670196176
Coverage	Polygon
Centre Point (lat,lon)	41.3743421,-8.427799
OSM	relation 3712216
Place Id	298194950 (on this server)
Computed Postcode	
Address Tags	
Extra Tags	**freguesia** (border_type) **8** (capital) **village** (linked_place) **3197** (population) **2021** (population:date) **031204** (ref:ine) **Q803962** (wikidata) **pt:Bairro (Vila Nova de Famalicão)** (wikipedia)

Figure 3.3: Nominatims polygon id collection



List of available polygons for id = 3712216

params ▾	timestamp	NPoints	Length	WKT	GeoJSON	poly	Image
0	2023-03-05 21:49:28.296457	161	0.0328099884023135	WKT	GeoJSON	poly	image

Refresh original geometry

Generate a simplified polygon

X, Y, Z are parameters for the following postgis equation. The default values are chosen according to the size of the original geometry to give a slighty bigger geometry, without too many nodes.

Note that:

- X > 0 will give a polygon bigger than the original geometry, and guaranteed to contain it.
- X = 0 will give a polygon similar to the original geometry.
- X < 0 will give a polygon smaller than the original geometry, and guaranteed to be smaller.

X 0.004000
Y 0.001000
Z 0.001000
Submit

SQL requests:

- X > 0: `ST_Union(ST_MakeValid(ST_SimplifyPreserveTopology(geom, 0.00001)), ST_Buffer(ST_SimplifyPreserveTopology(ST_SnapToGrid(ST_Buffer(geom, X), Y), Z)))`
- X = 0: `ST_Buffer(ST_SimplifyPreserveTopology(ST_SnapToGrid(ST_Buffer(geom, X), Y), Z)))`
- X < 0: `ST_Intersection(geom, ST_Buffer(ST_SimplifyPreserveTopology(ST_SnapToGrid(ST_Buffer(geom, X), Y), Z)))`

Figure 3.4: Open Street Maps polygon collection

in the given dataset and computes the average value of of the points that fall inside the polygon. Through this process, two new datasets of average elevation and leaf coverage for each neighborhood were created. The first rows of the created elevation database are shown in Table 3.5, and the leaf cover dataset follows the same pattern.

name	elevation
UNIÃO DAS FREGUESIAS DE ANTAS E ABADE DE VERMOIM	100.95
UNIÃO DAS FREGUESIAS DE ARNOSO (SANTA MARIA E SANTA EULÁLIA) E SEZURES	173.61
UNIÃO DAS FREGUESIAS DE AVIDOS E LAGOA	73.87
BAIRRO	108.56
BRUFE	174.39
UNIÃO DAS FREGUESIAS DE CARREIRA E BENTE	126.32
CASTELÕES	153.84
CRUZ	185.16

Table 3.5: First rows of elevation by neighborhood Dataset

Density Data

The crowd density dataset was also treated since the counts for the different moments in time for the same day were all in the same row, and that is not ideal for time series modeling. For modeling, the best format is one moment in time per row like in Table 3.6. Moreover, only the national counts are considered. So each row in the count database becomes three with the values from val_nac_00_06, val_nac_06_12, val_nac_12_14, val_nac_14_19 and val_nac_19_00. After the creation of the new rows, most columns are dropped, keeping only one date-time row, with the combination of the date and the middle time moment of two boundary moments; the sscode and the density for that moment.

sccode	Count	datetime
SC031224004	0.0	2021-11-08 03:00:00
SC031224004	30.0	2021-11-08 08:00:00
SC031224004	17.0	2021-11-08 13:00:00
SC031224004	26.0	2021-11-08 18:00:00
SC031224004	13.0	2021-11-08 23:00:00

Table 3.6: Example of count dataset after separating the time moments

The density data is then merged with the external features. First, the neighborhood is added based on the code to neighborhood mapping. Using the neighborhood the elevation and leaf coverage are associated to the rest of the data. Two new columns are created for the calendar information, one for the day of the week that contains a number from zero to six with zero being Sunday and six Saturday; and another for weekend or holiday which is binary. Finally the weather data is added based on the date and time, being the same for all zones.

This final version(Table 3.7) contains all the data in a way that is possible to model a time series for each zone. It can be fitted to a high-level model that accepts any sort of data, such as

sccode	Count	datetime	neighborhood	elevation	leafcover	day_of_week
SC031224004	0.0	2021-11-08 03:00:00	LOUSADO	59.312236	13.362963	0
SC031224003	74.0	2021-11-08 03:00:00	LOUSADO	59.312236	13.362963	0
SC031224001	174.0	2021-11-08 03:00:00	LOUSADO	59.312236	13.362963	0
SC031224002	282.0	2021-11-08 03:00:00	LOUSADO	59.312236	13.362963	0

is_weekend	temperature	relativehumidity	apparent_temperature	rain	cloudcover	windspeed
0	8.0	91.0	5.9	0.0	31.0	8.0
0	8.0	91.0	5.9	0.0	31.0	8.0
0	8.0	91.0	5.9	0.0	31.0	8.0
0	8.0	91.0	5.9	0.0	31.0	8.0

Table 3.7: Extract from final merged dataset

AutoML, but other models may need some modifications tough. The details of the modifications for it to work with each specific model are described in the model section. To facilitate the process of creating this high-level dataset from the other datasets, a Python program was created that is able to handle the geographical data, doing the average per region based on a map polygon and merging it with other data to generate the final dataset. This program can be used through a command line interface or as a pipeline for dataset treatment if imported as a library. The program is available on the GitHub page that accompanies this paper and where all the code used is available[60]

3.1.3 Data Analysis

Before working with the data, some analysis was done to be sure it was properly ready for being fed to the models. A Jupyter Notebook connected to a Python 3.10 runtime was used alongside the libraries Pandas for dealing with the datasets, Matplotlib for plotting graphs, and statsModels for doing decomposition.

First, the data were checked for missing values, for which none were found in the columns corresponding to the external values, but the crowd density data had some missing values. There are five days where there is no count of the density for any region and any moment in time. These are 2021-05-10, 2021-05-06, 2021-05-07, 2021-05-11, and 2022-03-09. This causes a small problem since three of the days are subsequent, and since each day has five time moments, it turns into a fifteen-entry hole. Many possible and common methods exist to solve this, such as removing the missing values, mean imputation(filling with the mean value), forward fill(filling the next data point value), backward fill(filling with the previous value), and interpolation.

For time series data, removing entries can interfere with the continuity of the series, so the ideal method is to give new values to the missing ones. In this case, an interpolation method was chosen to fill in the missing values using the pandas interpolate function. The pandas interpolate function calculates intermediate values for missing data points based on various interpolation techniques. It analyzes the existing data points around the missing values and generates estimates that fit well within the overall trend of the time series. Interpolation is a good idea for time series data because it helps to maintain the temporal order and captures the underlying patterns in the data.

By using the pandas interpolate function, the missing values in the crowd density data were replaced with estimated values that align with the surrounding observations. This ensures that the time series remains continuous and preserves the integrity of the data for further analysis or modeling purposes.

An STL decomposition was done to better understand the data and find patterns. The STL function requires a period to be set; three periods were analyzed to see the patterns, 5, 35, and 150, which correspond to a day, a week, and a month. It is clear to see from looking at the plots that the trends show an overall bigger number of people in most regions in the first months of the year, until the end of the summer, while the end of the year was the period with fewer people in most zones. Many zones have a very big difference in the number of people over the year, with a moment in time that it becomes much lower; however, this high difference could indicate some change in the measures and then some kind of data mistake. It is constant in all zones, with some keeping the values mean over the whole year and the moment of change varying between the others. The seasonal component indicates a very strong daily frequency that is present over all the regions and periods, with a peak in the number of people in the mid of the day and a lower amount at night. And the residual component does not show any pattern for all periods as expected. Examples of some zones decompositions and the three periods are present in Appendix section A.1

3.1.4 Clustering

The clustering was also done using a Jupyter Notebook with Python 3.10 runtime. Using the libraries Pandas for the data, Matplotlib for graphs, statsmodels for decompositions, and sklearn for machine learning tasks, specifically PCA, KMeans, and GridSearchCV.

Since a large number of statistical zones could affect the model's ability to capture the relations between them due to a big dimensionality, it was decided to experiment with clustering the zones to analyze the impact of modeling smaller regions. First, it experimented with clustering using the city's administrative boundaries so the zones in the same neighborhood were put together to create models. However, clustering based on the STL decomposition and using machine learning clustering methods was also used to see the impact on the resulting models.

The STL decomposition values were used as features for the decomposition. Each statistical zone became a row in a DataFrame, and the trend, seasonality, and residual decomposition values for the whole time period became columns. This way, the same moments in time for each zone would be in the same column and considered the same feature, no matter for which component of the decomposition. This resulted in a very large DataFrame of hundred and sixty-nine rows and 4531 columns. Hence the possibility of dimensionality reduction for applying the clustering algorithm.

To do the clustering, two common algorithms for this task were PCA for data transformation and KMeans for selecting the clusters. PCA stands for Principal Component Analysis, which is a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional space while preserving most of the information[61]. It achieves this by finding a new set of variables called principal components, which are linear combinations of the original variables. The principal components are ordered in such a way that the first component captures the

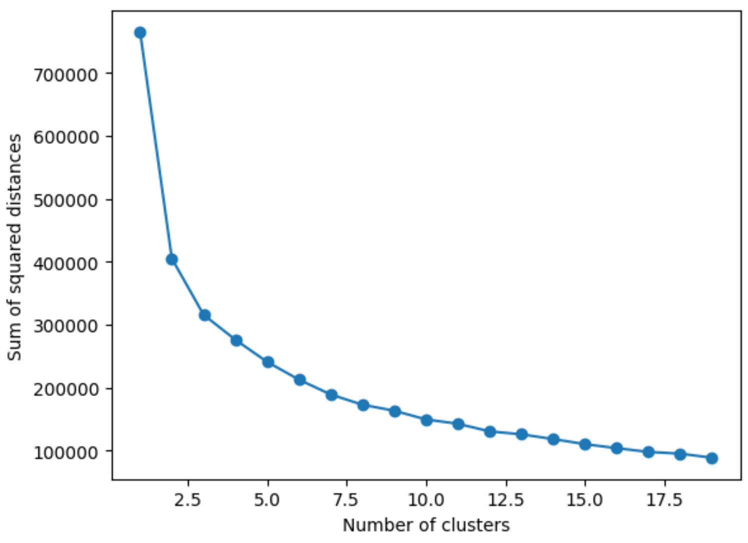


Figure 3.5: Elbow method plot for the number of clusters

maximum variance in the data, the second component captures the maximum remaining variance, and so on. PCA is commonly used in clustering because it helps reduce the data's dimensionality, remove noise, and identify the most relevant features for clustering.

KMeans is a popular clustering algorithm that aims to partition a dataset into a predefined number of clusters (K) based on their similarity. The algorithm works by iteratively assigning data points to the nearest centroid (cluster center) and updating the centroids based on the newly assigned points. The process continues until convergence, where the centroids stabilize, and no further changes occur[62]. KMeans clustering is an unsupervised learning technique and requires the number of clusters to be specified in advance.

Two complementary methods were used to determine the appropriate number of clusters in KMeans. First, the elbow method is employed. The elbow method involves plotting the number of clusters against the corresponding sum of squared distances between data points and their cluster centroids. The plot forms an "elbow-like" shape, and the idea is to identify the number of clusters at the "elbow" point where the incremental gain in clustering performance diminishes significantly. This point represents a trade-off between reducing intra-cluster variance (within-cluster distance) and increasing inter-cluster variance (between-cluster distance)[63].

