TÉCNICO LISBOA

Describing and predicting demand in Lisbon's public Bike sharing system

Cláudio Ascenso Sardinha

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Rui Miguel Carrasqueiro Henriques
Dr. Anna Carolina Nametala Finamore do Couto

Examination Committee

Chairperson: Prof. Alberto Manuel Rodrigues da Silva
Supervisor: Prof. Rui Miguel Carrasqueiro Henriques
Member of the Committee: Prof. Maria da Conceição Esperança Amado

November 19, 2020

Acknowledgments

Quero começar por agradecer aos meus orientadores, professor Rui Henriques e professora Anna Couto por todo o apoio e ajuda nestes últimos meses. Foi um percurso com os seus altos e baixo e que não conseguia acabar sem ajuda dos meus orientadores.

Tambem quero agradecer aos meus pais e a minha avo pelo apoio incondicional durante a tese e o mestrado.

Agradecer ao Dias, Tiago, Gonçalo, Loureio e Leo pela companhia no LAB 15 e no INESC. E especialmente ao Leo por ter estado comigo a treinar a apresentação. A Bela pela companhia aos domingos.

Resumo

A utilização de sistemas de partilha de bicicletas está a aumentar nas grandes cidades de todo o mundo. O funcionamento dos sistemas de partilha de bicicletas depende de uma distribuição geográfica equilibrada das bicicletas ao longo do dia. Neste contexto, a compreensão da distribuição espaço-temporal de entradas e saídas é fundamental para o balanceamento das estações e relocalização de bicicletas. Ainda assim, as contribuições mais recentes de aprendizagem profunda e de modelos de previsão baseados em distância mostram um sucesso limitado na previsão da procura dos sistemas de partilha de bicicletas . Estas observações podem ter origem em : i) à grande dependência entre a procura e o contexto meteorológico e situacional das estações; e ii) à ausência de contexto espacial, uma vez que a maioria dos preditores são incapazes de modelar os efeitos de estações vazias-cheias nas estações próximas. Este trabalho propõe um conjunto abrangente de novos princípios para incorporar fontes históricas e prospectivas do contexto espacial, meteorológico, situacional e calêndrico nos modelos de previsão da procura de estações. Para este fim, é proposta uma nova arquitetura de rede neuronal recorrente composta por LSTMs com duas principais contribuições: i) a utilização de máscaras multivariadas de séries temporais produzidas a partir de dados de contexto histórico na camada de entrada, e ii) a regularização das séries temporais previstas utilizando dados de contexto prospectivo. Para analisar o desempenho da rede neuroal recorrent, esta tese analisa tanto métodos clássicos quanto métodos mais recentes de previsão. Este trabalho também avalia o impacto da incorporação de diferentes fontes de contexto, mostrando a relevância dos princípios propostos para a comunidade, ainda que nem todas as melhorias dos modelos de previsão utilizando as diferentes fontes de contexto produzam significância estatística.

Keywords: Previsão sensível ao contexto; Dados espaço-temporais, Dados de Séries Temporais Multivariadas, LSTMs, Sistemas de partilha de bicicletas; regularização

Abstract

Bike sharing demand is increasing in large cities worldwide. The proper functioning of bike-sharing systems is, nevertheless, dependent on a balanced geographical distribution of bicycles throughout a day. In this context, understanding the spatiotemporal distribution of check-ins and check-outs is key for station balancing and bike relocation initiatives. Still, recent contributions from deep learning and distance-based predictors show limited success on forecasting bike sharing demand. This consistent observation is hypothesized to be driven by: i) the strong dependence between demand and the meteorological and situational context of stations; and ii) the absence of spatial awareness as most predictors are unable to model the effects of high-low station load on nearby stations.

This work proposes a comprehensive set of new principles to incorporate both historical and prospective sources of spatial, meteorological, situational and calendrical context in predictive models of station demand. To this end, a new recurrent neural network layering composed by serial long-short term memory (LSTM) components is proposed with two major contributions: i) the feeding of multivariate time series masks produced from historical context data at the input layer, and ii) the time-dependent regularization of the forecasted time series using prospective context data. To study the performance of the new recurrent neural network, this thesis analyses a set of state-of-the-art forecast method as baseline methods. This work further assesses the impact of incorporating different sources of context, showing the relevance of the proposed principles for the community even though not all improvements from the context-aware predictors yield statistical significance.

Keywords: Context-sensitive forecasting, spatiotemporal data, multivariate time series, LSTMs, bike sharing system, regularization

Contents

List of Tables

List of Figures

Acronyms

ARIMA Autoregressive integrated moving average

ARMA Autoregressive moving average

BSS Bike sharing system

CML Lisbon city council

DBA DTW barycenter averaging

DTW Dynamic time warping

ILU ILU Integrative Learning from Urban Data

INESC – ID Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento em Lisboa

IPMA Instituto Português do Mar e da Atmosfera

KNN K-nearest neighbors

LNEC Laboratório Nacional de Engenharia Civil

LSTM Long short-term memory

RNN Recurrent neural network

SARIMA Seasonal autoregressive integrated moving average

Chapter 1

Introduction

1.1 Motivation

In recent years, transportation dynamics are changing in large cities worldwide [1, 2, 3]. Shared mobility enables users to obtain short-term access to transportation as needed, rather than requiring ownership. Modes of shared mobility are rising in popularity, particularly bike sharing modes propelled by structural shifts in the culture and cycling infrastructures of urban systems [4, 5]. Worldwide, the total number of bikes is estimated to have increased from 700,000 bikes in 2014 towards over 18 million in 2018 [6]. To reduce the carbon fingerprint and meet this demand, over 1600 Bike Sharing Systems (BSS) are now operating and rapidly expanding [6] as they offer a reliable, low-cost and environmental-friendly mode of short-distance transportation. Most public BSS support cycling traffic along a network of docking stations, as opposed to dockless bike sharing systems [7]. Bike sharing system (BSS) can be categorised regarding the presence or not of a docking station. When a BSS does not have a docking station, it is called dockless BSS.

In contrast with other modes of transportation, the operation and planning of public BSS in metropolitan areas are hindered by unique challenges. First, the demand is unevenly distributed both geographically and temporally, with stations in residence and office areas being either frequently full or empty during peak hours, thus preventing local check-ins and check-outs and consequently hampering user trust [8, 9]. Second, bike sharing demand is affected by significant externalities, including daily variations on the users' profile and endeavors impacting their preference towards a given mode of mobility [10]. Balancing initiatives, including the ongoing bike relocation or dynamic user incentives for taking specific routes along certain time windows can be placed to counter-act this effect [9]. Yet, their efficacy is largely dependent on the ability to model and forecast the demand of BSS stations.

In this context, predicting bike sharing demand at fine spatial and temporal levels is essential for the proper operation and planning of public BSS. Despite the large attention placed by the communities of urban computing, machine learning, and intelligent transportation systems on this task [11, 12, 13, 14], BSS demand prediction is recognizably a difficult task. State-of-the-art predictors, whether grounded on classical, distance-based or neural processing principles, show unexpectedly large forecasting errors

in comparison with other transportation modes [15]. In addition to the aforementioned challenges, the difficulty on predicting demand is linked with the fact that bike sharing demand is significantly dependent on:

- the *situational and calendrical context*. The occurrence of public events and the presence of calendar-driven context – including academic breaks, festivities and weekly-monthly-yearly seasonal factors – affect demand, and can significantly alter expectations on station load [15];
- the *meteorological context*. Weather factors, including precipitation, humidity and significant deviations to the perceived temperature are known to condition user decisions [10];
- *spatial context* (station-wise interdependencies). States of full and empty station load not only hamper the demand analysis on the stations subjected to check-in and check-out impediments, but also affect the demand of nearby stations, which will receive an indirect increase in the demand for bike check-ins and check-outs that should be separated from their true demand under normal conditions. In this context, variables such as distance between stations, station capacity and other interdependence factors impact demand [16, 8].

Recent contributions have been proposed to consider calendrical constraints [10]; offer meteorological-based corrections by either segmenting data [17] or extending the learning process [18]; or model spatial dependencies using influence factors [19] or graph neural networks [20]. Despite their relevance, state-of-the-art contributions for predictive tasks generally fail to model the joint impact that these multiple sources of context exert on bike sharing demand.

In addition, existing work generally fails to separate the important role of both historical and prospective sources of context. For instance, historical weather data are important to assert the true impact of weather variables in demand, while prospective weather forecasts are essential to adjust predictions. Context-sensitive models generally tackle one of these two modes, either historical sources of context to consistently remove context-dependent factors from predictions or prospective sources to embed context-dependent factors throughout predictions.

Lisbon city council **(CML)** partnered with INESC-ID and LNEC (Laboratório Nacional de Engenharia Civil) to establish a research protocol, starting in 2019, in the context of the ILU (Integrative Learning from Urban data)[1]project with the goal of understanding transports dynamics in Lisbon. ILU aims at addressing two major tasks:

- Describe urban traffic by analysing data from sensors, spread around Lisbon to count the numbers of cars in a specific intersection, together with public transportation data;
- Predict Lisbon transportation flow using situational context information and spatial dependencies, such as football games, music concerts and among others.

To this end, ILU project attempts to analyse and integrate several data sources regarding public transportation such as Metro, Carris, GIRA, and combine them with sources containing situational context data and evaluate the impact of spatial dependencies between the different public transportation

[1] http://web.ist.utl.pt/rmch/ilu/

methods. Taking into account the main goals of the ILU project, this work addresses the two main challenges of cycling in Lisbon: analyse the GIRA's bike demand at a station and cluster levels and comprehend the influence of weather and spatial variables on GIRA's bike demand.

1.2 Thesis contribution

This work proposes a predictive deep learning approach that is able to incorporate heterogeneous sources of spatial, meteorological and calendrical context – both historical and prospective – into the time series forecasting task. To this end, we propose a new class of recurrent neural layering for context-sensitive demand prediction. Motivated by the solid performance of long-short term memory networks (LSTMs) in traffic data analysis, this work proposes a serial composition of recurrent components using two major principles:

1. to take advantage of the inherent ability of LSTMs to learn from multivariate time series data in order to model correlations between demand variables and an arbitrarily-high number of context variables derived from the available historical context. To this end, a set of simplistic yet effective masking procedures are proposed;

2. to include an additional LSTM or gated recurrent unit (GRU) layering for the time-dependent regularization of the forecasted signal using prospective context data collected along the horizon of prediction.

This proposal is motivated and assessed in the context of BSS demand prediction given its paramount relevance in this domain. In particular, our work further introduces a study case – GIRA, the public BSS in the Lisbon city (77 stations, approximately 700 bikes and 700 to 1500 daily trips) – and evaluates the impact of incorporating available sources of context in the ability to predict the demand along the GIRA's bike sharing system. The gathered results show the role of specific sources of historical and prospective context data in shaping demand forecasts, motivating the relevance of the proposed principles to support BSS balancing and planning.

When it comes to the bike-sharing system, find patterns of bike usage and understand user demand throughout the city is a necessity for CML. The Lisbon city hall needs a robust system that predicts the bike demand at a station level using weather and spatial variables. The meteorological data include temperature, wind speed, precipitation level and others. The spatial features should be able to express the influence of a full or empty station in a nearby station.

The major contributions of this work are four-fold:

- Extension of LSTM-based predictive models to incorporate heterogeneous sources of context data from masking principles

- Serial composition of LSTM components for the time-dependent regularization of the forecasted signal based on prospective context data

- Comprehensive assessment of the impact of the proposed context-awareness forecasters against state-of-the-art forecasters at different spatial and temporal granularities.
- Interactive display of GIRA's station capacity and bike-sharing demand, as well as context-enriched visualization of BSS dynamics.

1.3 Dissertation Outline

This thesis is constructed as follows:

- **Chapter 2** introduces the background with relevant concepts pertaining both the domain knowledge on bike-sharing systems and essentials on time series data analysis.
- **Chapter 3** surveys literature on incorporating context data in traffic problems and bike-sharing systems challenges.
- **Chapter 4** describes in detail the dataset, preprocessing methods, the forecast models and the new recurrent neural network architecture
- **Chapter 5** explores the results achieved using the methodology introduced in Chapter 4.
- **Chapter 6** offers a detail discussion of the results and provides closing observations.

Chapter 2

Background

This section formulates the target problem of the thesis and defines critical concepts on time series analysis, spatial data analysis, classical approaches for time series forecasting, and recurrent neural network learning.

2.1 Bike sharing system

A bike-sharing system allows users to access bicycles on an as-needed basis for one way (point-to-point) mobility and/or roundtrips.[21] BBS provides an option of transportation for short distances trips.

GIRA bike-sharing system has two entities: agents and the environment. The environment includes bikes and bike-stations. An agent is a user who needs to use a GIRA bike. An agent can also change the environment by doing **check-in** or **check-out** operation. A check-out operation is an act of removing a bike from a bike station. A check-in operation is an act of laying a bike in a station. The number of bikes in a station is a vital feature of a bike-station.

In the context of this thesis, some key concepts regarding BSS need to be defined, such as **bike demand, context data, historical traffic data, station and station cluster**. **Bike demand** is defined as the number of check-in and check-out acts that should be supported at a specific station in a given time to fill the user's needs. **Historical traffic data** are the monitored check-ins and check-outs on the Lisbon's public BBS, accompanied by information on the capacity and occupation of docks on every bike station. *Context data* are data produced from alternative urban sources of interest, including public events (situational), weather records (meteorological), seasonal determinants (calendrical), among others. A **bike station** is a set of docks for bicycles. A **station cluster** is collection of nearby stations.

Given a bike-sharing system, the **target task** in this work is to *forecast bike demand at desirable spatial and temporal levels from available traffic and context data*.

$$\boldsymbol{x} = \begin{pmatrix} x_{1,1} & \dots & x_{1,m} \\ x_{2,1} & \dots & x_{2,m} \\ \vdots & \ddots & \vdots \\ x_{n-1,1} & \dots & x_{2,m} \\ x_{n,1} & \dots & x_{k,m} \end{pmatrix}$$

Figure 2.1: Multivariate matrix

2.2 Time series

Traffic and context data are generally represented as *time series*, a set of ordered observations $\mathbf{x}_{1..T} = (\mathbf{x}_1, ..., \mathbf{x}_T)$, each observation $\mathbf{x}_t$ being recorded at a specific time point t. Time series can be *univariate*, $\mathbf{x}_t \in \mathbb{R}$, or *multivariate*, $\mathbf{x}_t \in \mathbb{R}^m$, where $m > 1$ is the multivariate order (number of variables).

Given a time series $\mathbf{x}$, the *modeling task* aims at finding an abstraction that describes $\mathbf{x}$, while the **forecasting task** aims to estimate h upcoming observations, $\mathbf{x}_{T+1..T+h}$, from available observations $\mathbf{x}_{1..T}$. *Descriptive* and *predictive* tasks typically aim at respectively modeling and forecasting a single variable (the target variable).

According to Brockwell et al. [22] the type of observations, continuous or discrete, describes a time series. A *discrete-time* time series is one in which the set T_0 of times at which observations are made in a discrete set, as is the case, for example, when observations are made at fixed time intervals. *Continuous time* time series are obtained when observations are recorded continuously over some time interval $T_0 = [0, 1]$.

If future values of a time series are exactly determined by some mathematical function such as

$$x_t = sin(\pi_1 ft) \tag{2.1}$$

the time series is said to be *deterministic*. On the other hand, if the set of values is defined in terms of a probability distribution, the time series is called non-deterministic or *statistical time series*. A statistical phenomenon that evolves in time according to a probabilistic law is called a *stochastic process*. A special kind of *stochastic process* is *stochastic stationary process*. In a strictly *stochastic stationary process*, the joint distribution of any set of observations must be unaffected by shifting the observations time forward or backward, that is the joint probability distribution of m observations $x_{t_1}, x_{t_2}, ..., x_m$, made in set of time $t_1, t_2, ..., t_m$ is the same as m observations $x_{t_1+k}, x_{t_2+k}, ..., x_{t_m+k}$ made in set of time $t_{1+k}, t_{2+k}, ..., t_{m+k}$. [23]

Time series can also be categorized on the number of variables per observation. When time-series observation has a single variable, $m = 1$ the time series is said to be *univariate*. Contrarily, when an observation has more than one variable, the time series is said *multivariate*, $m > 1$. In a *multivariate time series* an observation is itself a time series. Method of *multivariate* time series analysis study the relationships between the multiple variables that compose a time series observation.