Looking at the plot of KMeans for one to twenty clusters(Figure 3.5), the elbow is located at the second point. This means that adding more clusters beyond that point doesn't provide significant improvement in clustering performance. A grid search with cross-validation using the SkLearn GridSearchCV method is performed to confirm this observation. This technique systematically evaluates different combinations of the number of clusters and components for the PCA transform. The best number of clusters and components are selected based on the performance metrics obtained from cross-validation.

The resulting number of components for PCA was ten. Finally, with these two parameters selected, the two clusters are computed, resulting in a cluster with 51 zones and another with

118 zones. These two clusters are used alongside the clustering by neighborhood to compare the models. A list of the zones selected by each cluster can be found in Appendix section A.4

3.2 Proposed Solution

In order to find the most precise model for predicting the given data for crowd density of the city of Famalicão, four machine learning models are trained and compared. These are AutoRegressive models, referred to as Time Series models; LSTM Neural Networks; Graph Neural Networks; and AutoML. Each of the models was trained in four different scenarios for means of comparison: creating one model for each statistical zone, creating a single model for all zones, creating one model per neighborhood, and creating one model per cluster. It should be noted that for the GNN, there is no model for individual zones since it needs the graph information as part of its architecture, and a single point is not enough to be a graph in this context. For each of the trained models, the RMSE, MAE, and MAPE errors were collected to be analyzed after.

Additionally, AutoML is used to empirically test the effect of the external features on the resulting model by creating many models using different combinations of features in order to understand their effect on the final model. This approach is particularly chosen due to time constraints and the efficiency of AutoML in model training compared to other models that might have longer training times. AutoML internally tests various models and employs ensemble techniques to combine their predictions. This ensemble approach can be effective in detecting the best model for each input given, as it leverages the diversity of models and their respective strengths. By automatically evaluating and selecting the most suitable models from the ensemble, AutoML increases the chances of finding an optimal model configuration for the given task.

Each of the four model types was developed in a different Jupyter Notebook with Python 3.10 runtime and using many different libraries explained individually in each model section.

3.2.1 Time Series

AutoRegressive models are very traditional and widely used for Time Series data. ARIMA is the most commonly used for simple uni-dimensional sequential data, while VARMA and VARMAX are vectorial alternatives that can deal with higher dimensional data. The Time Series models are developed as traditional statistical models to be compared to more modern data-centric models in the crowd density forecasting task as they were present in a large portion of the reviewed literature and have shown good results in the past.

3.2.1.1 Methodology

The models were developed in an incremental process to prepare the data properly. First, a simple ARIMA model is created for a single zone. In order to do so, a zone is selected at random. The Python library pmdarima offers the auto_arima function that is able to model an ARIMA model from a time series finding the parameters automatically. The data needs to be adjusted a bit in

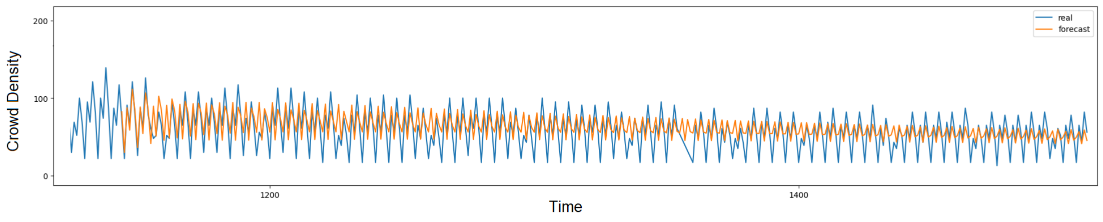


Figure 3.6: Actual and forecast data of autoARIMA

the original dataset; each row is a unique combination of DateTime and zone, but for the time series models, the rows cannot repeat the time, so each row represents a moment in time that can be put in order without repetition. So the counts for different zones become columns; this way, the time is unique, and the counts are present in the respective column. At first, only the density for the selected zone, without the other features, is used as input. Since the function only accepts uni-dimensional data, the simple time series sequence of values is like an array of numbers.

The automatic model selects an ARIMA(4,1,1) with a p-value of 4, indicating the number of lagged observations used for autoregression. The d-value of 1 denotes the degree of differencing applied to make the time series stationary, and the q-value of 1 signifies the number of lagged forecast errors considered for the moving average component. Since the next models are not integrated, and the differentiation needs to be done before, the value of 1 resulting from the automatic model will be used. The process of training the model with the amount of data given is fast, around 40 seconds to train. A list of the errors and parameters for all the zones is presented in the annexes.

The predictions on all of the Autoregressive models have a tendency to lose accuracy the further it gets from the training data, as can be seen in Figure 3.6. This phenomenon, known as forecasting horizon effect or forecasting horizon bias, is a characteristic of ARIMA models. As the distance between the observed data and the predicted values increases, the influence of unforeseen factors and external influences grows, leading to a gradual decrease in prediction accuracy. Therefore, it is crucial to consider this limitation and interpret the model's predictions with caution when extrapolating beyond the available training data.

Following the progressive development, a model is created using VARMA, a vector Autoregressive model, which means that instead of a single dimension, it models many variables simultaneously as a vector. This input is a matrix where each row is a subsequent moment in time, and each column is a zone. The disadvantage of this model to the autoArima is that it is not automatic, so the parameters must be given, and it does not differentiate the data. Differencing is a technique used to transform a non-stationary time series into a stationary one. It involves computing the differences between consecutive observations in the series. And since the data is non-stationary, it should be differentiated, which can be easily done by using pandas.

The importance of differentiation lies in the fact that autoregressive models are specifically designed to handle stationary time series. Stationarity simplifies the modeling process by making the statistical properties of the data constant over time. By achieving stationarity through differencing, we enable the autoregressive model to accurately capture the inherent patterns, trends, and dependencies within the data. Without proper differentiation, the model may produce unreliable results, as it would struggle to account for the changing statistical properties and dependencies present in non-stationary time series.

After the differentiation, the same parameters as the ones found in ARIMA are used, creating a model VARMA(4,1). VARMA takes much longer to train than ARIMA since it has more variables to consider and find relations. Nevertheless, the high dimensionality of the data caused models with many zones at the same time to not converge. So it was not possible to create some of the wanted models for Time Series methods, namely it was not possible to create a single model for all zones, nor the ones for the two clusters obtained by clustering algorithms. This means that the time series models for multiple zones were done through neighborhood clustering, although there is a single neighborhood from which the model could not be created. The neighborhood of União das Freguesias de Vila Nova de Famalicão e Calendário that has 28 zones and is the largest neighborhood in the city.

Until now, all the models used only the density for input. VARMAX is a model that can handle both vector data and exogenous variables as input(Table 3.8). Since the AutoRegressive models are statistical and model the data as purely time series, they are not able to extract the influence of features that are constant during the whole time series, that is, the elevation and leaf coverage. indeed, the model does not even accept its input, causing an error. So these features are left out of the exogenous variables in the train data. Nevertheless, with VARMAX, it is possible to model all the zones, with almost all the external features as wanted, that being the final Time Series model to be compared with the others. The same convergence problem from VARMA is also expressed in VARMAX, causing the models for all zones, for the algorithmic clustering, and for the neighborhood with 28 zones not to be created.

datetime	day_of_week	is_weekend	temperature	relativehumidity	apparent_temperature	rain	cloudcover	windspeed	SC031201001_Cover	...	SC031248006_Count	SC031248007_Count
2021-05-03 03:00:00	0	0	6.3	90.0	4.0	0.0	0.0	7.5	4.4	...	9.0	126.0
2021-05-03 08:00:00	0	0	12.4	71.0	11.7	0.0	0.0	7.2	4.4	...	48.0	837.0
2021-05-03 13:00:00	0	0	19.0	46.0	21.1	0.0	0.0	6.4	4.4	...	35.0	603.0
2021-05-03 18:00:00	0	0	16.4	56.0	14.7	0.0	0.0	14.5	4.4	...	61.0	1006.0
2021-05-03 23:00:00	0	0	9.8	88.0	7.7	0.0	2.0	7.6	4.4	...	35.0	607.0

Table 3.8: Extract from VARMAX input, with endogenous and exogenous variables in the same table

3.2.1.2 Challenges and Advantages

The main advantages of using AutoRegressive models, such as ARIMA, VARMA, and VARMAX, in the context of Time Series analysis, are well-documented and widely recognized within the field. These models offer several benefits, making them popular choices for analyzing and forecasting temporal data in various domains. One of the primary advantages is the relative ease of implementation. AutoRegressive models follow a straightforward mathematical formulation that captures the temporal dependencies within the data. Furthermore, AutoRegressive models have been extensively tested and validated over time, establishing their reliability and effectiveness in capturing key characteristics of Time Series data.

However, it is important to acknowledge that AutoRegressive models may encounter challenges when dealing with high-dimensional datasets. As the dimensionality of the data increases, the complexity of modeling and computation escalates, potentially impacting the convergence of the models. The training time required can be a limiting factor, especially when dealing with extensive datasets. The computational complexity associated with these models can lead to lengthy training processes.

3.2.2 LSTM Neural Networks

Neural Networks have been showing growing success in many fields, and LSTMs, especially for timed data, due to their recurent nature, making them more capable of modeling the inherent dynamics in crowd density data. Secondly, the memory cells of LSTMs enable them to effectively capture both short-term and long-term dependencies, enhancing their ability to capture the evolving patterns in crowd behavior over time. LSTM is used in [34] and [32].

3.2.2.1 Methodology

Neural Networks are implemented using Keras, a user-friendly interface and a simplified abstraction layer that acts as an interface to backend libraries, including TensorFlow, which is one of the most popular and widely used open-source frameworks for deep learning. TensorFlow is an end-to-end open-source platform for machine learning. It offers a comprehensive ecosystem of tools, libraries, and resources that enable the efficient development and deployment of neural network models. It provides a scalable and flexible framework for implementing a wide range of machine learning algorithms, including neural networks.

Neural Networks have the advantage of being data focused, and so need very minor data preparation before being fitted. Mainly the data needs to be Scaled and encoded, which SKlearn provides functions to do. A MinMaxScaler is used for all the numerical values: density, weather features, elevation, and leaf coverage. This scaler will transform the values into a number between zero and one relative to the minimum and maximum values present. This transformation is done before separating the data into zones, even for the cases where one model for each zone is created so that they keep consistency when evaluated by each other. Then a Label Encoder that transforms categorical labels into numerical representations is used for the non-numeric values, or the numbers represent something else: the zone code and the neighborhood. In that case, the zone and the neighborhood become a number from zero to the total number of zones or neighborhoods. The day of the week and weekend features did not need any work since the day was already encoded equivalently to the label encoder, and the weekend is a binary feature.

Unlike the time series methods, the neural networks do not model a function directly, and the training values are not interpreted sequentially. Hence, the approach to dealing with forecasting consists of giving as input the previous densities alongside the differentiating features that may give hints on how to deal with that different zones. The size of the look-back window is how many moments in time will be given as input. It is important to consider when choosing this value

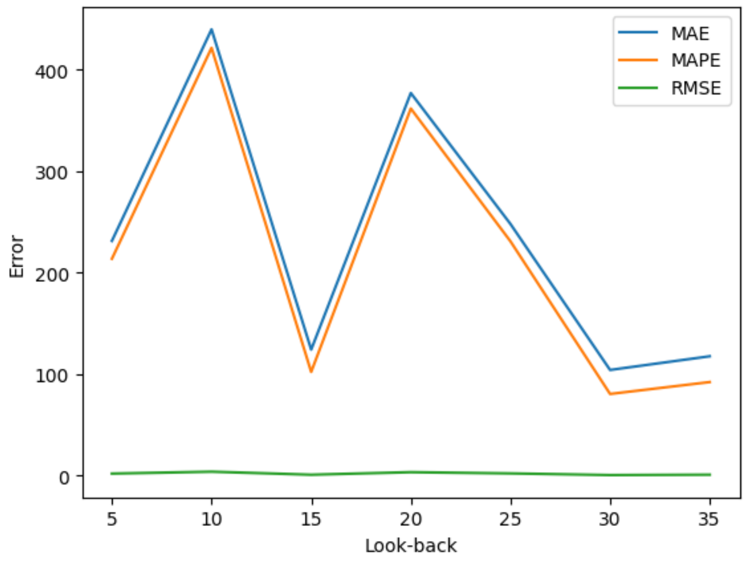


Figure 3.7: Error by look-back comparison

that each time moment is a fifth of a day, so having too few moments in time may lack on giving enough information for the model. However, too many may make it too high dimensionally and less precise.