A multivariate time series x is composed by a sequence of n time points, where each time point has m variable associated, $m > 1$: A multivariate time series, such as the time series in Figure 2.1, can be

roughly seen as a set of univariate time series x_{it}, where i indexes the variable and t is the time. Each row of the matrix $\boldsymbol{x}$ can be interpreted as a univariate time series. Form now on, an univariate time series will be represented by x_t.

According to Jain et al. [24] a time series, x_t can be expressed as a sum of four components, *trend, seasonal, cyclical, and error*. *Trend* component reflects a long-term movement of a time series. *Seasonal* component reveals a periodic movement in a series that repeats itself. *Cyclical* component represents long term oscillations occurring in a time series with a period of years or decades. *Error or residual* component is the sum of the random and irregular effects that are not included in the trend, cyclical, nor seasonal components.

If the time series is affected by interventions, which are the results of exogenous shocks to the series, then intervention components can be included as separate components. This information is incorporated into an equation called the decomposition equation:

$$x_t = \mu_t + \gamma_t + \omega_t + \varepsilon_t \tag{2.2}$$

$$\mu_t = 2\mu_{t-1} + \mu_{t-1} + \eta_t + \phi_1 \eta_{t-1} + \phi_2 \eta_{t-2} \tag{2.3}$$

$$\sum_{j=0}^{n} \gamma_{t-1} = \beta_{t-1} + \omega_t + \alpha_1 \omega_{t-1} + \alpha_2 \omega_{t-2} \tag{2.4}$$

$$\sum_{j=0}^{n} \beta_{t-j} = \zeta \tag{2.5}$$

Equation 2.2 is the decomposition equation, equation 2.3 is the component model for the trend, 2.4 and 2.5 are the equations representing the seasonal component model. Equation 2.2 represents a decomposition of a time series x_t, where μ_t is the trend,γ_t is the seasonal, ω_t is the cycle. In the equation 2.4 $beta_t$ represent the slope of the seasonal component. The random errors, ε_t and ζ_t in equations 2.2 and 2.5 are assumed to be mutually uncorrelated, each having zero mean and constant variance. The random errors μ_t and ω_t in equations 2.3 and 2.4 respectively are mutually, but not serially uncorrelated. The ϕ_1, ϕ_2 , α_1, α_2 are parameters of the moving average processes in these two equations to be estimated.

2.3 Autoregressive and exponential smoothing operations

Given a time series,x_t, a time series forecaster aims at predicting its upcoming values, $x_{T+1},..,x_{T+h}$, using the previously observed values, $x_T, x_{T-1}, ..., x_{T-k}$. Under this definition, next two sections describe the next two sections will describe three classical forecasting approaches.

2.3.1 ARMA model

Stationary time series models can be modeled using the following three general classes, *autoregressive (AR)* models, *moving average (MA)* models, or a combination of the two.

A *autoregression model* (**AR**) uses a linear combination of past values regarding the variable of interest to forecast future values. The term autoregression indicates that it is a regression of the variable against itself. Thus, an autoregressive model of order p can be written as [25]:

$$x_t = c + \sum_{j=1}^{p} \alpha_j x_{t-j} + \varepsilon_t \tag{2.6}$$

where α_j are the coefficients that model the linear combination, ε_t the random error, and p as the order of the model. We refer to this model as an AR(p) model, an autoregressive model of order p.

Rather than using past values of the forecast variable in a regression, a moving average model (**MA**) uses past forecast errors in a regression-like model.

$$x_t = c + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j} \tag{2.7}$$

where β_j are coefficients that model the linear combination, ε_t is the error at time point t, and q is the order of the MA model.

An autoregressive moving average process (**ARMA**) is the combination of the equations 2.6 and 2.7. Can be expressed as:

$$x_t = c + \sum_{j=1}^{p} \alpha_j x_{t-j} + \varepsilon_t + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j} \tag{2.8}$$

where p is the order of the **AR** model and q is the order of the **MA** q model. A **ARMA** model is denoted as ARMA(p,q)

2.3.2 ARIMA model

A non-seasonal **ARIMA** model is obtained by joining the differencing with autoregression and a moving average model. ARIMA is an acronym for AutoRegressive Integrated Moving Average. ARMA can be expressed by :

$$x'_t = c + \sum_{j=1}^{p} \alpha_j x'_{t-j} + \varepsilon_t + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j} \tag{2.9}$$

where x'_t is the differenced series. The "predictors" on the right-hand side include both lagged values of x_t and lagged errors. This is called an ARIMA(p,d,q) model, where:

- p = order of the autoregressive part;
- d = degree of first differencing involved;
- q = order of the moving average part.

White noise	ARIMA(0,0,0)
Random walk	ARIMA(0,1,0) with no constant
Random walk with drift	ARIMA(0,1,0) with a constant
Autoregression	ARIMA$(p,0,0)$
Moving average	ARIMA$(0,0,q)$

Table 2.1: Representation of general time series model using ARIMA

ARIMA model can be used to represent an important time series concept. Table 2.1 shows some concept examples.

The parameters of an ARIMA model explain some characteristics of a time series. The degree of differencing d in an ARIMA model describes the long-term trend. The long-term trend is influenced by the constant value of c. Some examples of the relation between d and c are shown below.

- If c = 0 and d = 0, the long-term forecasts will go to zero.
- If c = 0 and d = 1, the long-term forecasts will go to a non-zero constant.
- If c = 0 and $d = 2$, the long-term forecasts will follow a straight line.
- If $c \neq 0$ and $d = 0$, the long-term forecasts will go to the mean of the data.
- If $c \neq 0$ and $d = 1$, the long-term forecasts will follow a straight line.
- If $c \neq 0$ and $d = 2$, the long-term forecasts will follow a quadratic trend.

Usually to find the values for p and q in an ARIMA model plotting the data is not enough. ACF and PACF plots are needed to determinate the values for p and q. Auto-correlation function (ACF) measures the autocorrelations values between x_t and x_{t-k} for different values of k and the PACF measures the relationship between x_t and x_{t-k} after removing the effects of lags $1,2,3,\ldots,k-1$.. Now if x_t and x_{t-1} are correlated, then x_{t-1} and x_{t-2} must also be correlated. However, then x_t and x_{t-2} might be correlated, simply because they are both connected to x_{t-1}, rather than because of any new information contained in x_{t-2} that could be used in forecasting x_t. To reduce this problem, the partial autocorrelation function **(PACF)** is used.

2.3.3 SARIMA

A classic ARIMA model can only be used to describe non-seasonal data. However ARIMA model is capable of module seasonal data with the addition of some terms. A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models we have seen so far. It is written as follows:

$$ARIMA \quad \underbrace{(p,d,q)}_{\text{Non-seasonal terms}} \quad \underbrace{(P,D,Q)m}_{\text{Seasonal terms}} \tag{2.10}$$

where m is the number of observations per seasonal period. For simplicity, uppercase will be used to represent the seasonal parts of the model and lowercase notation for non-seasonal parts of the model.

The seasonal and non-seasonal parts of an ARIMA model are similar. The difference between them lies in the backshifts needed to take into account the seasonal period. For example, an ARIMA $(1,1,1)(1,1,1)_6$ model represent data with a biannual cycle. Can be express as follow

$$(1-\alpha_1 B)(1-\theta_1 B^6)(1-B)(1-B^6)x_t = (1+\beta_1 B)(1+\beta_1 B^6)\varepsilon_t \tag{2.11}$$

The seasonal terms of a SARIMA model (P,D,Q) can be seen in a seasonal PACF and ACF. For example, an ARIMA $(0,0,0)(0,0,1)_6$ model will show a spike at lag 6 in the ACF but no other significant spikes and exponential decay in the seasonal lags of the PACF (i.e., at lags 12, 24, 36, ...). Correspondingly, an ARIMA $(0,0,0)(0,0,1)_6$ model will show an exponential decay in the seasonal lags of the ACF and a spike at lag 6 in the ACF but no other significant spikes;

2.3.4 Holt–Winters algorithm

Another famous and classical time series forecast family are the Holt–Winters model. Holt-Winters' methods are based on exponential smoothing operations.

Using the naïve method, all forecasts for the future are equal to the last observed value of the series.

$$\hat{x}_{t+k} = x_t \tag{2.12}$$

for $h = 1,2,3,...$ Hence, the naïve approach assumes that the more recent value observed is the only time point needed to forecast future values. Also in the naïve approach all the previous observations provide no information for the future. This can be thought of as a weighted average where all of the weight is given to the last observation.

Using the average method, all the future values equal to a simple average of the p observed data,

$$\hat{x}_{t+h} = \frac{1}{p}\sum_{j=1}^{p} x_j \tag{2.13}$$

for $h = 1,2,...$ Hence the average method assumes that all the previous time data points have the same importance, therefore have the same importance when generating forecasts.

The equations 2.12 and 2.14 represent two extremely different forecast methods. It may be sensible to attach larger weights to more recent observations than to observations from the distant past. This is the idea behind the **simple exponential smoothing (SES)**. The forecast time point is calculated using weighted averages, where the weights decrease exponentially as observations go further into the past. The smallest weights are associated with the older observations

$$\hat{x}_{t+1} = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + ... \tag{2.14}$$

where $0 \leq \alpha \leq 1$ is the smoothing parameter. If the α is close to 0, more weight is given to observations from the more distant past. If the α is close to 1, more weight is given to more recent observations.

In Holt [26], based on the exponential smoothing, developed a forecasting model that allowed the

forecasting of data with a trend. The forecast model can be expressed with the following three equations:

$$\textit{Forecast equation} \qquad \hat{x}_{t+1} = \ell_t h b_t \tag{2.15}$$

$$\textit{Level equation} \qquad \ell_t = \alpha x_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{2.16}$$

$$\textit{Trend equation} \qquad b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \tag{2.17}$$

where ℓ_t denotes the estimate level of the time series at time t, b_t denotes the estimate values of the trend at time t, $alpha$ is the smoothing parameter for level, β^* is the smoothing parameter for the trend, $0 \leq \beta^* \leq 1$.

In 1960, Winters[27] extended the previously described Holt method to capture seasonality. The **Holt-Winters** seasonal method is composed by the forecast equation and three smoothing equations, one for the level ℓ_t, one for the trend β and one for the seasonality s_t, with corresponding smoothing parameters, α, β^* and γ.

Holt-Winters method has two variations that differ in terms of the seasonal component. The seasonal component can be described by two models: **additive** and **multiplicative**. The additive model is used when the seasonal component variations are constant, while the multiplicative model is used when the seasonal variations are changing proportionally to the level of the series. In the additive model, the seasonal component is expressed in absolute terms, while the multiplicative model the seasonal method is expressed in relative terms. In both methods, an adjustment needs to be done to the level equation, in the additive method the adjustment is done by subtracting the seasonal component, while in the multiplicative method the adjustment is done by dividing the level equation by the seasonal component.

The **additive method** can be expressed as follow:

$$\hat{x}_{t+h} = \ell_t + hb_t + s_{t+h-d(k+1)} \tag{2.18}$$

$$\ell_t = \alpha(x_t - s_{t-d}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{2.19}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \tag{2.20}$$

$$s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-d}, \tag{2.21}$$

where k is the integer part of $(h-1)/d$, which ensures that the estimates of the seasonal indices used for forecasting come from the final year of the sample. The level, trend, and seasonal equations, respectively in equations, 2.19, 2.20 and 2.21, equations are weighted average, with the corresponded weights α, β^* and γ. The level equation shows a weighted average between between the seasonally adjusted observation $x_t - s_{t-d}$ and the non-seasonal forecast $\ell_{t-1} + b_{t-1}$. The seasonal equation shows a weighted average between the current seasonal index, $x_t - \ell_{t-1} - b_{t-1}$, and the seasonal index of the same season(i.e., d time periods ago).

The **multiplicative model** can be expressed as follow:

$$\hat{x}_{t+h} = (\ell_t + hb_t)s_{t+h-d(k+1)} \tag{2.22}$$

$$\ell_t = \alpha \frac{x_t}{s_{t-d}} + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{2.23}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \tag{2.24}$$

$$s_t = \gamma \frac{x_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-d} \tag{2.25}$$

The multiplicative model follows the same logic as the additive model. The difference is that the composition of the observed time series is given by the product (instead of the sum) of the components.

2.4 Neural networks for time series analysis

Machine learning is a sub-area of artificial intelligence that attempts to develop methods so that the machine learns from experience. Machine learning encompasses different kinds of methods such as unsupervised, supervised and reinforcement learning. The main difference between them is how the method uses ground truth.

This thesis will focus on forecasting bike demand. This forecasting can be done using supervised learning methods, namely regression. According to Bishop et. al. [28], the goal of regression is to predict the value of one or more continuous target variables given m-dimensional x from m input variables. Regression can be done through multiple machine learning methods such as linear regression, decision tree regression, neural networks, and others.

2.4.1 Linear neural network

The neural network learning algorithm is inspired by neuroscience, in the sense that a unit, neuron, receives many inputs and produce an output. In the context of machine learning, a neuron is composed of multiple inputs and an activation function. The input of the activation is a linear combination of inputs of the neuron. A neural network is a set of neurons, those neurons are organized into layers. A basic neural network has three kinds of layers: input layer, hidden layer, and output layer.

A neural network model can be described as a series of functional transformations. To begin a neural network, it is needed to construct p linear combination of the input variables $x_1, .., x_m$, in the form

$$a_j = \sum_{i=1}^{m} w_{ji}^{(1)} + w_{j0}^{(1)} \tag{2.26}$$

where $j = 1, ..., p$, and the superscript *(1)* indicates that parameters correspond to the first 'layer' of the network, **input layer**. The w_{ji} values are known as weights and w_{j0} correspond to the bias value. The results of the equation 2.26, are known as *activations*. For each *activation* is applied a nonlinear *activation function*,

$$z_j = h(a_j) \tag{2.27}$$

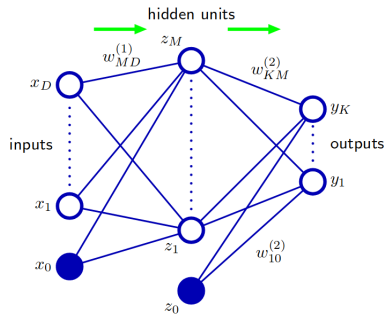


Figure 2.2: Neural network with one hidden layer

The values z_j correspond to *hidden units* in the context of neural networks. The *activation function* varies between application, but normally a sigmoidal function is used such as, logistic sigmoid or hyperbolic tangent function. Following 2.26, the values z_j are combined to give *output unit activations*

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} + w_{k0}^{(2)} \tag{2.28}$$

where $j = 1,...,k$ correspond to the numbers of outputs. Similarly to the first layer, $w_{kj}^{(2)}$ corresponds to the weight values of the second layer and w_{k0} is bias value. Finally to calculate the output values y_k an appropriate activation function is applied.

Linear neural network can be used to predict time series, by feeding the past values as inputs.

2.4.2 Recurrent neural network

Recurrent neural networks (RNN) were proposed by Rumelhart et al.[29] in 1986 and are a family of neural networks for processing sequential data. Recurrent neural network are a specialized neural network for processing a sequence of values, $x_1, x_2, ...x_T$..

These neural networks are termed recurrent because they follow a recursive pattern to calculated the output. For example, to calculate the value of the state at time x_t the value of the state at time x_{t-1} is needed and to calculate the value at the state x_{t-1} the state x_{t-2} is needed, so on and so forth.