The implementation of the Neural Network was also done incrementally. First, a model with just the density was created for a single zone, which helped to analyze and pick a good number for the look-back window size. Which was set to ten, meaning the two previous days. The model is a sequential model, which means that the layers are stacked sequentially on top of each other. The architecture consists of two main layers: an LSTM layer and a dense layer. This model was used in the incremental process for getting used to the process of creating the NN.

An Analysis was done to determine the best parameters. First, the look-back needed to be set; for this, a model was trained with the same parameters and just changed the look-back size. And the errors were analyzed with the graph Figure 3.7. Following, a cross-validation parameter tuning process was done, with different hyperparameters being tested for the number of LSTM units, batch size, activation function, the number of epochs, and optimizer. The cross-validation step was only done after setting the size of the look-back window.

The LSTM layer is the primary component responsible for capturing the temporal dependencies in the input data. In the model selected by the parameter tuning, the LSTM layer is initialized with 8 units, indicating the number of memory cells or hidden units within the layer. The look back was set to 15, although 30 had a similar error rate to keep the dimensionality low.

Following the LSTM layer, a dense layer is added to the model. The dense layer, also known as a fully connected layer, consists of one or more neurons that connect every input from the previous layer to the output. In this case, a single neuron is used, indicated by the parameter value of 1. The purpose of this dense layer is to transform the features learned by the LSTM layer into a single output value, which is often the predicted value or a relevant representation of the desired output.

The model is compiled using the MSE as the loss function, quantifying the difference between the predicted and true outputs. The optimization algorithm employed is Adam, which is a popular and efficient optimizer commonly used in neural networks. The compilation step finalizes the model configuration, preparing it for training and evaluation. The training is finally done with a hundred epochs. Once the look-back and the hyper-parameters were set, the model was tested with all the features(Table 3.9), first for a single zone and then for all zones and the clusters. This incremental process and testing parameters with simpler inputs saved time since the neural networks were by far the models that took longer to train.

Count^{-10}	Count^{-9}	Count^{-8}	Count^{-7}	...	elevation	leafcover	day_of_week	is_weekend	temperature	relativehumidity	apparent_temperature	rain
0.009947	0.069940	0.036331	0.083133	...	0.271972	0.285523	2	0	0.258462	0.958904	0.316953	0.000000
0.069940	0.036331	0.083133	0.049524	...	0.271972	0.285523	2	0	0.418462	0.684932	0.466830	0.000000
0.036331	0.083133	0.049524	0.009528	...	0.271972	0.285523	2	0	0.587692	0.438356	0.616708	0.000000
0.083133	0.049524	0.009528	0.071720	...	0.271972	0.285523	2	0	0.510769	0.643836	0.525799	0.000000
0.049524	0.009528	0.071720	0.039472	...	0.271972	0.285523	2	0	0.356923	0.945205	0.422604	0.000000

Table 3.9: LSTM NN input

3.2.2.2 Challenges and Advantages

The best advantage is that it does not need much preparation of the data to fit and that it can easily find relations without much effort. LSTM networks excel at capturing long-term dependencies and discovering intricate relationships within sequential data with relative ease. Their inherent memory cells and recurrent connections enable them to retain and update information over long sequences, making them capable of capturing complex patterns and dependencies that span across various time steps. However, it is important to acknowledge that not effectively managing the data can lead to challenges, such as the curse of dimensionality. The curse of dimensionality refers to the adverse effects of having excessive features or dimensions in the input data. When dealing with high-dimensional data, LSTM models may encounter difficulties learning meaningful representations and suffer from overfitting or reduced generalization performance.

On the other hand, a significant disadvantage of LSTM models lies in their time-consuming training process and the complexity associated with fine-tuning. Training LSTM networks can be computationally intensive, especially when dealing with large-scale datasets or complex architectures. The sequential nature and recurrent connections of LSTMs introduce additional computational overhead, as they require iterative computations over multiple time steps. Consequently, the time required to train LSTM models can be significantly longer compared to other machine learning algorithms, posing challenges in terms of computational resources and time constraints. Moreover, fine-tuning LSTM models to achieve optimal performance can be complex. Moreover, fine-tuning LSTM models to achieve optimal performance can be complex.

3.2.3 Graph Neural Networks

Graph Neural Networks are an easy choice for comparing models due to their rise in popularity and examples of success in forecasting spatio-temporal that comes from their unique ability to include graph information into their training process. Which has been shown to be an affective way of including the spatial dependencies into temporal data as shown in [25], [24], [39], [26], [42] and [43]

3.2.3.1 Methodology

The graph neural network was the model that needed more work. First, because none of the most consolidated libraries for neural networks offer functions to build GNNs directly, so with the use of Tensorflow, some classes were implemented following keras documentation[64] for a graph convolutional layer, graph information, and a graph LSTM class that integrated the graph convolutional layer with an LSTM layer and a dense layer that will form the neural network.

The GraphConv class represents a graph convolutional layer in the neural network. It takes a set of features as input and performs graph convolution operations to compute the representations of nodes in a graph. It computes the node's representations during the forward pass by multiplying the input features with a weight matrix. It then aggregates the representations of neighboring nodes. The aggregated messages are combined with the node's representations, and an activation function is applied to produce the final node representations.

One of the implemented classes, LSTMGC, represents a layer that combines graph convolution, LSTM, and dense layers, allowing for the integration of graph-structured data into sequential models. This class specifically captures temporal dependencies and patterns within graph-structured sequences. The layer takes input sequences of features and performs graph convolution operations using the GraphConv class, which leverages the provided graph information to aggregate and update node representations. The graph convolution step is followed by an LSTM layer, which processes the graph-convolved representations over the sequential dimension, capturing long-term dependencies and encoding temporal information.

The input for the Graph Neural Network was the most different one compared to the other models. First, the information on the connectivity was necessary in order to build the graph. For that, the origin-destination dataset was used, iterated through, and every zone with at least one movement from or to was considered connected. With that information, an adjacency matrix was created where the lines and columns were the zones, and the intersections of zones were set to zero or one if there was a connection. It is important to note that the destination dataset only contained data from a single day, so it was possible that not all the real connections were represented if two connected zones did not have any movement on that day or due to some problem with the data. This was the case since, when plotting the graph, it is possible to see that not all nodes are connected. Additionally, it should be noted that the dataset contained zones outside of the city, with the flow of people in and out of it. Still, since the density dataset was only for the zones inside the city, these nodes were removed from the graph. With the adjacency matrix, it was possible to construct a non-directed graph with the connections Figure 3.8.

Once the graph information was ready, the density count needed to be transformed into a matrix too, similar to the one used in the VARMA for Time Series, but transposed. Each row was a zone, and the columns were the subsequent moments in time. The data was normalized, and a

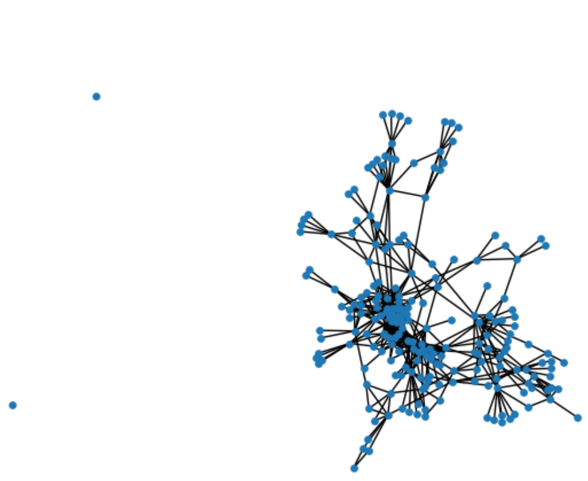


Figure 3.8: Zone adjacency graph

look-back of fifteen moments was used. Due to time restraints, the additional features were not used in the GNNs, only the density data. However, the graph information could be considered an additional feature that is not present in the other models. Other than the graph data, the data should work the same as the LSTM NN with a look-back window as input and a density as output.

A GNN model was trained for all the zones, the two clusters, and the neighborhood clustering. A single model for each zone would not make sense in the GNN since there would be no information to be extracted from a graph that is only one node, and it should be the same as the LSTM NN for a single zone. For six of the neighborhood clusters, it was not possible to create a NN since, due to the missing data in the graph, the zones inside the cluster were not connected. All the skipped neighborhoods had only two zones inside.

3.2.3.2 Challenges and Advantages

One of the primary advantages of graph neural networks is their efficient training time, which plays a crucial role in achieving desirable results, as discussed in detail in chapter 3. Another advantage is the unique architecture and ability to leverage the inherent graph structure of the data. GNN can effectively capture and exploit spatial dependencies between nodes by utilizing a graph representation, enabling them to extract meaningful relationships that other models may struggle to uncover.

However, one notable disadvantage is the scarcity of comprehensive libraries specifically tailored for graph neural network development. Unlike traditional neural networks, which benefit from an extensive range of well-established libraries and frameworks, the graph neural network domain still lacks the same level of tooling support. Another critical consideration when working with graph neural networks is the essential requirement of extra graph information. Therefore, it is impossible to leverage the full potential of graph neural networks without incorporating supplementary graph information. And acquiring and preprocessing this graph information can be a complex and time-consuming task, posing a challenge to the graph neural network's practical adoption and scalability.

3.2.4 AutoML

AutoML is a good approach, especially for comparison reasons, since it uses the power of an ensemble to find the best model inside its limitations. And offers a good way of comparing the external features of the data and their influence on the resulting model. It is not a very explored method in other works, but it has been used in some projects, especially for works using the same dataset before.

3.2.4.1 Methodology

There are many libraries for automatic machine learning, and the one used was from SKlearn: auto-sklearn offers two essential methods: AutoSklearnRegressor and AutoSklearnClassifier, for regression and classification, respectively. These methods automate the entire machine-learning pipeline by leveraging advanced techniques such as algorithm selection, hyperparameter optimization, and model ensemble.

The trained LSTM NN data already had all the transformations needed, such as the encoding and scaling, making it unnecessary to perform additional data processing. The same look-back window and feature encoding techniques employed during the LSTM NN training phase can be directly applied to the input data for the AutoML Regressor. So the data was easily fed, and the models could be created for the scenarios of a single model per zone, all zones, the cluster, and the neighborhoods.

The external features were also tested against each other to measure their impact on the resulting model. For that, the most differentiating features were selected, that being: 'day_of_week,' 'is_weekend,' 'temperature,' 'rain,' and 'windspeed, the features like apparent_temperature were kept out to reduce the number of combinations since they are related to other like temperature. Then a model for a single zone was created with each of the combinations of these features, resulting in 31 models. Additionally, a model for all zones was created using the combinations of elevation, leaf coverage, both features, and none of the two. This way, the data was collected to try to understand the impact of the features in the models.

3.2.4.2 Challenges and Advantages

The main advantage of AutoML is the ease of implementation, which makes it fast to implement without much preparation, in this case, where there were other models with minimal data preparation since the one from the other models could be reused. The training time is another advantage. The AutoML was very fast to train and gave the ability to control the maximum time per model, limiting the train time when training many models.

One of the main disadvantages is the potential lack of transparency and interpretability in the generated models. Due to the automated nature of the process, AutoML may involve complex model architectures and combinations of algorithms, making it challenging to interpret the underlying reasoning behind the model's predictions. Additionally, the automated nature of AutoML may limit the control over the fine-tuning of models, potentially resulting in suboptimal performance or models that do not align with specific domain requirements.

Chapter 4

Experiments and Results Analysis

This chapter presents the results obtained by the four kinds of models implemented, compares them, explains the differences in the obtained forecasts of each one, and presents a discussion about the best model obtained. Additionally, it compares the impact of each feature in the obtained results as well as the time to train each model and the impact of this time. Finally, the application of the results in the real world is presented and discussed.

4.1 Results

Four different kinds of models were trained: Time Series, LSTM, GNN, and AutoML, with four variations of the data each: one model per zone, one model for all zones, one model per cluster, and one model per neighborhood. From these models are extracted the error metrics MAE, RMSE, and MAPE, which are compared. AutoML was used to create different models for the different features too, from which the same metrics were extracted. In this section, the models are compared based on their errors and time to train.

Overall the best resulting models were the Graph Neural Networks which presented significantly lower errors than the other models for all the scenarios. The best performance for the GNN was for one model per neighborhood. However, the models did their best when creating a single model for all the zones at the same time. The AutoML models were the second best, with the LSTM NN not far behind, presenting very similar metrics. Although for the one model per zone, the Time Series exceeded the LSTM. The forecasts of the best GNN model for a single model against the actual data are present in Figure A.7, the bottom one being the best-predicted zone and the upper the worse, showing that both were very precise forecasts, their errors are present in Table 4.1. And a table with all zones error for this model can be found in Appendix section A.3.

Zone	MAE	MAPE	RMSE
SC031223001	1.28	2.36	1.35
SC031225002	0.22	0.24	0.24

Table 4.1: Geographical Features

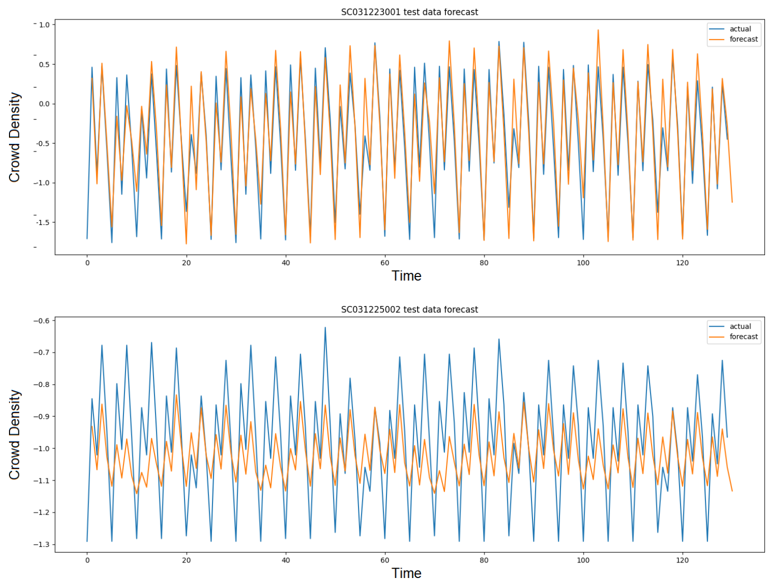


Figure 4.1: GNN single model forecast

4.1.1 Errors Comparison

When comparing the models between themselves, there is no single data division method that worked the best for all of them. Some of the model types expressed had their best results for single zone models, others for clusters or a single model for all zones. The standard deviation between the models also varies a lot, showing the difference in modeling different zones and clusters.

Each of the four scenarios tested is compared using the errors, and for each, there is a table. Table 4.2 shows the errors for the three kinds of models that are able to create models for this scenario and show the best-performing model to be AutoML. For one single model for all zones, the best model is GNN, as can be seen in Table 4.3. The neighborhood clustering has also GNN as the best, in Figure 4.2. Finally, GNN also performed best in the algorithmic clustering, shown in Table 4.5.

Three error metrics are used for comparison, RMSE, MAE, and MAPE. These were the most present in the revised articles. And are useful because And are useful because they provide a comprehensive evaluation of the accuracy and performance of the models. RMSE provides a measure of the overall prediction error magnitude. MAE offers insight into the average magnitude of the errors. Finally, MAPE enables a relative assessment of the forecast accuracy.

The Time Series models had their best performance when a single model was created for each zone, which was expected since the autoRegressive models are indeed not optimized for very large dimensionalities, even the vectorial methods. There was a large standard deviation between the errors tough, and the best model was for the zone SC031247003 with an MAE of 8.06, with many zones with MAEs less than fifty, and many zones with errors greater than two hundred, the worst one being SC031246002 with 611.37.

The LSTM NN, on the other hand, had more similar results between the different partitions. It shows its best overall results when creating a single model for all zones, with much lower errors

Error	Time Series	LSTM	AutoML
RMSE_{Avg}	141.86	164.39	74.07
RMSE_{Std}	132.10	125.50	72.14
MAE_{Avg}	112.90	134.94	57.48
MAE_{Std}	107.11	100.38	59.57
MAPE_{Avg}	3.91×10^{14}	5.99×10^{14}	2.57×10^{14}
MAPE_{Std}	2.53×10^{15}	3.80×10^{15}	1.28×10^{15}

Table 4.2: One Model per zone

than when compared to one model for each zone and the neighborhood clustering. Nevertheless, the errors for the second algorithmically determined cluster were actually lower than the one for all zones at the same time. Still, the first cluster was much worse, with cluster 1 having an MAE error of 184, while cluster 2 had 39, and the single model had 48.68. The deviation in the single models for each zone was similar to the time series, the SC031247003, which was the best zone for Time Series, was still between the best but the fourth lowest error, with the first place going to SC031219010 with an MAE of 17.45. Indicating that some zones were easier to model even across the different techniques and showing that the best zones for the Time Series actually performed better than the LSTM.

Error	LSTM	GNN	AutoML
RMSE	86.44	0.66	67.83
MAE	48.68	0.51	31.90
MAPE	$21{,}20 \times 10^{13}$	1.89	$22{,}42 \times 10^{13}$

Table 4.3: One Model for All Zones

The GNN showed its peak performance on the clustering by neighborhood, although its errors were not that lower between the different partitions of the data. However, this could be due to the lack of six neighborhoods from which there was not enough information for the creation of models. The best neighborhood was Nine, with an MAE of 0.12, which was not the case for the other models. For both LSTM and AutoML, Nine was between the lower errors but was not particularly easier to model, although the overall errors of LSTM were bigger than the AutoML. The GNN was the model that showed the least difference between the two clusters, although it kept the pattern of showing cluster 2 as easier to model. Since the difference between the partitions was very small, in the selection of the best model overall, the one for all zones would be preferred since it does not lose the information lost in the neighborhood and still has a very good performance with an MAE of 0.66, which was much lower than any other of the other kinds of models.

Figure 4.2 shows a map with the neighborhoods of the city colored by the average error of the forecast of the zones in the neighborhood. From the best performing GNN model for all zones at the same time.

AutoML got the smallest errors when creating a single model for all the zones but with very similar values for the models for the neighborhoods. The one model for each zone models had

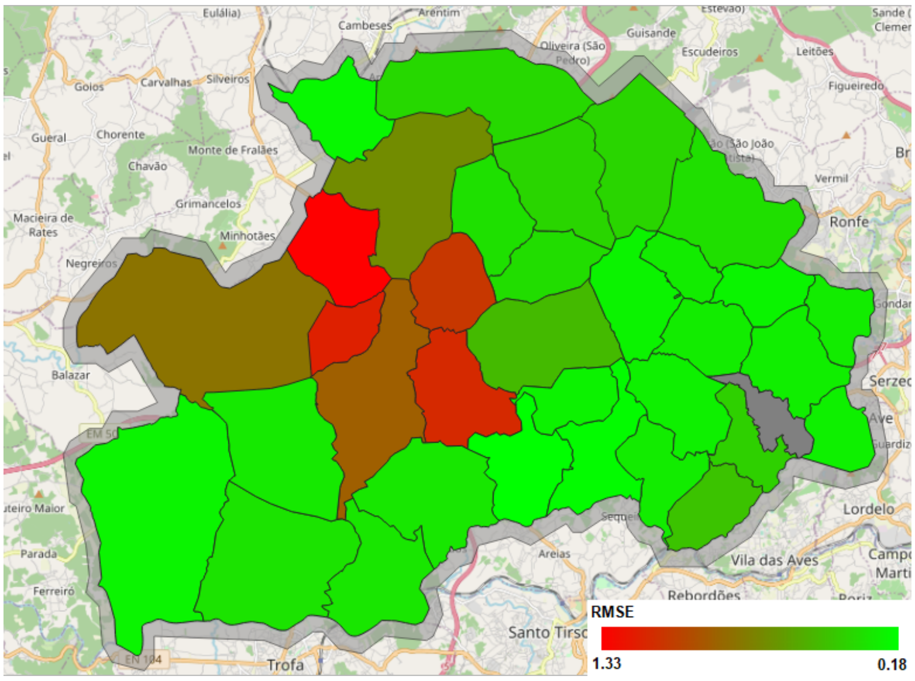


Figure 4.2: GNN single model mean forecast error by neighborhood map

the best performance out of the three types of models that had this partition, achieving an MAE of 4.23 for the zone SC031247003, and the zones SC031219010(the best for LSTM) got 5.68, and zone SC031247003(the best for Time Series) got 4.23, maintaining the trend of easier to train zones. Although the standard deviation was also big, as in the other models, AutoML managed to create many models with very low errors, much lower than the 31.9 MAE for the one model for all zones.

When considering the use of a final model, having a single model for all zones is preferable since it centralizes the task. And the GNN got the smallest errors in all tasks, achieving a very satisfactory result for using the single model for all zones. Nevertheless, the hardest of the problem proved to be, as expected, managing the different zone's particularities into a single model. This is made clear by the fact that many of the individual models for LSTM, AutoML, and Time Series got results near the GNN for individual zones, but none managed to reach it in a single unified model. Showing the capabilities of the Graph structures in the neural network model to capture the spatial relations in a superior way to the other models that only had the features as a resource to extract the spatial relations.

There are 41 LSTM models created for each zone and an average MAE of 38.89, 6 neighborhood ones with an average MAE of 36.84, that got lower errors than the single model for all zones. Which represent around 24% of the zones and 18% of the neighborhoods. AutoML got 73(43%) of the models for each zone with an average MAE of 15.97 and 17(50%) for the neighborhood and an average MAE of 17.87. Finally, the Time Series models did not have a unified model for comparison, but using the AutoML unified model error as a threshold, it has 35(20%) zone models with lower error and an average MAE of 21.05. This, together with the high standard deviation

Error	Time Series	LSTM	GNN	AutoML
RMSE_{Avg}	2124.51	160	0.53	68.32
RMSE_{Std}	1746.32	87.13	0.35	46.57
MAE_{Avg}	1649.25	126	0.45	42.44
MAE_{Std}	1369.15	72.30	0.32	32.28
MAPE_{Avg}	7.41 x 10^{14}	2.11 x 10^{15}	1.43	4.60 x 10^{14}
MAPE_{Std}	1.41 x 10^{15}	8.97 x 10^{15}	2.76	1.45 x 10^{15}

Table 4.4: Clustering by Neighborhood

values, shows that there was a big difference in the ease of modeling the different zones, with some zones being significantly easier and others being much harder. And add to the idea that the models can indeed generate very low error forecasts closer to the GNN, but they fail to generalize it into a single model.