Recurrent neural networks own one important characteristic that differs them from a feed-forward neural network: allowing information from previous computations steps to be preserved, in other words, a RNN possesses an internal memory. For example, if the RNN is used to predict the bike demand in a given time, it can preserve from previous days at the same time useful information such as number of bikes, number of check-in and number of check-outs

A RNN can be represented in two different graphs: folded and unfolded graphs. In Figure 2.3, the left side is the unfolded graph and the right is folded graph.
where h_t is the state at time t, o_t is the output at time t and b represent the bias component.

The states at Figure 2.3 are given by:

$$h_t = f(h_{t-1}, x_t) \tag{2.29}$$

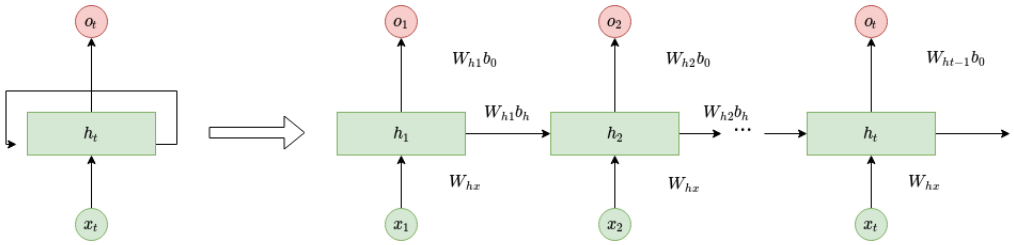


Figure 2.3: Recurrent neural network

when replacing f by a activation function, such as $tanh$, h can be describe as follow:

$$h_t = tanh(W_{hh}h_{t-1}, W_{hx}h_t), \tag{2.30}$$

where W is the weight vector, h represent a hidden layer, W_{hh} is the weight of the previous hidden layer, W_{hx} is input weight, and $tanh$ is the activation function.

A RNN has problems, namely the **vanishing gradient problem**. The Vanishing gradient problem happens when to predict the future value the RNN needs past values and the time gap between the future and the needed past values is large. Gradient are the values used to update the weights of the neural network. Vanishing gradient problem occurs when the gradient is close to zero which leads to the neural network to stop learning.

Gradient vanishing problem can be solved using a **Long Short Term Memory networks(LSTM)**. LSTM is a kind of recurrent neural network that can learn long-term dependencies. RNN has the form of a chain of repeating modules of neural networks like Figure 2.3. LSTMs also have this chain repeating modules, but the repeating module has a different architecture when comparing with a feed-forward neural network.

Long Short Term Memory (LSTM) is a kind of recurrent neural network that can learn long-term dependencies. An LSTM network has two major elements hidden state h and cell state c ,which plays the role of memory. The transformation from c_{t-1} to c_t 2.4, as four: **forget step**, **candidate state step**, **state generation step**, and **output step**.

In figure 2.4 is illustration an LSTM architecture where the boxes represent each step, i.e, yellow box is forget step, the orange box is candidate state step, the blue box is state generation step and brown box in the output step.

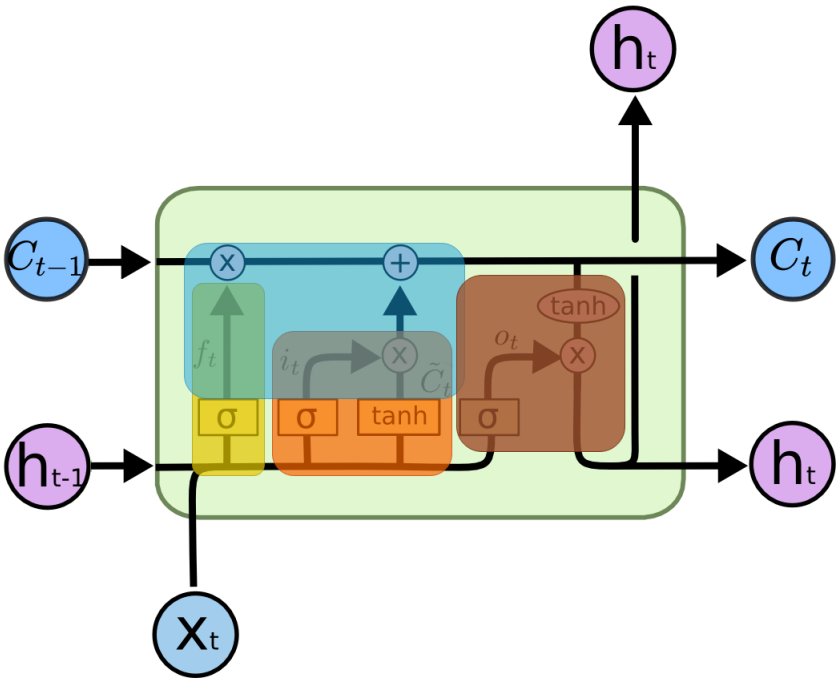


Figure 2.4: LSTM architecture.

The **forget step** decides how much information from the old state will be thrown away. This decision is made by a sigmoid layer called the 'forget gate layer'. The sigmoid translates the old state, h_{t-1}, and input, x_t, and outputs a number between 0 and 1. This output value decides how much information will be kept. The forget gate can be expressed by:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.31}$$

where the operator [] represents the concatenation of h_{t-1} with x_t.

In the **candidate generation step** is decided what new information will be stored in the cell state. The **input gate layer** is composed of a sigmoid activation function. This function decides which values will be updated. Next, a hyperbolic tangent function *tanh* creates a vector of new candidate values, $\hat{C}_t$, that could be added to the state, in the next step, the values from the forget and candidate generation steps will be combined to update the state.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.32}$$

$$\hat{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_C) \tag{2.33}$$

The **generation step** is where the old information and new candidate state will be merged into a new state. The new state, C_t is given by:

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.34}$$

In the **output step**, a filtered version of the current state, C_t, will be outputted. First, a sigmoid

function, (2.35), decides what part of the state will be outputted. Then, the state cell will be passed through a $\tanh$ function to normalize the values between -1 and 1. Finally, multiplication is done to filter the state cell based on the value of the sigmoid outputs.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.35}$$

$$h_t = o_t * \tanh(C_t) \tag{2.36}$$

2.5 Context

Diverse sources of urban context data are known to have soft-to-strong correlations with traffic dynamics, suggesting their relevance to guide descriptive and predictive learning tasks. These sources can be divided according to:

i *calendrical* sources, providing relevant information associated with academic and holiday periods, as well as daily, weekly, monthly and yearly seasonality;

ii *situational* sources, including historical and prospective public events (such as conventions, festivals, concerts, sport events) and road interdictions (including construction works);

iii *meteorological* sources, comprising observed and forecasted weather variables (such as perceived temperature, humidity, wind intensity, precipitation, visibility);

iv *spatial* sources capturing geographical dependencies with potential value for traffic data analysis [30].

2.6 Problem formulation

Recall, this works tackles the problem of **describing and predicting the check-ins and check-outs** for the nodes of a bike-sharing system.

Given a BBS with N nodes (stations), let the **check-ins** at the time interval t be denoted as $I^t = [ci_1^t, ci_2^t, ..., ci_N^t]$. Let the **check-out** at the time interval t be denoted as $O^t = [co_1^t, co_2^t, ..., co_N^t]$. Let the **bike demand** at the time interval be equal the **check-ins**.

Given the definition of check-ins, check-outs and bike demand, the **bike demand forecast task** is predicting the bike demand using historical data (series of observed check-ins and check-outs within the BSS) and available context data. This task aims at forecasting the expected number of check-ins and check-outs at a specific station or cluster of station or to all network of stations.

Chapter 3

Related work

This section is divided in three subsections: *data description, bike demand prediction,* and *context-sensitive analysis*. The *data description* subsection describes the different approaches to the BBS problems tackled in the literature. The *prediction* subsection outlines which models were applied to predict bike demand. Subsection *context* surveys which sources of context information were incorporated with historical data to solve the bike demand problem.

3.1 Descriptive approaches to bike demand

In the context of the bike-sharing system, the literature explores three major problems: predict bike demand at a station or a cluster level, predict destination stations based on the origin station and rebalance the number of bikes.

To rebalance the number of bikes between stations, Yexin li et al. [31] developed a forecaster model to predict the check-ins and check-outs at a station granularity. The proposed prediction model is composed of five major components: 1) A bipartite clustering algorithm which creates clusters stations based on geographical locations and historical transition patterns, 2) Learn the entire traffic model using historical data, 3) Allocate the proportion of the whole traffic model to each cluster, 4) Create an inter-cluster transition prediction model to predict the bike transition probability between the clusters, and 5) Predict the trip duration.

In Junming Liu et al. [32], the authors are dealing with three different bike-sharing systems challenges: 1) predict pick-up demand using a similarity weighted K-Nearest-Neighbor, 2) predict the station drop-off numbers by adopting an inter-station bike transition model considering the pick-up information of the nearby station and 3) solve the bike rebalancing problem to minimize total relacing travel distance. The proposed model starts by removing outlier stations and creating clusters among the non-outliers stations using a constrained K-centers algorithm. Then the rebalance of bikes is done within each group with the optimal route. Initially, a mixed-integer non-linear programming (MINLP) model with lazy constrains was used to attain the optimal rebalancing route within clusters. The MINLP took into account multiple factors such as the number of operating vehicles, the capacity limit of the vehicles, and trans-

portation distance. The MINLP was not solvable due size of the problem.

Yang et al.[33] identified two types of entities in a bike-sharing system, active objects (users) and reactive objects (bikes). Users through check-ins and check-outs actions can change the number of bikes in a station, which can lead to unbalanced stations. To evenly distributed bikes throughout the stations, Yang et al. proposed a model that describes the mobility pattern of bikes in a bike-sharing system based on user mobility.

Giot et. al[34] prediction model goal is different compared to other articles in the literature. Normally in the literature such as [33] and [35], the prediction models are used to re-balance the number of bikes across the bikes stations, Giot et al. aims to enable the user to create better plan trips. The prediction model was trained and tested with a dataset from Washington DC city with two years worth of data. The dataset provides the following features season (spring, summer, fall, winter), a holiday boolean, a weekday boolean, a working day boolean, type of weather(runny, sunny), temperature, felling temperature, humidity, wind speed, the number of bikes used by casual and registered users. For each sample were added two features the week number and number of bikes that were available one hour before, two hours before up to twenty-four-hour.

Tran et al.[36] proposed a linear regression model to estimate the bike-sharing flow during peak periods on weekdays. To learn the linear regression model was used a dataset from Vélo'v in Lyon city with trips information on a minute-by-minute level. Each trip had information about departure and arrival station, the date and hour of check-in and check out and the type of subscriber. In terms of subscribers was analyze two types of bike-sharing users: long-term subscribers with an annual subscription and short-term subscribers who have a one-day bike-sharing subscription. Before learning the regression model they aggregated the bike-sharing trips per hour, removed trips with less than three minutes and more than three hours, removed weekends trips and removed trips made during July and August. The authors considered two peak hours during weekdays a morning peak, between 7 am and 9 am and evening peak between 5 pm and 8 pm.

Ai et al. [37] proposed a forecast model to predict short-term spatiotemporal distribution. A long short-term memory(LSTM) with a convolution neural network (CNN) was used as a forecasting model.

Chai et al. [38] aim to predict the bike flow, number of check-ins and check-outs during a period t, at station level. The bike-sharing network was represented by a graph. In the proposed graph, nodes represent the stations and the edges represent the relations between stations. To validate the proposed solution two datasets were used one from New York City and another from Chicago city.

3.2 Predictive approaches to bike demand

When approaching a bike demand prediction problem a lot of factors should be taken into account. In the literate, multiple approaches have been proposed including, gradient boosting regression Tree [31], Graph Convolutional Neural Network [35] and among others.

In the article [31], Yexin li et al. was proposed a hierarchical prediction model to predict the bike demand at the cluster level in New York City and Washington D.C. datasets. The authors started by

grouping the bikes stations into clusters, taking into account two major proprieties of factors the geographical data and similar transition patterns. To discover clusters Yexin li et al. used a bipartite station clustering algorithm. The algorithm is an iterative method where at each iteration the following three steps are performed, Geo-Clustering, t-matrix generation, and t-clustering. The next step in the hierarchical prediction model, a Gradient Boosting Regression Tree (GBRT) is used to predict the traffic of the entire city, the number of check-outs of the entire city. To learn a good GBRT is necessary to use features regarding time and meteorology. A multi-similarity-based inference model is used to allocate the predicted traffic to each cluster. This model uses three similarity functions, Time similarity, Weather similarity, and Temperature-windspeed similarity. After leaning the check-out for each cluster, this model predicts the check-in numbers based on the predicted check-out numbers using an Inter-cluster Transition Matrix. Each entry in this matrix correspond to the probability to ride a bike from $cluster_i$ to $cluster_j$ in time t. To evaluate the prediction model were used RMSL (Root Mean Squared Logarithmic Error) and ER (Error Rate) as error functions.

Predicting bike demand can be done by applying different types of models, Junming Liu et. al [32] proposed a Meteorology Similarity Weighted K-Nearest-Neighbor model (MSWK) to predict the pick-up demand at a station level using a dataset form New York City. The proposed method uses a similarity measurement, this measurement is a linear combination of the following three similarity functions, weather similarity function, temperature similarity function, and humidity-wind-speed-visibility similarity function. The MSWK model uses the similarity measurement to find the *K* most similar days to the target day. The weights of different similarities functions are trained to reach minimum prediction absolute error using a brute force approach. The bike station drop-off prediction is given by the historical trip records, using the inter-station bike transition (ISBT) predictor. The number of bikes that will be drop-off at a station *j* from station *i* is a relation between the total pick-up demand at a station *i* and the total number of trips from station *j* to a station *i*. Not all the trips made from station *i* to station*j* will have the same duration, i.e, commuters will find the fastest path to reduce trip duration and tourists trip duration will vary. The trip duration distributions are fitted by two Gaussian functions. Junming Liu et. al also attempted to solve the station rebalancing problem using an Adaptive Capacity Constrained K-centers Clustering. This model starts by creating station clusters. The authors proposed three major factors that influence the clustering method distance between stations, the optimal number of bikes at each station, and outlier detection. The clustering algorithm begins by assign random stations as clusters centers, then each station is attributed to a cluster. For each cluster, if the capacity of the vehicle is less than the number of bikes need to balance all the stations, some stations are removed from the cluster. The stations are removed from the cluster until all the bikes needed to balance the cluster can be handled by one vehicle. The first station to be removed from the cluster is the closest station to another cluster. For each station removed from the cluster, a new cluster label is assigned. For each unlabeled station, the new cluster label is determined by the total balance of a cluster and the distance between the station and its nearest station in the cluster. The station must far from a cluster center are label first to ensure that outliers scattered at the central region of the studied area, and can be easily covered by other clusters. To evaluate the results of the MSWK model the following models

were used: HM(historical mean), MSI(Multi-similarity-based inference), IPPI (Inhomogeneous Poisson Process Inference), MSEWK (Multi-similarity-equally-weighted KNN). Junming Liu et. al found that the optimal number of neighbors to predict the pick-up demand is 23 during the weekdays and 10 during the weekends. The proposed MSWK model outperformed the other models with an MAE of 1.5923.

The mobility model proposed by Yang et al. [33] can be decoupled into two parts, predicting the number of check-ins, and predicting the number of check-out in a certain interval time. When predicting the check-in numbers at a station i, they identify two check-ins categories bikes that departed before t_{now} and bikes that did not yet departed. To predict the number of check-in bikes, Yang et al. takes into account the number of check-outs done in a station j, the transfer probability from station j to station i and feasible trip duration. To reduce the computation time, the authors removed trips above three hours.

According to Yang et al. [33], the number of check-outs are mutually independent between stations but are subject to external factors such as time factors (rush-hour, weekday, weekends) and meteorology. Since the check-outs are depending on these external factors the authors applied a random forest model. The features used in the random forest were divided into two groups, online and offline features. The offline features are composed of time factors(weekdays vs weekends/holidays and time of the day) and Meteorology factors (humidity, visibility and wind speed). The online features are the number of check-outs at a station level. The random forest was composed of three decision trees, each tree had random samples from the original dataset as train dataset. Each tree learned using the followings steps: at each node, some random feature were chosen as candidate split variables, the best split feature was found by the feature that maximizes the homogeneity of the two resulting groups. The authors used the mean square error (MSE) function as the homogeneity criterion.