4.1.2 Time to Compute

Another big concern for choosing a model other than their error performance is the time it takes to prepare the data, train and do the forecasts. While running the experiments, the best performance time-wise was for the GNN, which was able to train at very fast times alongside achieving low errors, and with a similar time to train for all sizes of datasets, it was indeed faster for smaller datasets as expected, but not in an order that made much difference. The GNN usually took minutes to train, and the largest model took a bit less than one hour. On the other hand, LSTM NN had the worst times on the models, not considering the Time Series models which did not converge. The largest model took two days to complete its training as seen in Table 4.6.

The AutoRegressive are very fast models to train with simple time-series data, taking usually less than a minute to train for single zones, and forecasting only takes a few seconds. Adding the exogenous variables itself does not change much the training time. What really causes it to be much slower is adding more zones. The models with the largest amount of zones, the biggest was 10, would take around one hour to train. The forecasting is done fast, tough, taking less than a minute. The data preprocessing time is fast and does not add much overhead. The most notable observation is that the model should be retrained every time there is new data to keep its performance since it loses precision the further it gets from the original data, which is not the case for other models, and this adds an additional step for using the model.

The LSTM NN are the slowest to train overall but varies a lot in time between different datasets. Compared to the other models, the overall time to train was the most directly related to the size of the dataset. For example, the time to train all the single zone models, when added, was not much slower than the time to train the single model for all zones. The individual models took some minutes to train alone, and the model for all zones took a bit less than two full days. A notable consideration for LSTM NN is the parameter tunning which also takes a long time, some few hours. The forecasting process is fast, and data processing is straightforward and repeatable if the model is to be retrained, though.

Error	LSTM	GNN	AutoML
$RMSE_{C1}$	184	0.82	143.44
$RMSE_{C2}$	39	0.58	292.70
MAE_{C1}	128	0.62	84.34
MAE_{C2}	25	0.43	270.97
$MAPE_{C1}$	2.88 x 10^{14}	2.65	2.29 x 10^{14}
$MAPE_{C2}$	2,90 x 10^{14}	0.97	8.10 x 10^{15}

Table 4.5: Clustering algorithmically (C1 = cluster 1 and C2 = cluster 2)

The GNN was the fastest overall, with similar times no matter the size of the dataset. The smaller datasets, the ones for the neighborhoods with few nodes in the graph, would train in a few minutes, while the one for all zones took a bit less than an hour. The forecasting time was the biggest among all models, but never more than three minutes. The data processing time was also the largest among all models. The process of extracting the graph data is a bit expensive but does not take much time either.

Analyzing the time for training with AutoML is trickier since the library gives the programmer the liberty of setting the maximum time to compute each model(since it builds many for the ensemble) and a maximum time for the whole training process. Nevertheless, if the maximum time is too low, it may not find models to make an ensemble from. A large time was set for the large models, so they could take as much time as needed, and it took a few hours to find a model. While for the smaller datasets, giving it too much time would make the whole process too slow since the time AutoML took to train was not directly related to the amount of data in the experiments done. So a time limit was put for the one model per zone, and when training with the different parameter combinations and even with that, it took some hours. The forecasting is very fast, always some seconds. And the data processing is the same as the LSTM, very straightforward and reproducible.

4.1.3 Features

AutoML was used to compare the performance of the models when considering different features in the train dataset. This was only done in AutoML instead of with all model types due to time constraints since this would add thirty-four more options for every model it was tested. There were five non-geographical features selected to test their combinations against each other using single-zone models. They were tested in a single-zone model since they are only time related; they are the same for all zones, so the error values are comparative, not general. They should be used as a reference to what extra features give better information to be added to the time-series data. Although ideally, this should also be done for many zones, since the influence could differ, it works as a reference. Additionally, the two geographical features of leaf coverage and elevation were compared using a multi-zone model since they vary geographically and should work as complementary information for the models to differentiate the zones.

Of all the thirty-one features tested, the best combination was having only the day of the week information with an MAE of 27.44. However, the standard deviation of the features was not that

Model	Time
Time Series$_{min}$	Seconds
Time Series$_{max}$	Hours
LSTM NN$_{min}$	Minutes
LSTM NN$_{max}$	Days
GNN$_{min/max}$	Minutes
AutoML$_{min/max}$	Minutes/Hours*

Table 4.6: Time to train

*AutoML enables the programmer to set the maximum time to train.

Features	RMSE	MAE	MAPE
Leaf cover	59.26	27.71	2.94 x 10^{14}
Elevation	59.27	27.59	3.03 x 10^{14}
None	59.09	27.44	3.07 x 10^{14}

Table 4.7: Geographical Features

big and taking into consideration possible variations that could appear when compared with other zones. There are some other combinations of features that achieved similar error rates to the day of the week only, explicitly the ones in Table 4.8: the day of the week plus the weekend indicator; the weekend indicator, and the rain; and the three features together, day of the week, weekend, and rain. This complies with the common sense that these should be some of the features that affect the movement patterns the most.

Some plots between the selected features and the density to see if the relation was visible. When doing a scatter plot of the rain against the density it is possible to see that overall the density does go down when there is more rain, although this is not always noticeable when looking at directly at single zone plots, and there are much more days with lower amounts of rain that could influence the visualization. The day of the week and is weekend are less direct to see in the plot, however when looking in a zone it is possible to see some zones in which some days have significantly less density the other, and overall the weekends tend to have less people in most zones. All these plots are available in Appendix section A.2.

Between the geographical features, the inclusion or not of them did not show any significant improvement relative to not including them in the AutoML models(Table 4.7). The RMSE and MAE for no feature had slightly lower errors for no feature, but the MAPE was a bit lower. This could mean This the models without any features may not have captured the variations in crowd density accurately in terms of relative differences, and models with additional features may incorporate more relevant information, enabling them to provide better predictions in terms of relative accuracy. However, the difference is so little that might just be a random fluctuation. Considering the graph information as a feature in the GNN, since it serves a similar role as the leaf coverage, neighborhood, and sscode features, to differentiate each zone, but with additional information of their connections. It would be the feature that most impacted the models positively.

Features	MAE	MAPE	RMSE
day_of_week	27.44	1.80 x 10^{14}	61.91
"is_weekend, rain"	25.91	1.81 x 10^{14}	59.22
"day_of_week, is_weekend"	25.31	1.81 x 10^{14}	58.42
"day_of_week, is_weekend, rain"	25.91	1.81 x 10^{14}	59.22

Table 4.8: Temporal Features

4.2 Solution Applied to the Real World

The models achieved a good margin of error and performance that could be used in the city of Vila Nova de Famalicão for forecasting reasons. With this knowledge, city planners can know how many people will be in each region of the city with high precision and map the movement for the near and far future since the GNN does not show a loss in precision as the forecast gets further from the original data. With the data train test split of 70% for train and 30% for test, and considering that the whole time series was of a span of one year, the model should be able to predict for around three and a half months in the future.

A margin could be considered for precision, for example, considering the forecast for the next month instead of three and retraining the model every month. Retraining the model with incoming new data should not be a problem since it has a fast learning time. It would be a good practice if it was applied in real-world scenarios to keep it updated for upcoming new emerging patterns in the city movement. Nevertheless, one month of forecast should be more than enough for most tasks.

Other models could also be used since they achieved similar. The GNN with neighborhood clusters could be used for neighborhood administrations. The models for single zones could be applied for in-loco prediction, which means: each of the instruments that collect the density information could keep track of its own time series and could use the best model for each zone, either AutoML, AutoRegressive, or LSTM.

Near-future and far-future predictions could also be used for different ends. Near future models could predict the density for the next hours, day, or week and could be used for crowd control, preventing crowded areas, and advising the citizens how to move inside the city in cases of events that cause the accumulation of people in spaces. On the other hand, far-future, such as many weeks or more than a month forecast, can be useful for infrastructure, for example, allocating resources for a new season coming, like summer or winter.

Chapter 5

Conclusion

This chapter revises the work done in this thesis and presents the final conclusions taken after the application of the proposed solution and the analysis of the obtained results. First, are presented the main results obtained, an then are outlined possible improvements that could be done in future research in similar topics.

5.1 Main Contributions

This thesis proposes a comparison of machine learning models for Crowd Density Forecasting using mobile data from the City of Famalicão. Four kinds of models were compared AutoRegressive, LSTM Neural Networks, Graph Neural Networks, and AutoML. The city was divided into 169 zones, and the data contained the density of five moments of the day over one year for each zone. The models should be able to compute the density for future data based on the historical data given.

Four scenarios of data partition were used to compare the models, the division into the 169 zones, with one model for each zone; division into the city administrative neighborhoods, with one model for each neighborhood, detecting the densities inside the zones of each neighborhood; the division into two algorithmically determined clusters, and prediction of the densities for the zones inside this clusters; and finally one model with all zones that should be able to predict the density on each one accurately.

Additionally, extra features were added to the density data, such as weather, calendar information, region elevation, and how green the area is. These features were compared to find which features contribute more to the performance of the final model. These features are of two kinds, temporal features, the ones that apply equally to all zones, and spatial features, which could help the models differentiate the zones for the forecast.

Between the temporal features, the day of the week proved to be the best, which correlates with the intuition that the people's movement inside the city changes routinely through the week. The geographical features did not present much improvement compared to not including them.

Nevertheless, the inclusion of information about the connection between each zone using a graph in the Graph Neural Network showed that spatial features can indeed improve prediction abilities.

The best-performing model was a Graph Neural Network for each neighborhood, although the same method for all zones at the same time got very similar error rates and is more general than should be preferred. However, many of the single-zone models achieved very similar performance to the best model, although others had a very bad performance. Noticeably, the LSTM Neural Network because it has the same architecture of an LSTM layer followed by a dense layer, with the only difference being the fact that the GNN passes the input through a Graph Convolutional layer before. This suggests that the superiority does not come from the model itself but from its ability to use adjacency information to model harder zones that were harder to model individually. And is supported by the fact that density is a measure that is directly related and changes based on the connections between the zones. For a zone density to increase, one of its adjacent zones must decrease. And the Graph Neural Network, due to the use of a Graph Convolutional layer, is capable of extracting and applying this property through the graph.

Finally, the models showed to be usable by the city of Famalicão, with good results that could be used by many entities of the city and could improve the city planning with the forecasting of the crowd density. The methods for treating the data and training new models with new data as it is collected could also be used as a contribution to the city. The models are able to train over the data and then forecast future data in applicable times for real-world use and achieve precision that could also be applied to real solutions.

5.2 Future Work

Future work in the same subject could improve the results in some areas and clarify questions that were left out due to constraints. The addition of extra features than the density to the GNN could improve it even more, its results and find more about the impact of them in a spatial-sensitive model instead of the AutoML models in which it was tested.

If work is done using the same data, the graph information could be updated to include the complete graph based on data from more than one day. If flow data for more days is available, the impact of it on the density could also be measured. Knowing the precise location of each zone would enable geographical data with a better resolution and could impact the models.

The algorithmic clustering did not have the best results since cluster 1 had a significantly lower performance than cluster 2, but this second division achieved some of the best performances. The zone clustering could be revisited in order to obtain better clusters or maybe divide the first cluster into more.