In the article [34], Giot et al. studied how regression models performed comparing with baseline classifiers. They choose three baseline classifiers mean value (mean number of bikes on the dataset), mean hour (the mean number of bikes on the training set at this specific hour), and last Hour (the number of bikes of the latest hour available). The regressors used were Ridge Regression, Adaboost Regression, Support Vector Regression, Random Forest Regression, and Gradient Tree Boosting Regression. To evaluate the performance of each prediction system, the dataset was split into training and validation datasets. The authors had to ensure that the validation dataset only contains datapoints gather after all the data points of the training dataset. The evaluate metric used was the Root Mean Squared Difference.

Chai et al.[38] to solve the bike demand problem at a station level proposed a multi-graph convolutional neural network model. Their predicting model can be separated into three parts: **Graph Generation, Multi-Graph Convolution** and **Prediction Network**. In the **graph generation** three graphs are created distance graph, interaction graph, and correlation graph. The values of the adjacency matrix represents the type of relation between stations. In the distance adjacency matrix, the distance between stations corresponds to the matrix value. The values of the interaction graph are the number of rides between stations. The values of the correlation graph are the Pearson correlation between stations. In the Multi-Graph convolution step, previously three graphs are merged. The graphs are combined by a weighted sum of the adjacency matrices at the element level. A convolution layer is applied on top of the merged graph to capture spatial correlations between stations. The prediction network has two

major goals, find the temporal pattern and predict the target value. Learning the temporal correlation is done through an encoder-decoder with LSTM. Encoder receives as input the convolution result sequence. The final state of the encoder is combined with external features to a fully connected network to predict the bike demand. To evaluate the proposed solution, the authors split the data into a train-validation-test. The test data were the last 80 days of each dataset, the 40 days before the test data are validation data, and the remaining data is used to train the model. The proposed solution was compared with ARMA, ARIMA, and LSTM. The multi-graph convolutional neural network model outperformed the baseline model, reducing the error by 25% in the New York city dataset and 17% in the Chicago dataset.

3.3 Context-aware approaches to bike demand

Bike demand is influenced by multiple context factors such as weather conditions, period of day(rush hours vs non-rush), weekend vs workdays.

Regarding context data, Chen et al. [39] divided context factors into two categories, common context factors, and opportunistic contextual factors. Common context factors are a circumstance that influence bike demand daily such as weather conditions, air temperature and date and time. Opportunistic contextual factors include social events and traffic events. These events affect bike demand before and after the event.

To analyze the impact of social events, Tomaras et al. [19] used a metric, influence factor, to measure influence off an event in the nearby bike station. The influence factor is a ratio between sum the drop-off and pick-up when an event happen and drop-off and pick-up in a typical day.

The distribution of bike demands is not uniform throughout the day. Previous works such as [17] and [39] found different ways to break down a day. In [17] a day was broken down in five chunks peak morning (6:01am–10:00am); midday (10:01 – 14:00); peak afternoon (14:01 – 18:00); evening (18:01 – 23:00) and overnight (23:01 – 6:00).

In article [39] a day was broken into works days and holidays. In the workdays, a day was divided into four groups morning rush hours (07:00–11:00); day hours (11:00 – 16:00); evening rush hours (16:00–20:00); night hours (20:00 - 24:00). In holidays a day was divided into two chunks day hours (09:00 - 19:00) and night hours (19:00 - 01:00). Ye Xin li et al. [31] used two times related features to create a decision tree, the time of day and the day of the week.

In the article, Yexin li et al. [31] identified three main features regarding meteorological conditions, weather, temperature and wind speed. To reduce the weather scenarios, they categorized all weather patterns into four categories, snowy, rainy, foggy and sunny.

In the article[17], Wafic El-Assi et al. used perceived temperature instated of the original temperature. To create perceived temperature for sub-zero temperatures were used a wind chill formula. For above zero temperature were used a humidex formula.

While developing a regression model to predict the bike-sharing system flow, Tran et al. [36] used five categories of context variables, public transport variable, socio-economic variable, topographic variable, bike-sharing network variable, and leisure variable. All these variables were calculated in a buffer of 300

meters from each station except the bike-sharing network variable in each was calculated with a 3,500 meters buffer zone. Entire traffic prediction model to predict check-out of the entire city.

Kwoczek et al. [40] proposed a method to predict and visualize traffic congestions caused by planned special events. Public events are generally characterized by two waves of congestion: people arriving and leaving the event. Recognizing the difficulty of estimating the impact of these waves (the popularity of event), the authors developed a system to predict the waves using nearest neighbors from past PSEs and showed that event-sensitive predictions yield improvements.

Soltani et al. [41] conducted a web-based survey from public BSS and private BSS users (OfO and O'Bike) at Adelaide, Australia, and explored key findings to support policy makers. Context factors, including relatively low car dependency; young user profile; large share of students, visitors, and non-australian residents; safety concerns; and facility conditions were shown to highly impact demand.

Studies that have used context to enrich forecasting using neural networks, such as Thu et al. [42] propose multi-layer perceptron regressors from multi-source context data to predict bike pick-up demands from New York city BSS considering clusters of stations based on their geographical locations and transition patterns. The proposed networks combine weather factors (condition, temperature, wind speed, and visibility) and taxi trip records. Despite its relevance, temporal dependencies between observations are disregarded. Pan et al. [43] incorporate weather record data at input layer of LSTMs to improve the prediction of bike sharing demand for balancing of distribution of bikes across stations. Results evidenced improvements against context-unaware LSTMs. Recent contributions on deep learning research also show the possibility of incorporating specific forms of calendrical and spatial awareness [20, 30].

Chapter 4

Solution

In chapter 1, we explain that the existing works fail at detaching the important role of historical and prospective context sources, and the Lisbon city council intentions toward comprehending Lisbon's transportation modes. To obtain the contributions presented in section1.2, we proposed a solution with the following pipeline.

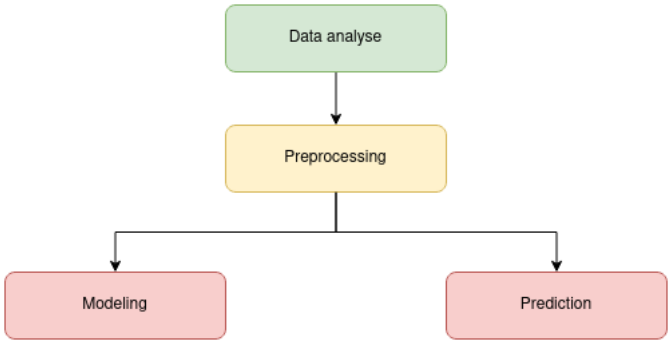


Figure 4.1: Proposed solution methodology.

This chapter is divided into multiple sections. In section 4.1 we describe the GIRA and meteorology data sources. Then we explain the data transformations needed to perform the modelling and prediction, in section 4.2. Section 4.3 we define the methodology used during the training of the models. Afterwards, we present the prediction models used as baselines, in sections 4.4 and 4.4.2.In section 4.6 we detail the different sources of context. Section 4.5 we introduce a novel deep neural network architecture capable of incorporate historical and prospective context. Section 4.8, we detail the optimization technique. Finally, section 4.9 we explain the ILU app architecture.

4.1 Description of datasets

To comprehensively assess the proposed contributions, we consider Lisbon's public BSS, termed GIRA, under the joint responsibility of EMEL and the Lisbon's City Council. The data associated with the target GIRA network contains all bike trip records from December 2018 until February 2019, with timestamps

for every change in stations' state and the corresponding load value. Structural changes to the stations' capacity associated with the BSS expansion are also considered. Given the station load data, the check-out and check-in events within the Lisbon's public BSS were inferred from the differences in the load state of each station along time. Between December of 2018 and February of 2019, the GIRA network had 75 stations dispersed around Lisbon operating between 6 am to 2 am.

The historical GIRA data, provided by CML, was set of JSON files. Each JSON file was composed of multiple JSON objects. Each JSON object had the following format:

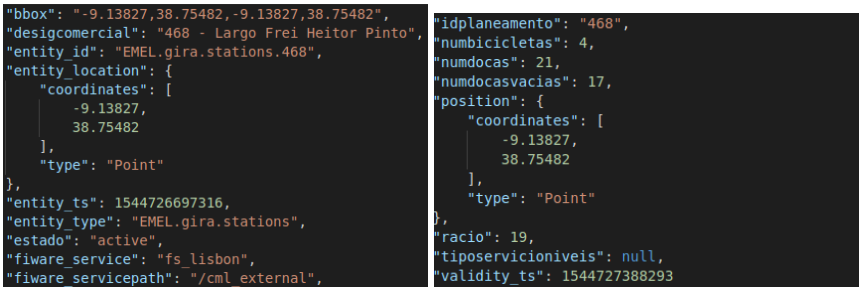


Figure 4.2: Raw gira data

where, *entity_id* is the station id, *entity_location* are the coordinates of the station, *numbicicletas* is current number of bicycles, the load value, in that station and *validity_ts* is a timestamp.

Historical and prospective *weather record data* was sourced by Instituto Português do Mar e da Atmosfera (IPMA) and Instituto Superior Técnico (IST) in a collection of JSON files.[1]

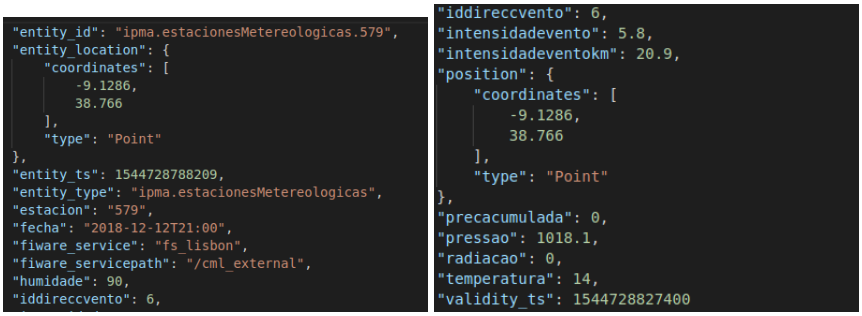


Figure 4.3: Raw gira data

Figure 4.3 is an example of a JSON object found in the dataset provided. The variable *estacion* correspond to the meteorology station id, *entity_ts* is the record timestamp, *humidade* is humidity value, *intensidadeventokm* is the wind speed in km/hour, *pressao* is pressure value and *temperatura* is temperature value in Celsius.

[1] http://www.ipma.pt/pt/index.html/

4.2 Preprocessing

In the previous section 4.1, we described the two primary sources of data, **GIRA and meteorology** datasets. In this section, we will detail the data flow from raw JSON files to the data given to the model. Firstly we describe data process from raw JSON file to the OLAP Cube. Afterwards, we explain in detail the python execution flow used to transform OLAP cube data to the data provided to the models. Finally, we define the generic training techniques.

4.2.1 Raw data transformations

ILU project is a decision support system that attempts to analyse and integrate several data sources such as Metro, Carris and combine them with additional data sources containing situational data. To integrate the multiple sources, we decide to create an OLAP database.

An OLAP database is a database optimised for querying and reporting. The OLAP data is organised hierarchy and stored in cubes. A cube is a multi-dimensional data structure that aggregates the measures by levels and hierarchies. A dimension represents an attribute in the database, and a hierarchic is a logical tree structure that organises the members of a dimension.

The ILU OLAP database has a cube per each major data source, i.e. GIRA, Metro, Carris, meteorology, espiras. Regarding dimension, ILU database has date, route and location dimensions. In the context of this thesis, we will use GIRA and meteorology cubes together with the date and location dimensions.

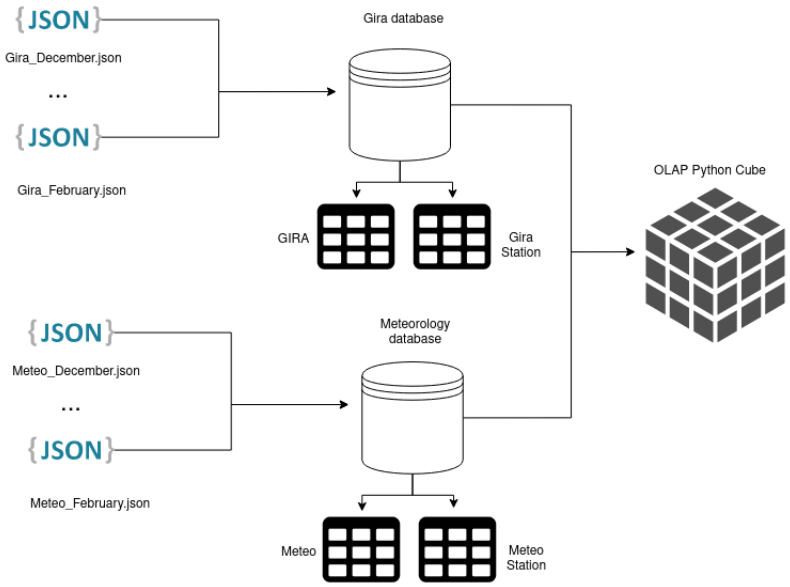


Figure 4.4: ILU database architecture

Figure 4.4 is describing the transformations from raw data, JSON file, into the OLAP database. The transformation process can be divided into two parts: create multiple relational tables using PostgreSQL(PSQL), transform PSQL tables into an OLAP cube.

4.2.2 Data augmentation

The period comprising the available GIRA records is small, with only 75 days worth of data. Deep learning methods such as LSTM or CNN, required more data to be more efficient. Data augmentation is the act of creating more training instance using the original dataset.

In the context of our problem, we used the library TSAUG method Time Wrap. The Time Wrap method augmentation changes the temporal locations. [44]. The framework TSAUG allows us to specify the number of time seriesde augmented from a single time series. We will augment the historical and forecast training instances using the same augmenter.

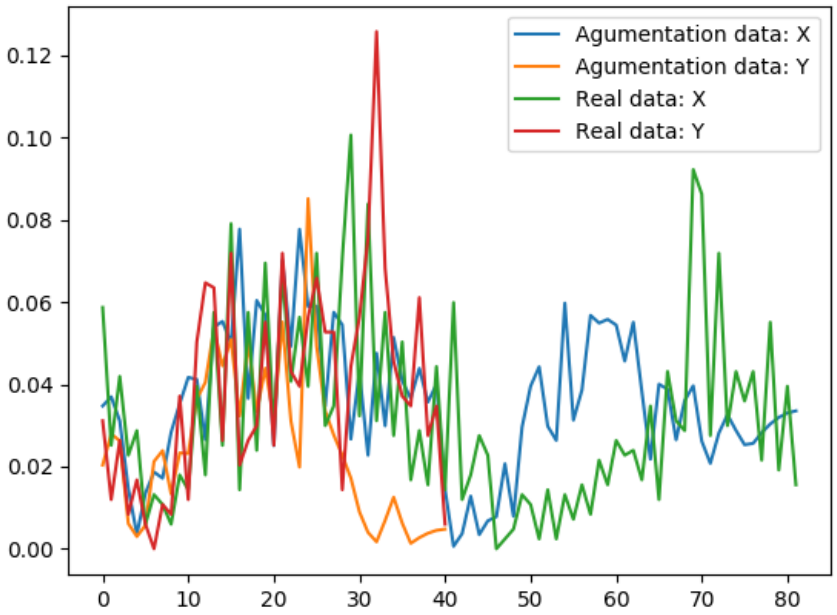


Figure 4.5: Example of a augmented time-series

4.2.3 Missing values

During the data exploration phase, we encounter portation of the data set with missing instances, such as the blue box in Figure 4.6. This intervals without data are recurrent in GIRA's dataset.

Since check-ins and check-outs trends are almost null, and these two variable have daily seasonality we decided to replace the missing data using the mean of all the data points at the same time as the missing value. For instance, the two boxes red and green in the Figure 4.6, corresponding to 5th of January and 6th of January, have a similar trajectory in the number of check-ins with low values in the begin of the day succeeded by a rising in the middle of the day followed by a drop in the end.

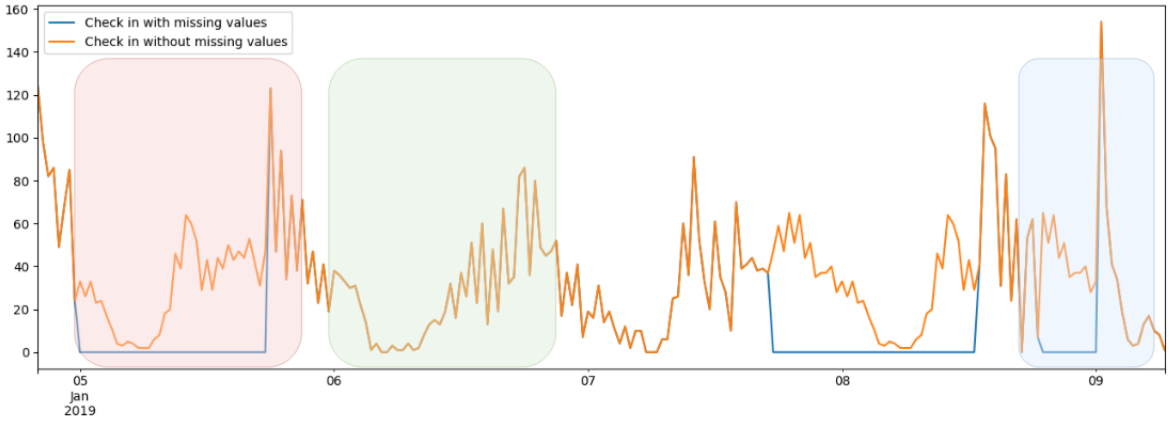


Figure 4.6: Example of a missing value

Before replacing the missing values, we need to find a period without records. We considered as a missing value any interval of 5 hours without check-ins nor check-outs. In Figure4.6, we find three intervals without record: 5-6, 7-8 and 8 and 9 of January. From this image, we inferred that the GIRA system has a recurring error and during the missing intervals the system is detecting check-ins and check-outs because after every missing period the system is registering an abnormal number of check-ins.

To solve the problem of intervals without any record, we created the following Algorithm 1. The algorithm has two main steps, find the missing intervals, 6-17 lines, and replace the missing data with the mean, 19-21 lines.

In the Algorithm 1, the input $dataset$ is a time series with the GIRA data, $window\_size$ is the number of timestamps for an interval to be conceded as a missing value, $step\_size$ is a difference in minutes between two consecutive timestamps, and $column$ is the variable to be replaced.

The GIRA bike-sharing system schedule is between 6 am to 2 am. Therefore we removed any datapoint at an interval of 2 am, and 6 am since those values can be a source of noise during the training phase.

Algorithm 1: Find and replace missing values

```
Function Find_replace_missing_values(dataset, window_size, step_size, column):
    missing_list = []
    start_interval_index = dataset.index[0]
    end_interval_index = start_interval_index + window_size
    end_index = dataset.index[-1]
    while end_interval_index <= end_index do
        window_values = dataset[start_interval_index: end_interval_index]
        if sum(winodw_values[column] == 0) then
            end_interval_index = find_non_zero_index()
            missing_list.append([start_interval_index, end_interval_index])
            start_interval_index = end_interval_index
            end_interval_index = start_interval_index + window_size
        end
        else
            start_interval_index += step_size
            end_interval_index = start_interval_index + window_size
        end

    end
    for missing_index in missing_list do
        dataset[missing_index] = replace_mean(dataset, missing_index)
    end
```

4.2.4 Generation of check-ins and check-outs

Section 4.1 introduced the variables available in the GIRA dataset. The variables check-in and check-outs are not available in a raw state, yet using the feature *numbicicletas*, the number of bikes at a specific time, we can create these two variables. The number check-ins and check-outs are the difference between two consequence time point. For example, Image that the station 106 has five bicycles in the docks at 15:33 and 15:34 the same station has six bicycles, in this scenario the number of check-ins in one.

The GIRA data has a different record of timestamps between stations. To aggregate the number of check-ins and check-outs of different stations, we need to create a time series per station with the same timestamps. We developed the following two algorithms, algorithm **A** and **B**. The algorithm **A** has the following steps:

1. For each station, generate the check-ins and check-outs features using the original time series.

2. Create the timestamps for the new time series.

3. For each new timestamp, find the closest past datapoint on the previously generated time-series and copy the check-ins and check-outs.

The algorithm **B** has the following steps:

1. Create the timestamps for the new time-series.

2. For each station, create a new time series, where the check-ins and check-outs are sums of between the consecutive timestamps.

The major difference between Algorithms **A** and **B** is the generation of the check-ins and check-outs. In Algorithm **B** the check-ins and check-outs values per timestamps are the sums all the check-ins and check-outs between the timestamps.

4.2.5 Feature scaling

To achieve better results, machine learning methods such as artificial neural networks or KNN, need to have the features in the same range. Some of the feature scaling methods are min-max normalization and Standardization.

The min-max normalization is a process of rescaling the original feature range for a range between 0 and 1. The formula for min-max method is given by:

$$\mathbf{x'} = \frac{\mathbf{x} - min(\mathbf{x})}{max(\mathbf{x}) - min(\mathbf{x})} \tag{4.1}$$

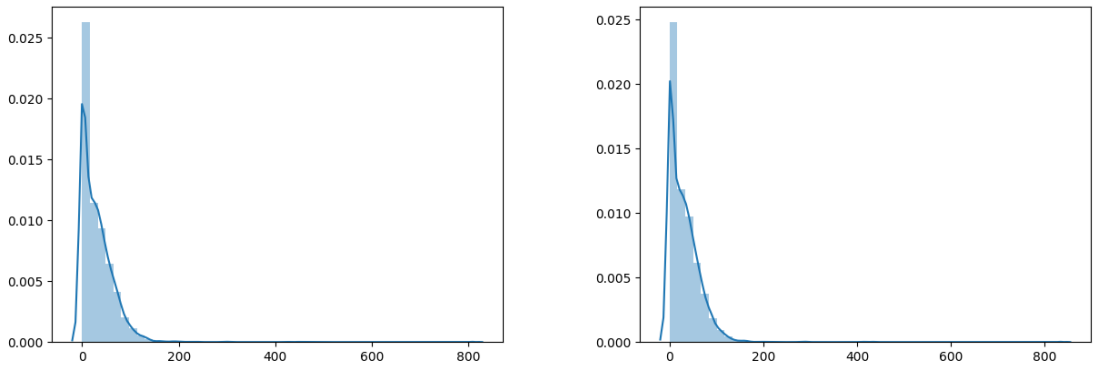
where x is the original feature vector and $min(x)$ and $max(x)$ are features minimum and maximum values, correspondingly.

Standardizing a dataset involves **rescaling the distribution of values** so that the **mean of observed values is 0** and the **standard deviation is 1**. The standardizing formula is given by:

$$\mathbf{x'} = \frac{\mathbf{x} - \bar{\mathbf{x}}}{\sigma} \tag{4.2}$$

Where x' is the feature vector, $\bar{x}$ is the feature means, and σ is the feature standard deviation.

Standardization expects that data suits a Gaussian distribution with a well behaved mean and standard deviation. Standardize can be applied to time-series data that does not match Gaussian distribution, but results may not be reliable.



(a) Check-in distribution

(b) Check-out distribution

Figure 4.7: Check-in and Check-out distribution along all the GIRA stations

The image 4.7 suggests that the check-in and check-out distribution are not normal distributions; therefore standardizing the dataset is hypothesized to be more appropriate.

4.3 Training methodology

To guarantee the proper learning of the proposed neural networks, considerations associated with both the segmentation and the learning setting should be placed. Given a specific horizon of prediction h and its time resolution Δ, the series should be segmented in a set of instances, each instance being a pair (*input*, *output*) where the *input* is a series with at least twice the length of the prediction horizon (by default $7 \times h$), the *output* is the subsequent h-length series, and both series have the same time resolution, Δ. Segmentation is further characterized by the presence of a sliding window to compose the

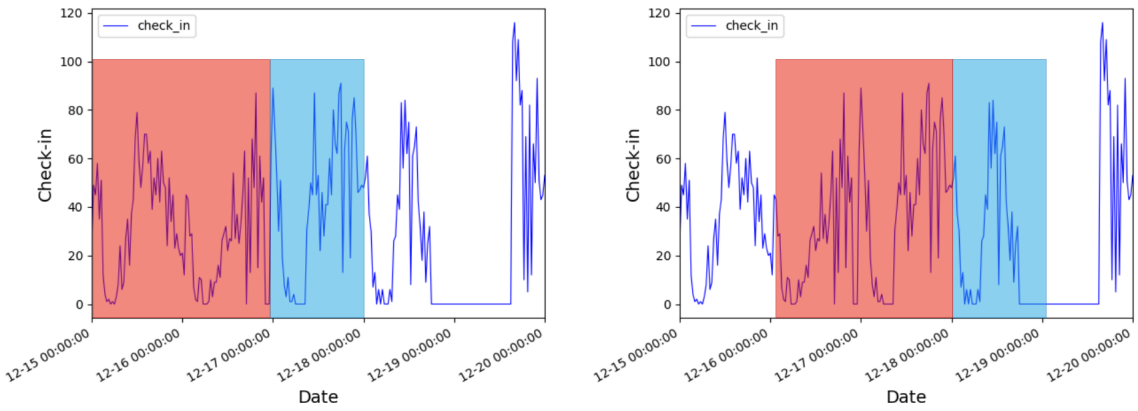
data instances, algorithm 2 . By default, the sliding window corresponds to a single day (24 hours).

Algorithm 2: Create input output pair algorithm

```
1  Function create_pair_input_output(dataset, input_size, output_size, step_size):
2      list_input_output = []
3      window_size = input_size + output_size
4      end_input_ouput_index = window_size - 1
5      len_dataset = len(dataset)
6      while end_input_ouput_index <= len_dataset do
7          begin_input_output_index = end_input_ouput_index - window_size
8          input = dataset[begin_input_output_index :begin_input_output_index + input_size]
9          output = dataset[begin_input_output_index + input_size : begin_input_output_index +
             input_size + output_size]
10         list_subdataset.append((input, output))
11         end_input_ouput_index += step_size
12     end
13     return list_input_output
14 End Function
```

where $dataset$ is a subdataset generated with algorithm 3, $input\_size$ is the number of timepoint used as hisotrical data, $output\_size$ is the of the output, and the $step\_size$ is the lag value between pairs input-output. The output $list\_input\_output$ is a list containing all the possible pairs input-output for that $dataset$.

In the figure 4.8 is a graphical demonstration of the algorithm 2 with $input\_size$ of two days, $output\_size$ and $step\_size$ of one day. The red boxes are the inputs values, and the blue boxes are the output values.



(a) Window between 15 and 18 of December

(b) Window between 16 and 18 of December

Figure 4.8: Sliding window example

The above principles are applied to segment series in the context of a single dataset. For a more robust validation of predictors, cross-validation is suggested. Particular attention should be paid to guarantee the soundness of the cross-validation setting given the inherent time frame of the series data. Algorithm 3 shows how multiple datasets are produced. In this context, *series* is the inputted (univariate

or multivariate) time series with a time frame Δ given by the user; *window_size* is the length in days of each dataset, and *step_size* is the lag value between datasets in days. Figure 4.9 provides an example of the algorithm 3, where *window_size* = $T - 2$, *step_size* = 1 and *datasets* is the resulting set of three datasets.

Algorithm 3: Split dataset algorithm

```
1  Function create_subdatasets(dataset, window_size, step_size):
2      datasets = {}
3      end_dataset_index = window_size - 1
4      while end_subdataset_index ≤ |series| do
5          begin_dataset_index = end_dataset_index - window_size
6          dataset = series[begin_dataset_index:end_dataset_index]
7          datasets = datasets ∪ {dataset}
8          end_dataset_index += step_size
9      end
10     return datasets
11 End Function
```

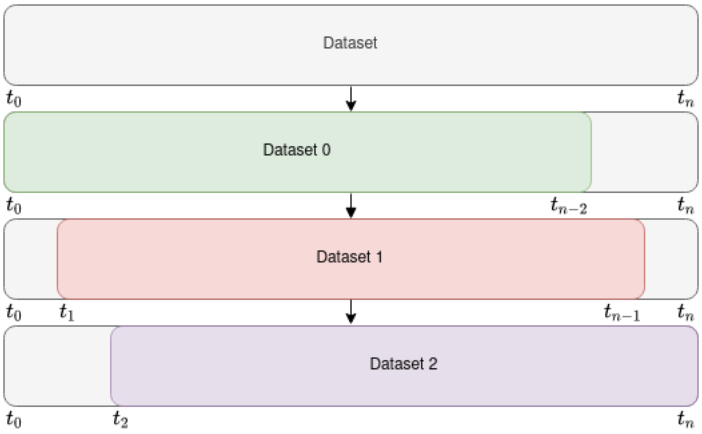


Figure 4.9: Example of algorithm 3 with $step\_size$ of 1

After applying Algorithm 3, we split each sub-dataset into train, validation and test data, guaranteeing that training data instances always precede validation and testing data instances in order to better mimic real-time testing scenarios.

4.4 Distance-based approaches

4.4.1 Barycenter

In this section, we explore the definition of barycenter, euclidean barycenter and DBA. Throughout this section, let $A =< a_1 \cdots a_T >$ an $B =< b_1 \cdots b_T >$ be two sequences and $\mathbb{S} =< \mathbb{S}_1 \cdots \mathbb{S}_\mathbb{N} >$ be a dataset with sequences. A barycenter is a time series A which minimizes the sum of squared distances to the time series of a given dataset $\mathbb{S}$:

$$\min \sum_i d(A, \mathbb{S}_i)^2 \tag{4.3}$$

where d is the distance function.

A barycenter is a family of methods, we will focus on **euclidean barycenter** and **DTW barycenter averaging (DBA)** [45].

A **euclidean barycenter** follows the equation 4.5 where d is the euclidean distance function. Given the sequences A and B, the euclidean distance is given by:

$$d(A,B) = \sqrt{\sum_i (a_i, b_i)^2} \tag{4.4}$$

Before explaining the details of **DTW barycenter averaging**, we recall the definition of DTW. **Dynamic time wrapping** finds the finds the optimal alignment between two sequences of numerical values, and captures flexible similarities by aligning the coordinates inside both sequences.The DTW value can be computed by:

$$D(A_i, B_j) = \delta(a_i, b_j) + \min \left\{ \begin{array}{l} D(A_{i-1}, B_{j-1}) \\ D(A_i, B_{j-1}) \\ D(A_{i-1}, B_{j-1}) \end{array} \right\} \tag{4.5}$$

The DTW value of the sequences A adn B is given by $D(A_T, B_T)$.

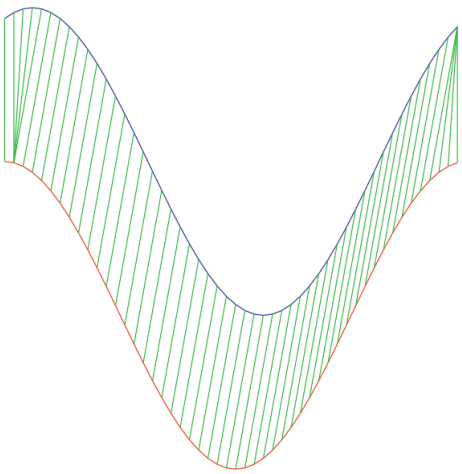


Figure 4.10: Example of DTW

An example DTW-alignment of two sequences can be found in 4.11

The dynamic time wrapping barycenter averaging (DBA) is a iterative process that refine an initial average sequence $C =< C_1 \cdots C_T >$, to minimize the DTW distance to sequences of the dataset, $\mathbb{S}$. Each iteration has two steps:

1. Compute DTW between each sequence in $\mathbb{S}$ and $\mathbb{C}$, in order to find associations between coordinates of the average sequence and coordinates of the set of sequences.