Additionally, the impact of the external features was only tested in AutoML and could be expanded to other models. And the impact of each feature in the values could be explored more extensively, not only the impact on the accuracy of the forecast. If a certain feature causes the density to rise, decrease or stay the same.

Finally, other models could be compared against, especially other AutoRegressive models that could be capable of capturing the spatial relations more efficiently than the ones tested since not all of them converged.

References

[1] Rebecca Hammons and Myers Joel. Smart cities. *IEEE Internet of Things Magazine*, 2(2):8–9, 2019.

[2] Chris Chatfield. Time-series forecasting, 2005. Available online at http://dx.doi.org/10.1111/j.1740-9713.2005.00117.x.

[3] *Regression: Models, Methods and Applications*. Available online at https://doi.org/10.1007/978-3-662-63882-8_2.

[4] T. M. Mitchell. *Machine Learning*. 1997. Available online at https://books.google.pt/books?id=EoYBngEACAAJ.

[5] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*, volume 3. 2018. Available online at https://www.otexts.org/fpp3 (Accessed: June 2023).

[6] Zurich expertise for smart cities around the world. Available online at https://www.greaterzuricharea.com/en/news/zurich-expertise-smart-cities-around-world (Accessed: June 2023).

[7] Smart city korea portal. Available online at https://smartcity.go.kr/en/ (Accessed: June 2023).

[8] Copenhagen solutions lab | nordic smart city network. Available online at https://nscn.eu/Copenhagen (Accessed: June 2023).

[9] Roger D. Peng. A very short course on time series analysis, 2020. Available online at https://bookdown.org/rdpeng/timeseriesbook/.

[10] Jonathan D Cryer. *Time series analysis*, volume 286. 1986.

[11] Joos Korstanje. *The VARMAX Model*, pages 141–145. 2021. Available online at https://doi.org/10.1007/978-1-4842-7150-6_10.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 1997.

[13] Spatial and spatio-temporal modeling. Available online at https://globalhealthdata.org/spatial-and-spatio-temporal-modelling/ (Accessed: June 2023).

[14] Phillip E. Pfeifer and Stuart Jay Deutsch. A starima model-building procedure with application to description and regional forecasting. *Transactions of the Institute of British Geographers*, 5:330, 1980.

[15] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[16] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[17] Nilam Sjarif, Siti Mariyam Shamsuddin, Siti Mohd Hashim, and Siti Yuhaniz. Crowd analysis and its applications. *Communications in Computer and Information Science*, 179:687–697, 2011.

[18] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. Available online at https://www.sciencedirect.com/science/article/pii/S0169207006000239.

[19] Zahraa Al Sahili and Mariette Awad. Spatio-temporal graph neural networks: A survey. 2023.

[20] Zi Wang, Jia Hu, Geyong Min, Zhiwei Zhao, Zheng Chang, and Zhe Wang. Spatial-temporal cellular traffic prediction for 5g and beyond: A graph neural networks-based approach. *IEEE Transactions on Industrial Informatics*, 19(4):5722–5731, 2023.

[21] Yifan Li, Yu Lin, Yang Gao, and Latifur Khan. Unified spatio-temporal graph neural networks: Data-driven modeling for social science. pages 1–8, 2022.

[22] Asif Mehmood, Talha Ahmed Khan, Afaq Muhammad, and Wang-Cheol Song. Multi-class traffic density forecasting in iov using spatio-temporal graph neural networks. pages 1–6, 2022.

[23] Han Qiu, Qinkai Zheng, Mounira Msahli, Gerard Memmi, Meikang Qiu, and Jialiang Lu. Topological graph convolutional network-based urban traffic flow and density prediction. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4560–4569, 2021.

[24] Xu Zhang, Cao Ruixu, Zhang Zuyu, and Xia Ying. Crowd flow forecasting with multi-graph neural networks, 2020.

[25] X. Fu, Yu G., and Liu Z. Spatial-temporal convolutional model for urban crowd density prediction based on mobile-phone signaling data. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14661 – 14673, 2022. Available online at http://dx.doi.org/10.1109/TITS.2021.3131337 ,.

[26] Xiaojie Bu, Zebing Wei, Zhishuai Li, Xiao Wang, Yuanyuan Chen, Yunquan Song, and Yisheng Lv. Temporal-difference spatial sampling and aggregating graph neural network for crowd flow forecasting. pages 160–163, 2021.

[27] Razvan-Gabriel Cirstea, Bin Yang, Chenjuan Guo, Tung Kieu, and Shirui Pan. Towards spatio-temporal aware traffic time series forecasting–full version, 2022. Available online at https://arxiv.org/abs/2203.15737.

[28] Mariana Oliveira, Luís Torgo, and Vítor Santos Costa. Evaluation procedures for forecasting with spatiotemporal data. *Mathematics*, 9(6), 2021. Available online at https://www.mdpi.com/2227-7390/9/6/691.

[29] Zhengyang Ai, Kai Zhang, Shupeng Wang, Chao Li, Xiao-yu Zhang, and Shicong Li. A novel partition method for busy urban area based on spatial-temporal information. pages 234–246, 2019.

[30] Tascikaraoglu Akin. Evaluation of spatio-temporal forecasting methods in various smart city applications. *Renewable and Sustainable Energy Reviews*, 82:424–435, 2018. Available online at https://www.sciencedirect.com/science/article/pii/S1364032117313308.

[31] Zheng Hong, Deng Xiao, and Wenxuan Deng. A crowd pre-warning system based on mobile locators and behavior prediction, 2016. Available online at http://dx.doi.org/10.1109/ICDH.2016.035 ,.

[32] Utkarsh Singh, Jean-François Determe, François Horlin, and Philippe De Doncker. Crowd forecasting based on wifi sensors and lstm neural networks. *IEEE Transactions on Instrumentation and Measurement*, 69(9):6121–6131, 2020.

[33] M. Bessho and Sakamura K. Design and implementation of street-level crowd density forecast using contact tracing applications, 2022. Available online at http://dx.doi.org/10.1109/ISC255366.2022.9922572 ,.

[34] Renhe Jiang, Cai Zekun, Wang Zhaonan, Yang Chuang, Fan Zipei, Chen Quanjun, Tsubouchi Kota, Song Xuan, and Shibasaki Ryosuke. Deepcrowd: A deep model for large-scale citywide crowd density and flow prediction. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):276–290, 2023.

[35] Alket Cecaj, Marco Lippi, Marco Mamei, and Franco Zambonelli. Comparing deep learning and statistical methods in forecasting crowd distribution from aggregated mobile phone data. *Applied Sciences*, 10(18):6580, 2020. Available online at https://www.mdpi.com/2076-3417/10/18/6580.

[36] Dorine C. Duives, Guangxing Wang, and Jiwon Kim. Forecasting pedestrian movements using recurrent neural networks: An application of crowd monitoring data. *Sensors*, 19(2), 2019. Available online at https://www.mdpi.com/1424-8220/19/2/382.

[37] Alket Cecaj, Lippi Marco, Mamei Marco, and Zambonelli Franco. Forecasting crowd distribution in smart cities, 2020.

[38] Can Rong, Feng Jie, and Li Yong. Deep learning models for population flow generation from aggregated mobility data, 2019.

[39] Qiming Hao, Zhang Le, Zha Rui, Zhou Ding, Zhang Zhe, Xu Tong, and Chen Enhong. Urban crowd density prediction based on multi-relational graph, 2021.

[40] Claudia Bergroth, Olle Järv, Henrikki Tenkanen, Matti Manninen, and Tuuli Toivonen. A 24-hour population distribution dataset based on mobile phone data from helsinki metropolitan area, finland. *Scientific Data*, 9(1):39, 2022.

[41] Xiaojie Bu, Zebing Wei, Zhishuai Li, Xiao Wang, Yuanyuan Chen, Yunquan Song, and Lv Yisheng. Temporal-difference spatial sampling and aggregating graph neural network for crowd flow forecasting, 2021. Available online at http://dx.doi.org/10.1109/DTPI52967.2021.9540169 ,.

[42] Xu Wang, Zimu Zhou, Yi Zhao, Xinglin Zhang, Kai Xing, Fu Xiao, Zheng Yang, and Yunhao Liu. Improving urban crowd flow prediction on flexible region partition. *IEEE Transactions on Mobile Computing*, PP:1–1, 2019.

[43] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, Xiuwen Yi, and Tianrui Li. Predicting citywide crowd flows using deep spatio-temporal residual networks, 2017.

[44] Xucai Zhang, Yeran Sun, Fangli Guan, Kai Chen, Frank Witlox, and Haosheng Huang. Forecasting the crowd: An effective and efficient neural network for city-wide crowd information prediction at a fine spatio-temporal scale. *Transportation Research Part C: Emerging Technologies*, 143:103854, 2022. Available online at https://www.sciencedirect.com/science/article/pii/S0968090X2200273X.

[45] Marco Cardia, Massimiliano Luca, and Luca Pappalardo. Enhancing crowd flow prediction in various spatial and temporal granularities, 2022.

[46] Ine. Censo 2021 - ine, 2021. Available online at https://censos.ine.pt/xportal/xmain?xpgid=censos21_mainamp;xpid=CENSOS21amp;xlang=pt (Accessed: June 2023).

[47] Portal do município de vila nova de famalicão - portugal. Available online at http://www.famalicao.pt/ (Accessed: June 2023).

[48] B-smart famalicão: O caminho para uma cidade inteligente. Available online at https://www.nos.pt/empresas/transformacao-digital/transformacao-de-empresas/tendencias-e-inovacao/b-smart-famalicao (Accessed: June 2023).

[49] B-smart famalicão. Available online at http://b-smart.famalicao.pt/ (Accessed: June 2023).

[50] Anacom. Anacom factos e números - 1.o trimestre 2023, 2023. Available online at https://www.anacom.pt/render.jsp?contentId=1746135 (Accessed: June 2023).

[51] Portuguese National Institute of Statistics. Population data for famalicão, 2021. Available online at https://www.ine.pt/.

[52] Portugal holidays. Available online at https://feriados.com.pt/feriados/portugal/ (Accessed: February 2023).

[53] Open-Meteo. Global weather data, 2021. Available online at https://www.open-meteo.com/.

[54] Ranga Myneni, Yuri Knyazikhin, and Taejin Park. MCD15A2H MODIS/Terra+Aqua leaf area Index/FPAR 8-day L4 global 500m SIN grid V006, 2015.

[55] Nasa Jpl. NASADEM merged DEM global 1 arc second V001. 2020. Available online at https://doi.org/10.5067/MEaSUREs/NASADEM/NASADEM_HGT.001.

[56] Qgis project. Available online at https://qgis.org/en/site/ (Accessed: June 2023).

[57] Open street maps. Available online at https://www.openstreetmap.org/about (Accessed: June 2023).

[58] Open street maps polygons. Available online at http://polygons.openstreetmap.fr/ (Accessed February 2023).

[59] Nominatim. Available online at https://nominatim.openstreetmap.org/ (Accessed: June 2023).

[60] Iohan Soares. Empirical evaluation of prediction models of people flow github, 2023. Available online at https://github.com/IohanSardinha/Empirical-Evaluation-of-Prediction-Models-of-People-Density.

[61] Jonathon Shlens. A tutorial on principal component analysis, 2014.

[62] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. Available online at http://www.jstor.org/stable/2346830.

[63] Elbow method for optimal value of k in kmeans. Available online at https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/ (Accessed: June 2023).

[64] Keras Team. Keras documentation: Traffic forecasting using graph neural networks and lstm, 2023. Available online at https://keras.io/examples/timeseries/timeseries_traffic_forecasting/ (Accessed: June 2023).