2. Update eeach coordinate of $\mathbb{C}$ as barycenter of coordinates associated to it during the first step.

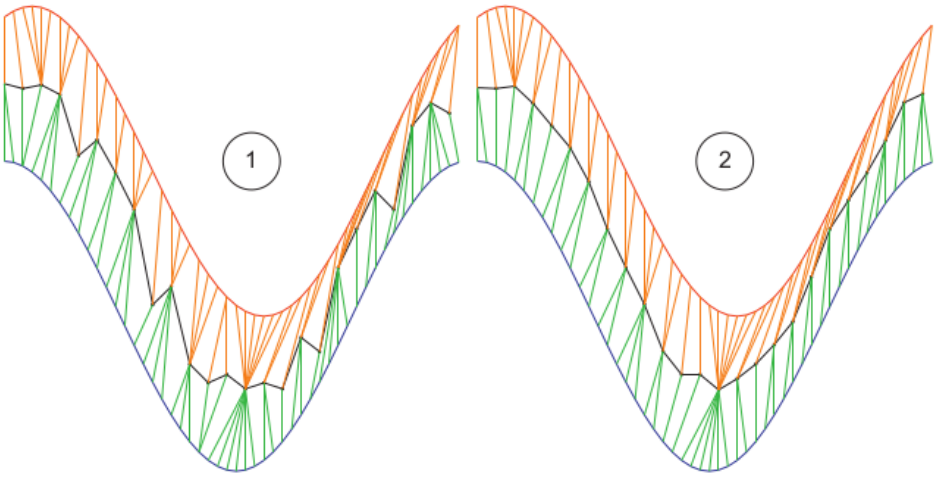


Figure 4.11: Example of DBA

In figure 4.11 represent two iteration of the DBA algorithm. In this case, $\mathbb{S} =< \mathbb{S}_1, \mathbb{S}_2 >$ where the black line is the average sequence, $\mathbb{C}$, and the orange and blue correspond to $\mathbb{S}_1, \mathbb{S}_2$.

4.4.2 Lazy approach

Before exploring the multiple lazy methods, we need to define what is a lazy method. A lazy method compares the stored training data with the test data and predicts a test instance using the most similar training instances. This category of methods is called lazy because they do not try to learn a model.

KNN

K-nearest neighbours (KNN) is a supervised machine learning algorithm used to solve classification and regression problems. The KNN method starts by finding the k closet training instances to the test data point using a distance function. For classification tasks, the KNN determinate the class of the test instance using majority voting principle. For regression problems, the KNN algorithm predicts using the mean of the k closest training instance.

The GIRA data is a time series, i.e. a sequence of numbers ordered by time. The KNN is a supervised method, thus requiring input variables, X, and output variables, Y. To use the KNN with GIRA data we had to create the input and output variables.

We need to define what is input and output variables in the GIRA context. The input variable is a time series with **historical data** required to predict the output. The output variable is a time series with the **desired forecast** values. In section 4.3 explains in detail the algorithm and methods to create the historical and desired forecast variable.

Classical methods

In chapters 2.3 and 4.4.2 , we explain some classical models such as ARMA, ARIMA and Holts-winters. From the multiple models describe in chapter 4.4.2, we settle on using **Holts-winter** since, in the literature, this model as an acceptable performance for a base model. In this chapter, we recall the definition of the Holts-Winters model and define the methodology applied.

4.5 Neural network approach

In recent years, the literature has proved that recurrent neural network, specially Long Short Term Memory networks (LSTM) neural network, are effective at doing forecast tasks.

In this subsection, we summarize the definition of LSTM, explain the training technique and the hyperparameters optimization algorithm.

4.5.1 LSTM

As introduced, Long Short Term Memory (LSTM) networks are recurrent that can learn long-term dependencies. An LSTM netwrok has two major elements hidden state h and cell state c ,which plays the role of memory. The transformation from c_{t-1} to c_t has four: **forget step**, **candidate state step**, **state generation step**, and **output step**.

Firstly, the **forget step** select how much information from the old state, c_{t-1}, will be thrown away. Afterwords, **candidate state step** which decides what new information will be added to the cell state. The next step, **state generation**, combines the old state with the new information creating a new cell state, c_t. Finally, the **output step** generates an output, h_t by filtering the new cell state.

In Image 2.4 is illustration an LSTM architecture where the boxes represent each step, i.e, yellow box is forget step, the orange box is candidate state step, the blue box is state generation step and brown box in the output step.

We propose a new recurrent neural network layering able to incorporate both historical and prospective sources of context to guide forecasting tasks. The proposed architecture is a sequential composition of two components, C_1 and C_2. C_1 is a LSTM that receives context-enriched multivariate inputs and returns the forecasted series as the output. C_2 is a LSTM (or a gated recurrent unit) that receives as input the forecasted series from C_1 and prospective sources of context along the horizon of prediction and returns the true forecasting. Section 4.6 describe how historical and prospective sources can be fed into the proposed architecture, and further discuss their significance.

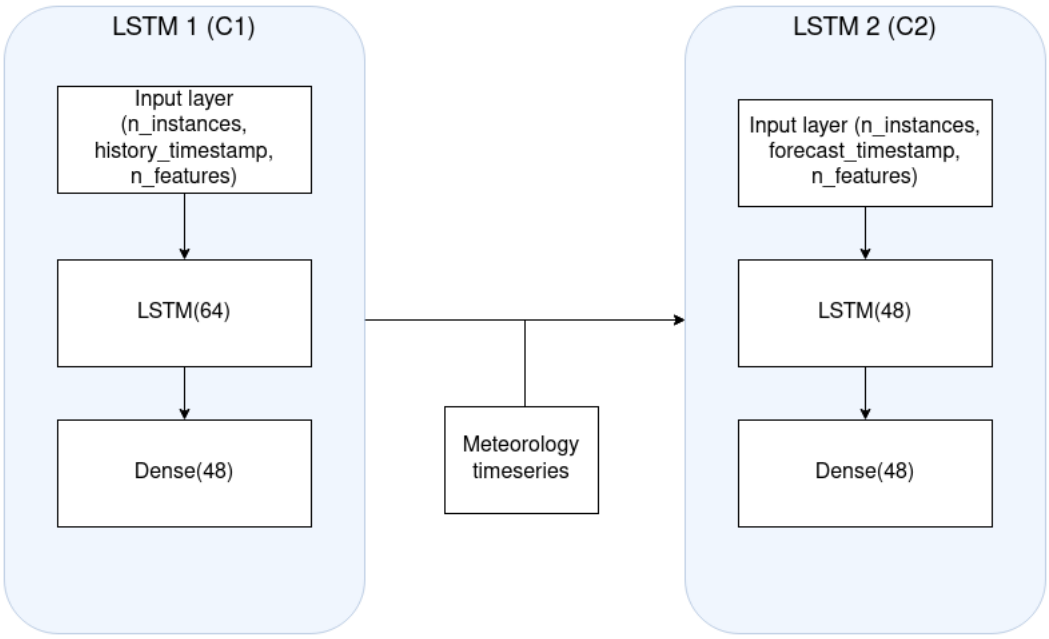


Figure 4.12: Neural network architecture

4.6 Historical context incorporation

A time series is categorized by the number of variables per observation, i.e., univariate or multivariate. We define context enrichment as being the prediction of a multivariate time series where one or more input variable are context variables.

To incorporate historical sources of context data, we take advantage of the fact that LSTMs are inherently prepared to learn mapping functions from multivariate time series into the target univariate time series (demand variable along a prediction horizon). In this way, an arbitrarily-high multiplicity of context variables can be combined at the input layer of the C_1 component to guide the learning task. Sections 4.6.3 to 5.5.3 offer important masking principles on how to compose the multivariate time series from difference context sources. For the purpose of illustrating the principles introduced along these sections, consider that the task at hands is to learn a predictive model that forecasts the number of hourly bicycle check-outs for the upcoming days at a given station using a two-day historical data with an half-hour aggregation,

$$\begin{aligned}\mathbf{x}_{\text{example}} &= (\mathbf{x}_{0h:1/1/2018}, \mathbf{x}_{0:30h:1/1/2018}, .., \mathbf{x}_{23:30h:1/2/2019}),\\ &= ((9), (6), .., (10)),\end{aligned}$$

a series to be further segmented for the purpose of learning.

Under masking principles, sources of historical context are hypothesized to guide the forecasting task in two major ways. First, by looking to the past and modeling how a certain context variable affects bike demand at a given station offers the possibility of removing context-dependent factors. Considering

wind intensity as an illustrative context variable, deviations from regular wind ranges can be considered by the LSTM unit to better model the co-observed demand.

Second, some sources of historical context can be correlated with future context and/or demand. Therefore, this relation can be directly learned by the LSTM unit to shape short-term forecasts. For instance, strong wind intensity at a given hour may decrease the observed demand with effect lasting up to three hours.

4.6.1 Station context

In addition to the target variable, complementary demand variables can be inputted to guide the prediction. For instance, the analysis of check-ins can be complemented with the volume of check-outs to account for correlated deviations.

Another example is to consider the load state of the station as we know that states of high-load (low-load) can impact check-outs (check-ins). Illustrating, $\mathbf{x}_{\text{example}}$=((*check-in*=9,*check-out*=6,*load*=0.9), ..,(*check-in*=10,*check-out*=12,*load*=0.3)), is an augmented series with a multivariate order of 3.

4.6.2 Situational context

Similarly to the masks produced from year and academic calendars, situational masks mark periods where events of interest may impact the demand observed at a given station. The major difference between these sources of context is the fact that situational context is usually circumscribed to a specific geographical area, therefore only impacting a subset of stations from a given BSS. Given a station, the situational variables produced using these masking principles generally correspond to public events (concerts, conventions or sport events) that occur within a maximum distance from the target station.

Situated events can produce varying levels of impact on the demand at a given station. For instance, stations nearby a stadium and a small concert hall should be able to differentiate between sport events and concerts. In this context, these masks can explicitly capture the magnitude of an event, as well as variations on that same magnitude throughout the period where the effects of the event lasts. Illustrating, $(0,0,1,3,3,1,1,3,3,1,0...)$ series may correspond to a mask modeling the magnitude of a gathering that can be placed as a complementary inputted series to guide the learning.

4.6.3 Meteorological context

Meteorological variables, including wind intensity and precipitation, can be as well inputted to guide the learning. The closest meteorological station can be selected or k-nearest stations to compute the weather series (under distance-weighting schema). Additional weather variables can be produced from the raw weather records produced by meteorological stations, including the perceived temperature or integrative scores.

A high number of weather variables leads to high a multivariate order of the inputted, which can hamper the learning if not enough historical observations are available. In this context, variable selection can be considered to guarantee that only the most informative weather variables are inputted.

4.6.4 Calendrical context

To supplement the GIRA's time series with calendrical data, we develop two different approaches: split the dataset into workdays-weekends and we suggested masks.

Four masks are suggested:

- a *day mask* can be added to the time series with information pertaining to the week-day of each observation,

$$\mathbf{x}_{\text{example}} = ((9, monday), (6, monday), .., (10, tuesday)).$$

  In this way, LSTMs are inputted with information that helps them internally separating neural pathways in accordance with the week-day. This is a simplistic yet essential guiding information whenever segmentation creates instances starting at different timings.

  In addition, this masking principle can act as way of augmenting data as it allows for the combined inclusion of weekends and weekday periods in the forecasting task, instead of learning separate forecasting models.

  If weekdays show similar patterns of demand, simpler daily masks can be applied using mappings of lower cardinality, such as {weekday, saturday, sunday};

- a *holiday mask* can be added as a separated binary variable in order to indicate whether a given observation was produced or not within a holiday;

- an *hour mask* can be added as an additional variable to guide the learning task whenever the segmentation step does not guarantee that every data instance starts at the same time step. As a result, daily seasonalities can be implicitly captured within the LSTM as a non-linear function of the hour within the day. If the time granularity of the demand series is finer (coarser) than hour, a finer (coarser) mask can be produced. For instance, a mapping {dawn, morning, afternoon, evening} is adequate for temporally misaligned data instances and time windows with up to 6 hours;

- other relevant masks, include *academic period masks* or *festivity period masks* that may not necessarily coincide with the aforementioned holiday mask. In the context of our work, the influence of these masks were found to be essential.

4.6.5 Spatial context

In the context of station-wise dependencies, we propose a geographical mask, referred as the nearby mask. The nearby mask is the occupation ratio of the neighbouring stations. The algorithm to create the nearby variables, Algorithm 4, receives the target station or cluster of stations under analysis, SS, a distance radius, R, list of all stations, S, and time series TS. Algorithm 4 has three-steps: 1) computes the centroid of the selected stations, line 3; 2) finds all the stations inside the R radius using the previous centroid as the centre (lines 4–8), and updates the time series with the occupation ratio of the nearby stations (lines 10–12).

Under the nearby mask, the target LSTMs are now able to consider the impact that stations with high- and low-load have on the demand of nearby stations. Here, the occupation ratio is proposed in contrast with full-and-empty station states to allow for more flexible learning (nearly full stations may be full in the upcoming time steps). Still, alternative encodings can be proposed by customizing line 11 of Algorithm 4.

Algorithm 4: Create nearby mask algorithm

1 **Function** `create_nearby_mask` (SS, S, R, TS) **:**
2 nearby_station = []
3 lat_centroid, long_centroid = calculate_centroid(SS)
4 **foreach** *station* $s_i \in S$ **do**
5 distance = distance_two_points(lat_centroid, long_centroid, s_i.lati, s_i.long)
6 **if** *distance* $<$ *radius* **then**
7 nearby_station.append(s_i)
8 **end**
9 **end**
10 **foreach** *station* $s_i \in$ *nearby_station* **do**
11 TS = update_ts(TS, $\frac{s_i.num\_bicicletas}{s_i.location\_capacity}$)
12 **end**
13 **End Function**

4.7 Prospective context incorporation

Prospective sources of context, including weather forecasts or planned events, may be available along the horizon of prediction. In this context, their use can further guide the learning. Recall that the proposed architecture is a composition of two components, C_1 and C_2. C_1 is a LSTM that receives multivariate inputs (in accordance with the principles introduced along sections 4.6.1 to 5.5.3) and returns the forecasted series as the output. C_2 is a recurrent neural component that receives the forecasted series from C_1 and the prospective sources of context along the horizon of prediction as inputs, and returns the true forecasting. In this context, C_2 can be though as a context-aware denoiser or time-dependent regularizer of the forecasts. Empirical analysis show optimal performance when C_2 is a LSTM unit. Nevertheless, a gated recurrent unit (GRU) provides a competitive alternative given the properties of this step.

Under the proposed architecture, masking principles – similarly to the ones introduced along sections 4.6.1 to 5.5.3 – can be considered to compose the multivariate time series to be inputted to the C_2 component. In this context:

- calendrical masks can be produced to regularize the forecasts whenever the prediction horizon contains holidays or any other meaningful season (such as academic break). In addition, and more generally, time and day masks can be as well inputted for a calendric-guided revision of

forecasts;

- situational masks can be produced from planned events available from cultural and sport agendas, as well as semi-structured repositories containing information on the usage of public spaces. See section 4.6.2 for principles on duration- and intensity-sensitive encodings for situational masks;
- meteorological masks can be produced from weather forecasts. As weather forecasts are typically produced under a coarse-grained time granularity, upsampling may be necessary to guarantee the weather and demand series share the same granularity. See section 4.6.3 for principles on station and variable selection;
- spatial masks can be derived from forecasts of nearby stations' load (see principles in section 5.5.3) when load forecasts are able to satisfy upper bounds on the observed error.

By incorporating prospective sources of context, the C_1-forecasted demand series can now be further subjected to context-dependent corrections, guiding the forecasting task. For instance, periods with forecasts of high wind intensity can suffer an adjustment on the C_1-based demand expectations by capturing relationships between the multivariate input series and the true forecast.

4.8 Optimization

Most of the modeling methods have two types of parameters: **model hyperparameters** and **model parameters**. **Model parameters** are values estimated during the model training, such as coefficients in linear regression or logistic regression. **Model hyperparameters** are configurations chosen before the model training. Therefore the values cannot be learned from the training data.

Finding the hyperparameters value can be view as an **optimization problem**. The optimization problem aims to achieve the right hyperparamets values that lead to the lowest error value. This problem can be solve with multiple approaches such as manual search, random search, grid search and others.

In this section, we explain the manual search, grid search and bayesian optimization.

Manual search

Manual search is one of the most straightforward approaches to hyperparameters tuning. This manual approach is an iterative process, where the user chooses the model hyperparameters based on experience and judgment. The process stops when the performance metric is satisfactory. A significant advantage of manual tunning is that the user learns the behavior of the hyperparameters. A drawback of this process is that the user can stop the approach before reaching the best model performance.

Grid search

Grid search is a approach where for every possible set of hyperparameters candidates a model is trained. The grid search returns the set of hyperparameters associate with the best performance model.

The grid method ensures that the best hyperparameters are selected. For some problems , the grid search is impossible to use since the method has a large temporal complexity.

Bayesian Optimization

Bayesian optimization is a machine learning approach to solve the problem of maximizing an objective function. Typically the objective function is continuous and expensive to evaluate. The objective function is a black box because we can only query the function.

Bayesian optimization has two major components: a Bayesian statistical model for modelling the objective function and acquisition function for deciding where to sample next. The Bayesian process has the following steps:

1. Build a surrogate model using the available data about the objective function, using a Gaussian process
2. Obtain the inputs of the objective function using the acquisition function
3. Query the objective function
4. Update the model of the objective function
5. Repeat the steps 1-3 until max interaction or time are reached

4.8.1 Optimization in the GIRA context

In the previous sections 4.4.2 and 4.5 we explain the methods for modelling and predict the bike demand however the optimization approach used is missing.

For the **distance** and **lazy** approaches, we settled on using a manual search since these the methods have few hyperparameters. However for the **neural network** approach we used Baysian optimization , GPyOpt[46] framework. The GPyOpt is an open-source framework that allows using of Gaussian process optimization.

Method	Hyperparameter	Type	Domain
KNN	K	Continuous	1-20
	Distance	Discrete	DTW, euclidean
Barycenter	Type	Discrete	DTW, euclideanm, DBA
LSTM	Learning rate	Continous	$10^{-6}, 10^{-3}$
	Regularization	continuous	$0.001, 0.1$
	Batch size	Discrete	$(1, 2, 4, 6, 8, 12)$
	Regularization type	Discrete	$L1, L2, L1_L2$

Table 4.1: Hyperparameters

Manual search optimization can provide insides about the hyperparameters functionality in a machine learning model that Bayesian optimization can not offer. To infer the hyperparameters impact on the performance, we explored the space of the parameters shown in Table 4.1.

4.9 Gira ILU app architecture

The ILU app has various UI components one per transportation method, i.e., Metro UI, Carris UI, GIRA UI, and others. Figure 4.13 represents the ILU app section responsible for predicting and modeling the bike-sharing system. The system can be split into four logical components, UI component, controller component, data component, and model component.

UI component has two major roles: display the training and model options, second receive and transform the user request into a python object, dictionary object, with training and model settings. The UI component allows the user to select a model, model hyper-parameters, and training settings. In chapter 4.3, we explain in detail training methodology.

The controller component has as input the model and training settings, and the controller is the intermediary between UI, Data, and Model components. The controller component is responsible for gathering data, sending data to the model component, and sending the predicted time series to the UI component. Training settings are composed of two categories: data category and training category. Data category has information specified by the user about temporal data boundaries, i.e., begin and end dates, and the stations' ids. On the other hand, the training category has the training details, such as window size, instance size, and moving window step.

The data component is the interface between the OLAP cube and the controller component. The data component has two primary responsibilities: fetching the data from the OLAP cube using the data details and send it to the controller component.

The model component is composed by multiple forecasting models, such as LSTM, Hotls-Winter, Barycenter, and KNN. In chapters 4.4.2, 4.4.2 and 4.4, we explain the different forecast techniques used.

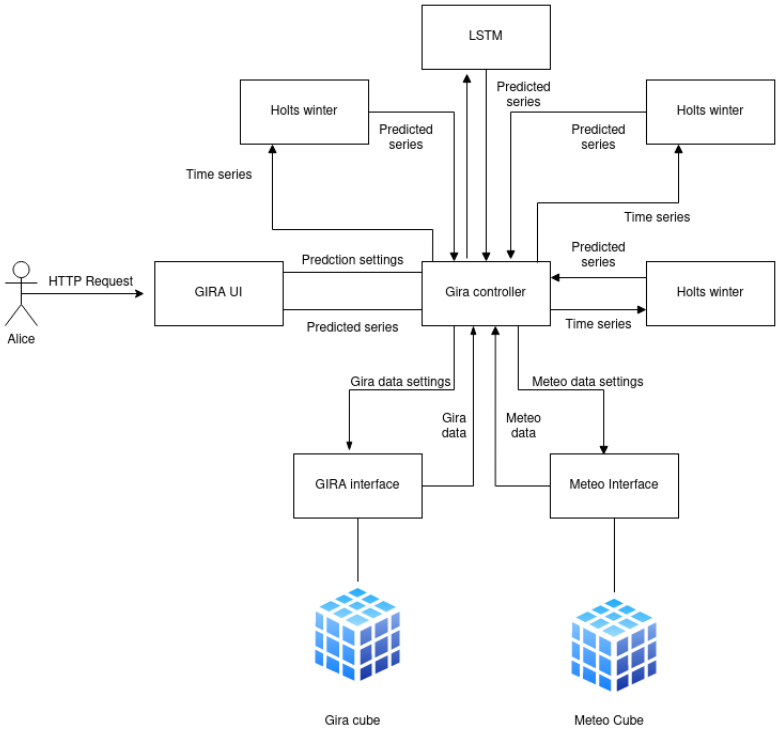


Figure 4.13: ILU: Use case diagram

Chapter 5

Results

Results are organized into three significant steps. First, we introduced background considerations on the: i) target public BSS (GIRA) and considered sources of context, ii) assessment setting and iii) general exploratory statistics on GIRA's bike demand and validation methods. Second, we provide a thorough comparison of state-of-the-art forecasters against the proposed context-aware predictors. Third, we offer greater detail on the performance of the proposed context-aware predictors, describing the impact that different sources of context have on the forecasting residues.

The proposed context-aware predictors were implemented using tensorflow, python. The remaining state-of-the-art predictors were implemented using facilities from statsmodels, tslearn (barycenter calculus) and scikit-learn packages in python. The parameters of the state-of-ther-art predictors (such as smoothing factors in Holt-Winters or k in k-nearest neighbor regressors) were subjected to Bayesian optimization.

5.1 Experimental setting

To comprehensively assess the proposed contributions, we consider Lisbon's public BSS, termed GIRA, under EMEL's joint responsibility and the Lisbon's City Council. The data associated with the target GIRA network contains all bike trip records from December 2018 until April 2019, with timestamps for every change in stations' state and the corresponding load value. Structural changes to the stations' capacity associated with the BSS expansion are also considered. Given the station load data, the check-out and check-in events within Lisbon's public BSS were inferred from the differences in each station's load state along time. Consult 4.2.4 for more detail on generation the check-ins and check-outs.

Historical and prospective *weather record data* was sourced by Instituto Português do Mar e da Atmosfera (IPMA) and Instituto Superior Técnico (IST). Humidity, wind speed (km/hour), pressure, precipitation, and temperature (Celsius) weather variables were sourced. Four meteorology stations were available. The weather station closer to a GIRA station is selected for providing weather data. When analysing clusters of stations, the centroid of each cluster is computed to identify the closest meteorology station.

Bike demand analysis can be pursued at different spatial granularities and user types following the targeted aim, such as establishing balancing initiatives (finer granularity) or general planning decisions (coarser granularity). The GIRA's users can be breach into three types: students, workers, and tourists. To this end, we predict bike demand at three distinct levels: i) station level; ii) cluster of stations; and iii) BSS level (all stations).

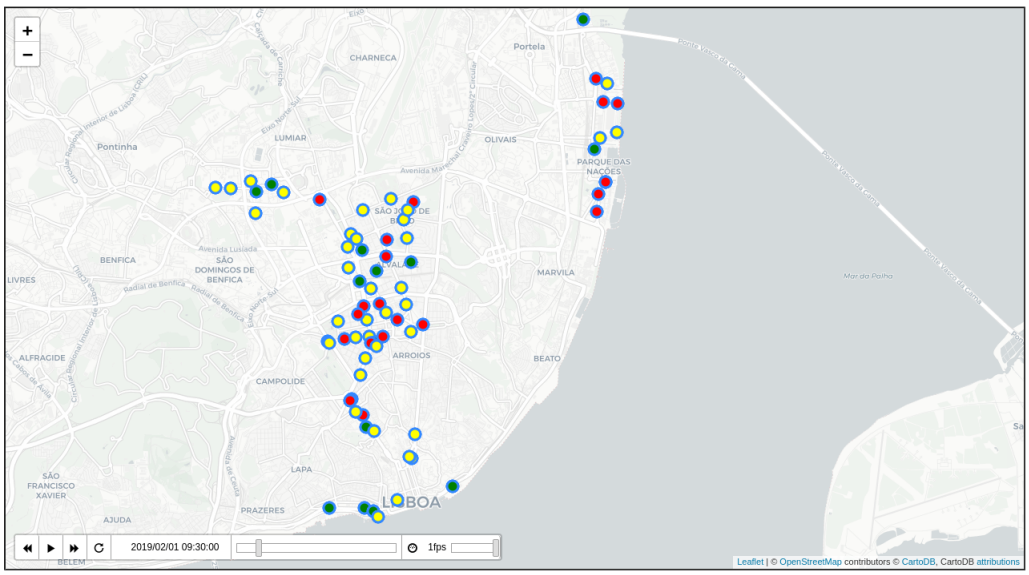
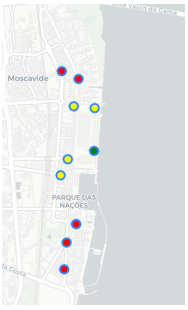


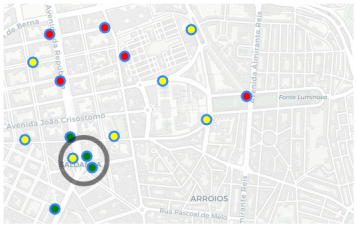
Figure 5.1: GIRA stations (Screenshot from ILU app)

Figure 5.1 is a screenshot from interactive visualization in the ILU app, where it is possible to monitor the number of bicycles per station at a given time and highlight a specific station or cluster of stations.

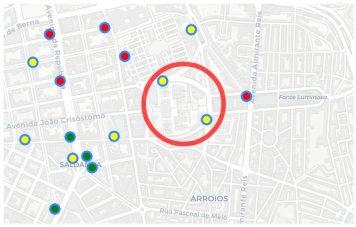
To help the city Council understanding demand, we identify three clusters of interest: IST cluster, Saldanha cluster and Oriente cluster.



(a) Oriente cluster



(b) Saldanha cluster



(c) Ist cluster

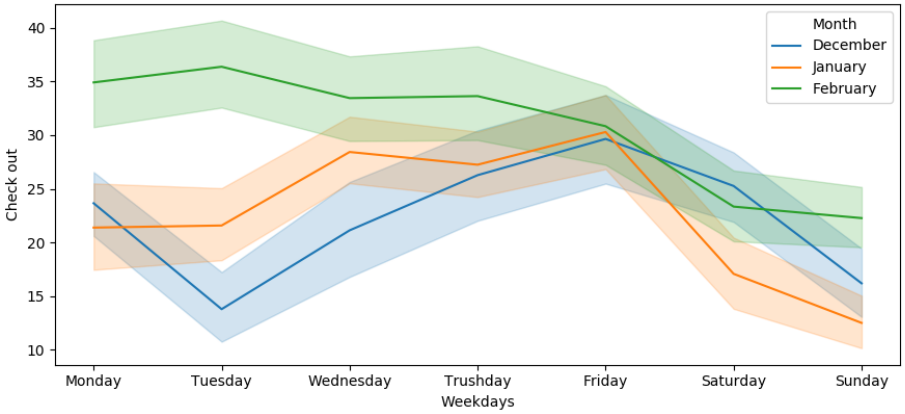
Figure 5.2: Clusters (Screenshot from ILU app)

Each of these clusters yields unique aspects of interest. The IST cluster has a high share of students since its stations are close to the campus of Instituto Superior Técnico's faculty. Saldanha cluster has a high share of workers since its stations are located within a well-known business district. Finally, Oriente cluster has a high share of tourists since its stations are located near Parque das Nações, a place with diverse cultural, expo, and leisure attractions.

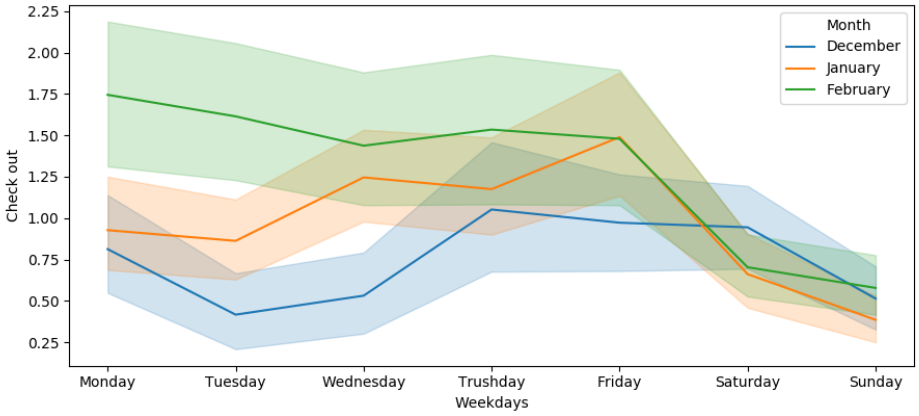
The Oriente cluster is the larger cluster, encompassing six docking stations and a total capacity of 144 bicycles. The Saldanha cluster has three stations with a total capacity of 44 bicycles. Even though IST cluster has only two stations, it is capable of holding 51 bicycles.

5.2 Exploratory data analysis

Figures 5.4 and 5.3 provide a high-level view on GIRA's bike demand patterns at different granularities. Those figures capture the variability of demand along the targeted period, where the bounds (standard deviation) confirm that bike-sharing demand is susceptible to diverse idiosyncrasies causing significant variability.