Appendix A

Appendix

A.1 Decompositions

Different Period Decomposition for the Same Zone

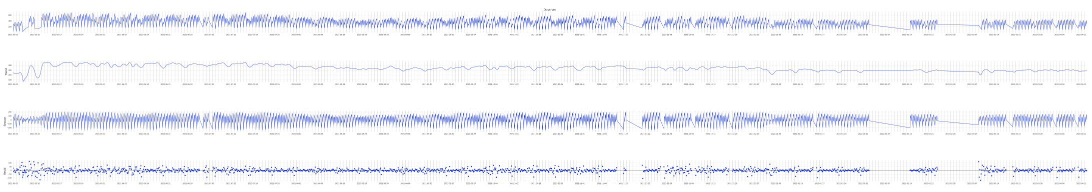


Figure A.1: One day period STL Decomposition

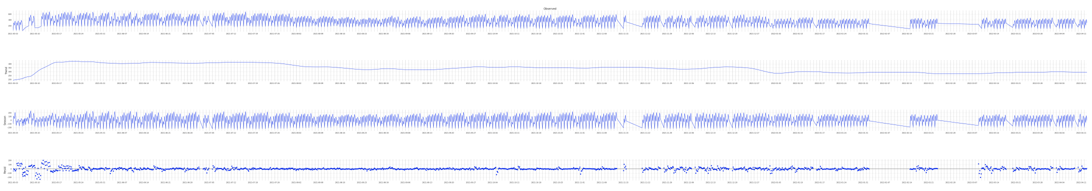


Figure A.2: One week period STL Decomposition

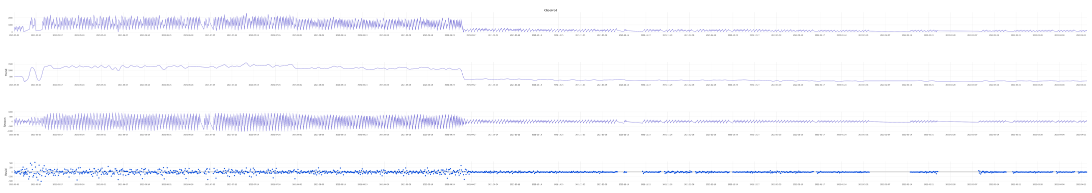


Figure A.3: One month period STL Decomposition

Examples of Zone STL Decomposition

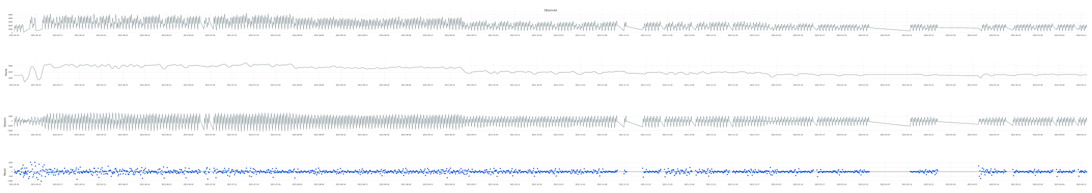


Figure A.4: STL Decomposition 2

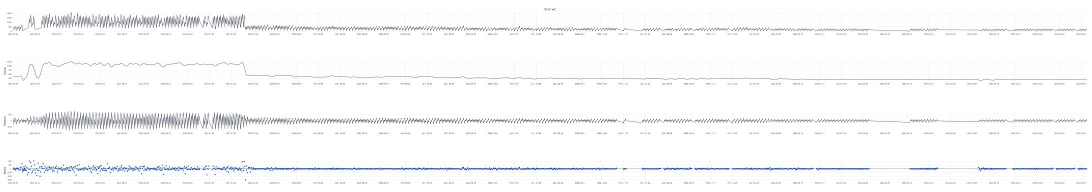


Figure A.5: STL Decomposition 3

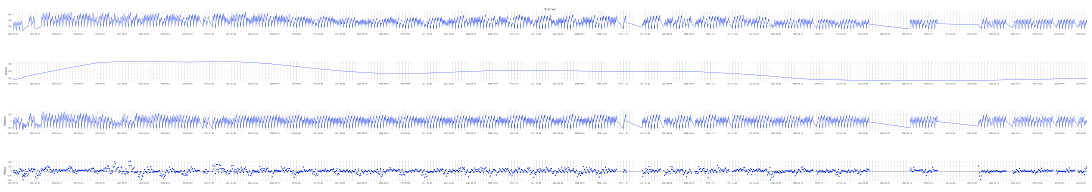


Figure A.6: STL Decomposition 1

A.2 Feature Analysis

Density vs Feature plots

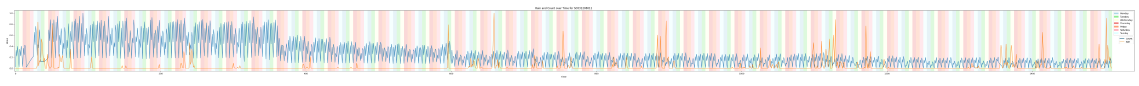
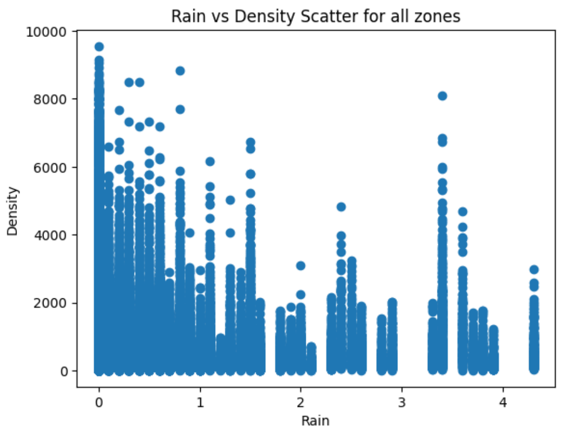
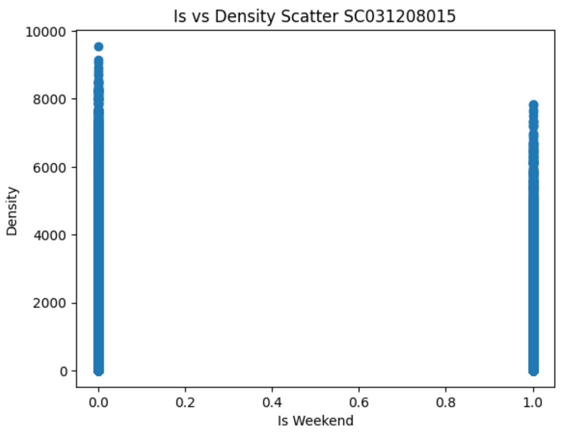


Figure A.7: Density over time, with rain and week day

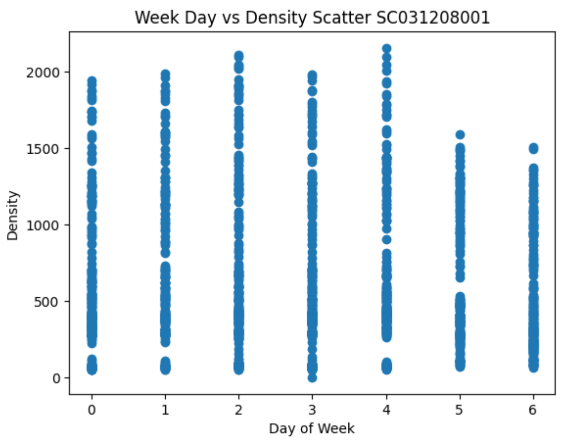


(a) Rain vs Density scatter plot

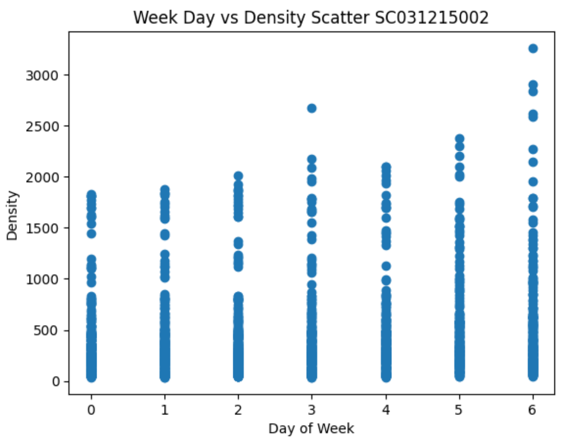


(b) Is Weekend Scatter plot

Figure A.8: Rain and Weekend features scatter plots



(a) Zone with higher activity during the week



(b) Zone with higher activity during the weekend

Figure A.9: Week day scatter plots

A.3 Error Tables

GNN Final Single Model Errors by Zone

Zone	MAE	MAPE	RMSE
SC031202001	1.27	2.8	1.34
SC031202002	1.27	2.8	1.34
SC031202003	1.27	2.75	1.35
SC031202004	1.28	2.8	1.36
SC031202005	0.78	2.28	0.83
SC031202006	1.2	2.6	1.28
SC031202007	0.78	2.38	0.84
SC031202008	1.2	2.8	1.28
SC031202009	0.75	1.16	0.81
SC031203001	0.18	0.16	0.23
SC031204001	0.46	0.62	0.49
SC031204002	0.46	0.61	0.5
SC031204003	0.47	0.62	0.5
SC031204004	0.47	0.62	0.5
SC031204005	0.46	0.62	0.5
SC031205001	0.26	0.33	0.27
SC031206001	1.21	2.46	1.28
SC031206002	1.28	2.31	1.35
SC031206003	0.9	3.51	0.96
SC031207001	0.28	0.24	0.33
SC031207002	0.2	0.18	0.26
SC031208001	0.41	0.46	0.45
SC031208002	0.37	0.4	0.41
SC031208004	0.37	0.4	0.41
SC031208006	0.36	0.4	0.4
SC031208007	0.42	0.46	0.46
SC031208008	0.29	0.35	0.32
SC031208009	0.37	0.4	0.4
SC031208011	0.38	0.41	0.42
SC031208012	0.38	0.4	0.42
SC031209001	0.33	0.39	0.34
SC031209002	0.33	0.4	0.35
SC031210001	0.29	0.35	0.32
SC031210002	0.3	0.29	0.32

Table A.1: GNN Final Single Model Errors by Zone part 1

Zone	MAE	MAPE	RMSE
SC031224003	0.28	0.34	0.3
SC031211001	1.27	2.35	1.34
SC031211002	0.69	0.84	0.75
SC031212001	0.33	0.43	0.36
SC031212002	0.36	0.46	0.39
SC031213001	0.31	0.32	0.34
SC031213002	0.46	0.6	0.49
SC031213005	0.47	0.62	0.5
SC031214001	0.34	0.3	0.4
SC031214002	0.32	0.27	0.39
SC031215001	0.28	0.34	0.29
SC031215002	0.26	0.33	0.27
SC031215003	0.25	0.33	0.26
SC031215004	0.3	0.34	0.32
SC031216001	0.87	16.33	0.93
SC031216002	0.87	4.94	0.93
SC031216003	1.28	2.79	1.36
SC031216004	0.86	5.34	0.92
SC031216005	1.27	2.77	1.34
SC031217001	0.69	1.47	0.73
SC031217003	0.29	0.26	0.35
SC031218001	0.32	0.29	0.38
SC031219001	0.58	0.37	0.64
SC031219002	0.2	0.13	0.27
SC031219003	0.28	0.22	0.35
SC031219005	0.27	0.4	0.31
SC031219006	0.27	0.21	0.36
SC031219009	0.29	0.28	0.34
SC031220001	0.19	0.17	0.26
SC031221001	0.26	0.32	0.27
SC031221002	0.23	0.21	0.28
SC031221003	0.23	0.2	0.28
SC031221004	0.19	0.18	0.2
SC031222001	0.64	0.8	0.69
SC031222002	0.64	0.8	0.69
SC031223001	1.28	2.36	1.35
SC031223002	1.27	2.31	1.34
SC031223003	1.24	4.15	1.31
SC031224001	0.3	0.35	0.33
SC031224002	0.29	0.21	0.34
SC031224004	0.36	2.16	0.42
SC031225001	0.29	0.28	0.33
SC031225002	0.22	0.24	0.24
SC031226001	0.77	2.26	0.82
SC031226002	1.02	75.12	1.08
SC031227001	0.23	0.2	0.27
SC031227002	0.22	0.19	0.27
SC031227003	0.23	0.21	0.28
SC031227004	0.23	0.2	0.27

Table A.2: GNN Final Single Model Errors by Zone part 2

Zone	MAE	MAPE	RMSE
SC031228001	0.33	0.4	0.35
SC031229001	0.98	10.19	1.03
SC031230001	0.26	0.27	0.29
SC031230002	0.3	0.25	0.34
SC031230003	0.31	0.33	0.34
SC031231001	0.22	0.13	0.28
SC031232001	0.27	0.2	0.35
SC031232002	0.27	0.4	0.29
SC031233001	0.33	0.41	0.35
SC031233002	0.71	0.98	0.76
SC031233003	0.71	1.08	0.76
SC031233004	0.27	0.34	0.28
SC031234001	0.28	0.49	0.3
SC031234002	0.28	0.49	0.3
SC031234003	0.3	0.51	0.32
SC031234004	0.29	0.5	0.31
SC031235001	0.29	0.33	0.31
SC031235002	0.3	0.35	0.32
SC031235003	0.34	0.34	0.38
SC031235004	0.28	0.33	0.31
SC031235005	0.27	0.33	0.29
SC031235006	0.27	0.33	0.29
SC031235007	0.36	0.28	0.44
SC031235009	0.39	0.31	0.47
SC031235010	0.35	0.28	0.41
SC031236001	0.26	0.33	0.28
SC031236002	0.25	0.32	0.26
SC031236003	0.33	0.39	0.34
SC031237001	0.3	0.23	0.39
SC031238001	0.32	0.24	0.42
SC031238002	0.32	0.23	0.43
SC031239001	0.29	0.29	0.31
SC031240001	0.31	0.23	0.42
SC031240002	0.34	0.33	0.4
SC031240003	0.35	0.32	0.42
SC031241001	0.34	0.45	0.36
SC031241002	0.37	0.36	0.41
SC031243001	0.26	0.32	0.28
SC031244001	0.26	0.23	0.31
SC031245001	0.3	0.23	0.39
SC031246001	0.34	0.24	0.41
SC031246002	0.51	0.3	0.57
SC031247001	0.29	0.24	0.35
SC031247002	0.26	0.33	0.27
SC031247004	0.25	0.32	0.26
SC031248001	1.24	6.98	1.32
SC031248002	1.28	2.89	1.36
SC031248003	1.2	2.9	1.28
SC031248004	1.27	2.9	1.35
SC031248005	1.19	2.72	1.28

Table A.3: GNN Final Single Model Errors by Zone part 3

Zone	MAE	MAPE	RMSE
SC031248006	1.27	2.77	1.35
SC031248007	1.19	2.82	1.27
SC031248008	1.26	2.61	1.34
SC031248009	1.29	4.55	1.37
SC031248011	1.2	2.96	1.28
SC031248012	1.2	5.28	1.28
SC031248013	1.2	4.62	1.28
SC031249001	0.3	0.36	0.32
SC031249002	0.32	0.37	0.35
Avg	0.54	1.74	0.59
Std	0.37	6.49	0.39

Table A.4: GNN Final Single Model Errors by Zone part 4

GNN Final Single Model Average Errors by Neighborhood

Neighborhood	Mae	Mape	Rmse
Bairro	0.46	0.62	0.5
Brufe	1.13	2.76	1.2
Castelões	0.29	0.32	0.32
Cruz	0.35	0.45	0.37
Delães	0.41	0.51	0.44
Fradelos	0.27	0.33	0.29
Gavião	1.03	6.43	1.09
Joane	0.32	0.27	0.38
Landim	0.23	0.23	0.26
Louro	1.26	2.94	1.33
Lousado	0.31	0.77	0.35
Mogege	0.25	0.26	0.29
Nine	0.22	0.2	0.28
Oliveira (santa Maria)	0.29	0.29	0.31
Pedome	0.29	0.28	0.32
Pousada De Saramagos	0.27	0.3	0.32
Requião	0.5	0.7	0.54
Riba De Ave	0.29	0.49	0.31
Ribeirão	0.32	0.32	0.36
União Das Freguesias De Antas E Abade De Vermoim	1.09	2.49	1.16
União Das Freguesias De Arnoso (santa Maria E Santa Eulália) E Sezures	0.31	0.23	0.41
União Das Freguesias De Avidos E Lagoa	0.18	0.16	0.24
União Das Freguesias De Carreira E Bente	0.31	0.37	0.32
União Das Freguesias De Esmeriz E Cabeçudos	0.28	0.25	0.34
União Das Freguesias De Gondifelos, Cavalões E Outiz	0.78	3.02	0.84
União Das Freguesias De Lemenhe, Mouquim E Jesufrei	0.68	15.86	0.73
União Das Freguesias De Ruivães E Novais	0.29	0.36	0.31
União Das Freguesias De Seide	0.26	0.27	0.3
União Das Freguesias De Vale (são Cosme), Telhado E Portela	0.34	0.26	0.42
União Das Freguesias De Vila Nova De Famalicão E Calendário	0.86	2.27	0.93
Vale (são Martinho)	0.36	0.4	0.38
Vermoim	0.27	0.3	0.29
Vilarinho Das Cambas	0.31	0.36	0.33

Table A.5: GNN Final Single Model Average Errors by Neighborhood

Feature Errors

The errors of the AutoML model for a single zone with different features as input.

Features	MAE	MAPE	RMSE
day_of_week	27.44	1.80 x 10^{14}	61.91
is_weekend	25.31	1.81 x 10^{14}	58.42
temperature	29.15	2.00 x 10^{14}	62.79
rain	28.15	1.78 x 10^{14}	62.79
windspeed	28.63	1.79 x 10^{14}	62.41
"day_of_week, is_weekend"	25.31	1.81 x 10^{14}	58.42
"day_of_week, temperature"	29.15	2.00 x 10^{14}	62.79
"day_of_week, rain"	28.15	1.78 x 10^{14}	62.79
"day_of_week, windspeed"	28.63	1.79 x 10^{14}	62.41
"is_weekend, temperature"	26.48	1.86 x 10^{14}	59.47
"is_weekend, rain"	25.91	1.81 x 10^{14}	59.22
"is_weekend, windspeed"	26.51	1.71 x 10^{14}	59.25
"temperature, rain"	29.62	1.94 x 10^{14}	63.14
"temperature, windspeed"	30.42	1.87 x 10^{14}	64.02
"rain, windspeed"	29.21	1.82 x 10^{14}	62.97
"day_of_week, is_weekend, temperature"	26.48	1.86 x 10^{14}	59.47

Table A.6: Features Errors part 1

Features	MAE	MAPE	RMSE
day_of_week	27.44	1.80 x 10^{14}	61.91
is_weekend	25.31	1.81 x 10^{14}	58.42
temperature	29.15	2.00 x 10^{14}	62.79
rain	28.15	1.78 x 10^{14}	62.79
windspeed	28.63	1.79 x 10^{14}	62.41
"day_of_week, is_weekend, rain"	25.91	1.81 x 10^{14}	59.22
"day_of_week, is_weekend, windspeed"	26.51	1.71 x 10^{14}	59.25
"day_of_week, temperature, rain"	29.62	1.94 x 10^{14}	63.14
"day_of_week, temperature, windspeed"	30.42	1.87 x 10^{14}	64.02
"day_of_week, rain, windspeed"	29.21	1.82 x 10^{14}	62.97
"is_weekend, temperature, rain"	27.07	1.86 x 10^{14}	59.91
"is_weekend, temperature, windspeed"	27.32	1.76 x 10^{14}	60.09
"is_weekend, rain, windspeed"	26.75	1.74 x 10^{14}	59.77
"temperature, rain, windspeed"	30.66	1.94 x 10^{14}	64.1
"day_of_week, is_weekend, temperature, rain"	27.07	1.86 x 10^{14}	59.91
"day_of_week, is_weekend, temperature, windspeed"	27.32	1.76 x 10^{14}	60.09
"day_of_week, is_weekend, rain, windspeed"	26.75	1.74 x 10^{14}	59.77
"day_of_week, temperature, rain, windspeed"	30.66	1.94 x 10^{14}	64.1
"is_weekend, temperature, rain, windspeed"	27.51	1.76 x 10^{14}	60.09
"day_of_week, is_weekend, temperature, rain, windspeed"	27.51	1.76 x 10^{14}	60.09

Table A.7: Features Errors part 2

A.4 Clustering

Zones by Clustering

The zones contained in the two algorithmically selected zones.

Zones			
SC031202002	SC031202003	SC031202007	SC031202009
SC031203002	SC031206001	SC031206003	SC031207001
SC031208007	SC031208011	SC031210002	SC031211001
SC031211002	SC031212001	SC031212002	SC031214002
SC031216001	SC031216002	SC031216003	SC031216004
SC031218001	SC031220001	SC031223001	SC031223002
SC031223003	SC031224001	SC031224002	SC031225001
SC031226001	SC031226002	SC031229001	SC031230001
SC031231001	SC031233002	SC031233003	SC031235002
SC031237001	SC031238001	SC031239001	SC031239002
SC031240001	SC031240002	SC031240003	SC031241002
SC031245001	SC031246001	SC031246002	SC031248001
SC031248007	SC031249001	SC031249002	

Table A.8: Cluster 1 zones

Zones				
SC031201001	SC031202001	SC031202004	SC031202005	SC031202006
SC031202008	SC031203001	SC031204001	SC031204002	SC031204003
SC031204004	SC031204005	SC031205001	SC031206002	SC031207002
SC031208001	SC031208002	SC031208003	SC031208004	SC031208005
SC031208006	SC031208008	SC031208009	SC031208010	SC031208012
SC031208013	SC031208014	SC031208015	SC031209001	SC031209002
SC031210001	SC031213001	SC031213002	SC031213003	SC031213004
SC031213005	SC031214001	SC031215001	SC031215002	SC031215003
SC031215004	SC031216005	SC031217001	SC031217002	SC031217003
SC031219001	SC031219002	SC031219003	SC031219004	SC031219005
SC031219006	SC031219007	SC031219008	SC031219009	SC031219010
SC031221001	SC031221002	SC031221003	SC031221004	SC031222001
SC031222002	SC031224003	SC031224004	SC031225002	SC031227001
SC031227002	SC031227003	SC031227004	SC031228001	SC031230002
SC031230003	SC031232001	SC031232002	SC031232003	SC031233001
SC031233004	SC031234001	SC031234002	SC031234003	SC031234004
SC031234005	SC031235001	SC031235003	SC031235004	SC031235005
SC031235006	SC031235007	SC031235008	SC031235009	SC031235010
SC031236001	SC031236002	SC031236003	SC031238002	SC031239003
SC031239004	SC031241001	SC031242001	SC031242002	SC031242003
SC031242004	SC031243001	SC031244001	SC031247001	SC031247002
SC031247003	SC031247004	SC031248002	SC031248003	SC031248004
SC031248005	SC031248006	SC031248008	SC031248009	SC031248010
SC031248011	SC031248012	SC031248013		

Table A.9: Cluster 2 zones