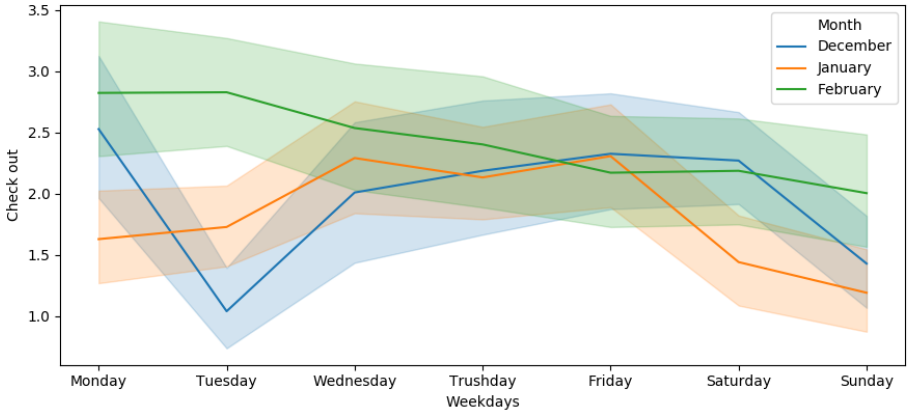


(a) Network (all stations)

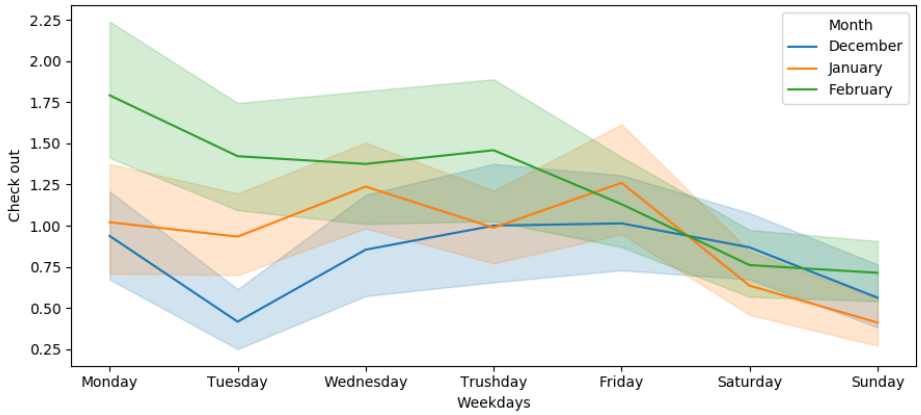


(b) Ist cluster

Figure 5.3: Weekly volume and variation of check-outs per cluster and the network (December 2018 to February 2019

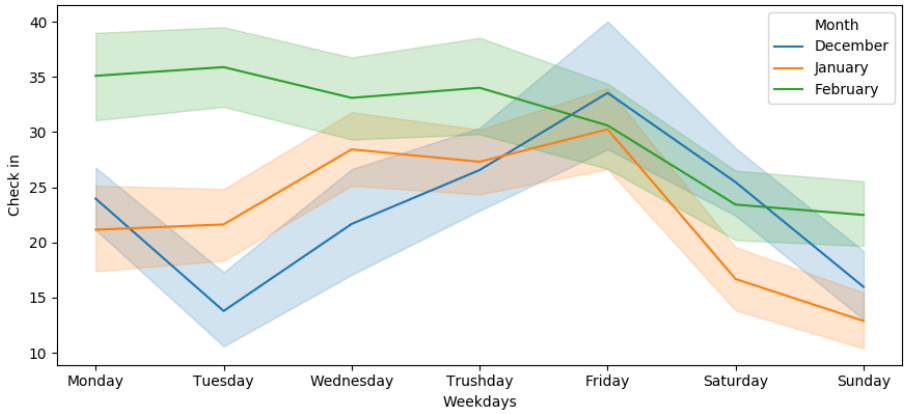


(c) Oriente cluster

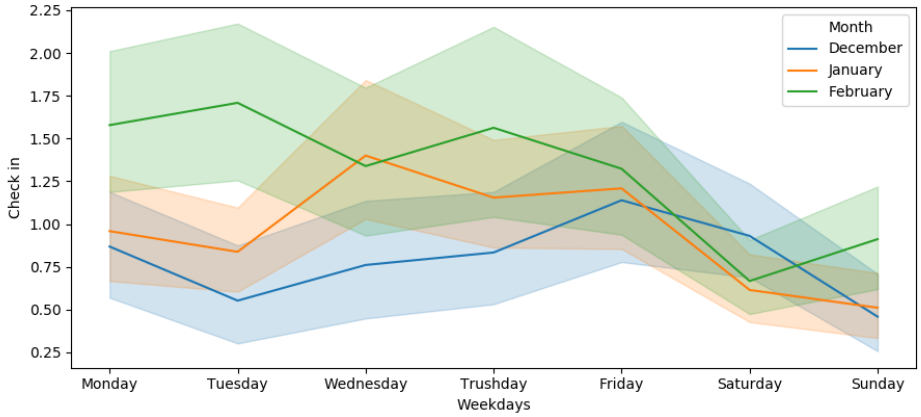


(d) Saldanha cluster

Figure 5.3: Weekly volume and variation of check-outs per cluster and the network (December 2018 to February 2019

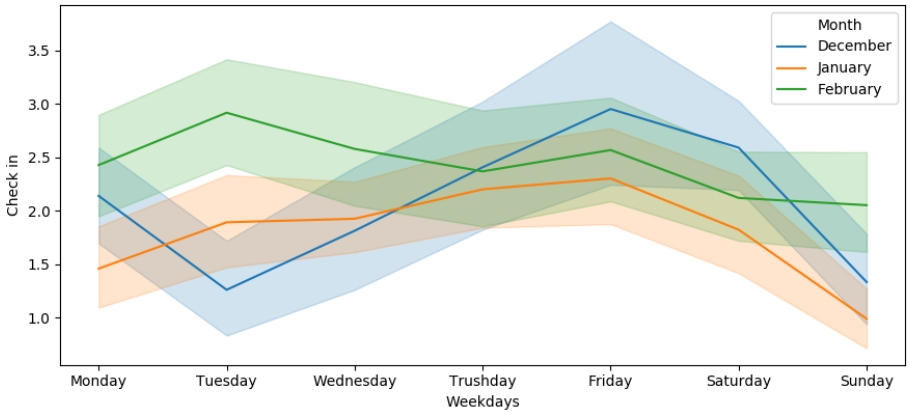


(a) Network(all stations)

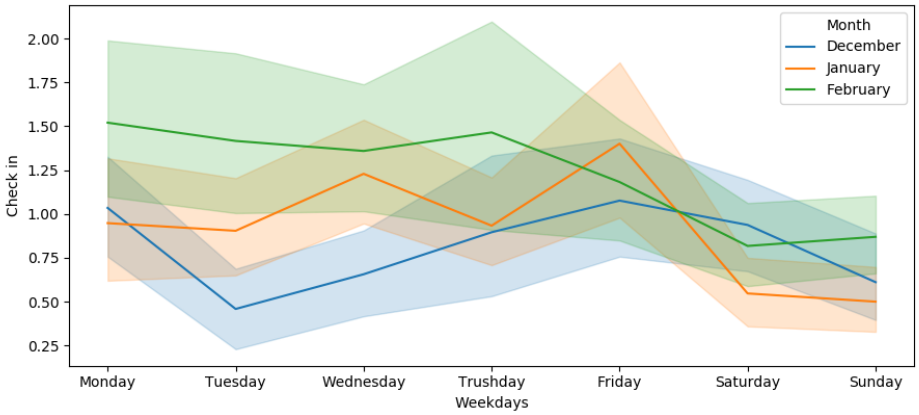


(b) Ist cluster

Figure 5.4: Weekly volume and variation of check-in per cluster (December 2018 to February 2019)



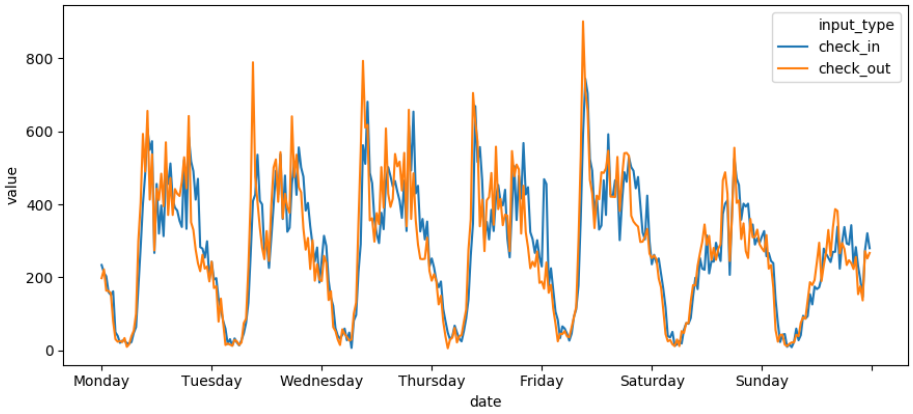
(c) Oriente cluster



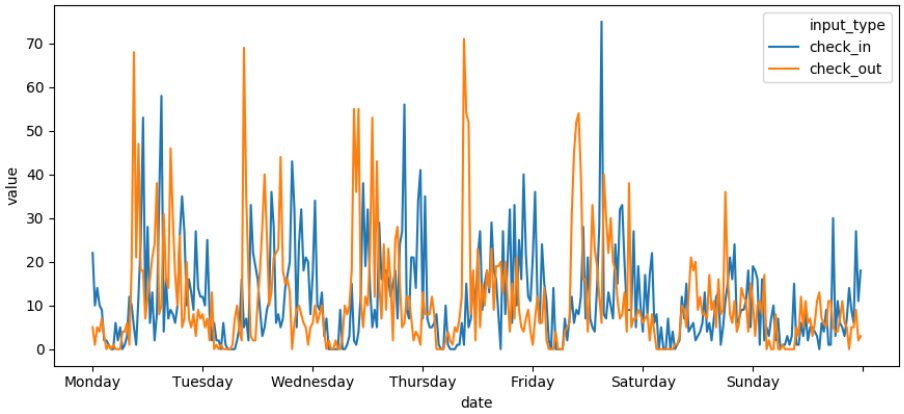
(d) Saldanha cluster

Figure 5.4: Weekly volume and variation of check-in per cluster (December 2018 to February 2019)

The variability of check-ins and check-outs follows a similar patron along the week with a peak, normally, on Friday, especially on the IST cluster. The check-ins and check-outs movement along months are different. During December and January, the check-ins and check-outs values had an upward moment at the begging of the week and a downward movement at the weekends.

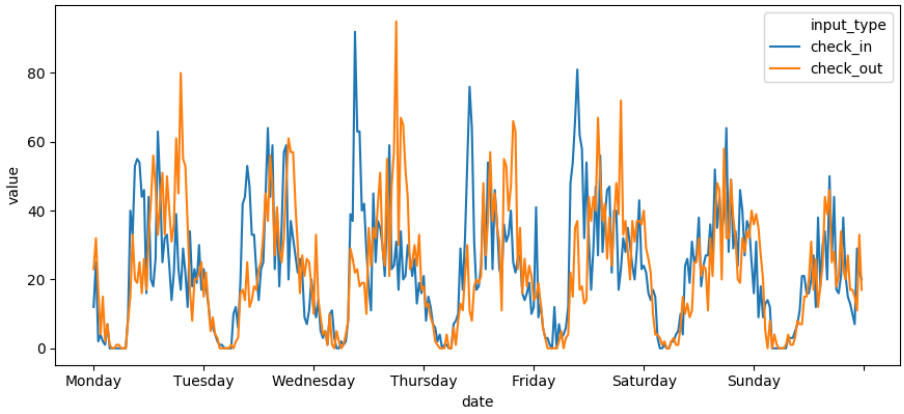


(a) Network(all stations)

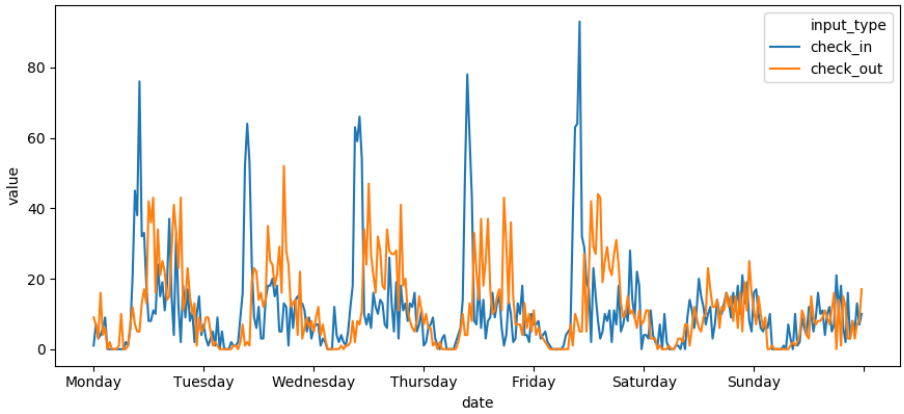


(b) Ist cluster

Figure 5.5: Weekly cumulative volume and variation of check-in and check-outs per cluster (December 2018 to February 2019)



(c) Oriente cluster



(d) Saldanha cluster

Figure 5.5: Weekly cumulative volume and variation of check-in and check-outs per cluster (December 2018 to February 2019)

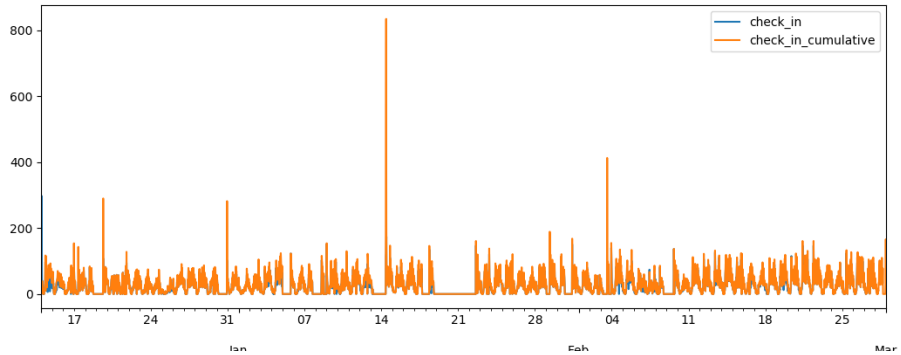
Figure 5.5 provide another top-level perspective on how demand is distributed along a normal week, evidencing the calendrical differences and the effect produced by public events on Saturday night. From all the selected clusters, the Oriente cluster, Figure 5.5c, is the most similar to the GIRA network distribution, Figure 5.5a. The number of stations in the Oriente cluster could be a factor that influences the similarity between them.

The demand in the Saldanha cluster, Figure 5.5c, has a higher peak of check-in them check-outs, indicating that the GIRA users arrive in Saldanha at the same time and leave at different hours during the day. This cluster is most used during the week, hinting that these stations are most likely to be shared by workers.

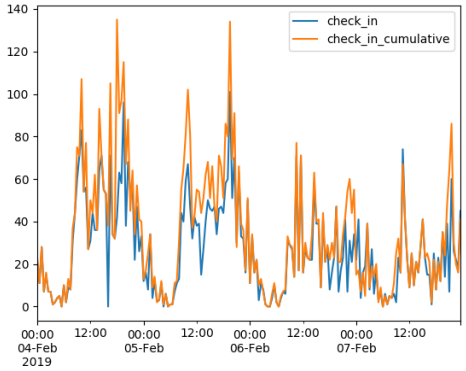
Identical to the Saldanha cluster, the IST cluster has a higher volume of demand during the working days, confirming that this cluster's main users are students of IST.

Figure 5.6 show the difference between the two approaches to generate the check-in and check-out feature. The orange line is the check-in time series generated by approach A, and the blue line represents the check-in time series generated by approach B. In section 4.2.4 explains the difference between the two approaches.

The results of approach A and B on generating the check-in and check-out feature are identical. Figure 5.6 suggests that during must of the 30 minutes, intervals have only one record, but in some intervals, Figure 5.6b, these two methods produce different values.



(a) Check-ins values between 17th of December 2018 and 28th of February 2019



(b) Check-ins values between 4th and 7th of February

Figure 5.6: Different approaches to generate check-ins variable

5.2.1 Validation

We used two different approaches to validate and compare the results, mean absolute error.

The **mean absolute error**, or MAE, is the average of the forecast errors, where all the forecast errors are forced to be positive.

$$\frac{\sum_{t=1}^{n} |\hat{x}_t - x_t|}{n} \tag{5.1}$$

Where x_t is the real value and $\hat{x}_t$ is the forecast value.

Throughout this work the MAE value is calculate with the test instances from the 3 subdataset generate using the methodology explained in 4.3. For example suppose that algorithm A is capable of forecast the check-in demand, the MAE value associate with the algorithm A is the MAE's mean from 3 subdatasets.

The **Student's t-Test** is a statistical hypothesis test for testing whether two samples are expected to have been drawn from the same population. The test works by checking the means from two samples

to see if they are significantly different from each other. It does this by calculating the standard error in the difference between means, which can be interpreted to see how likely the difference is if the two samples have the same mean (the null hypothesis).

The comparison method used to decide if two forecasting methods have different expected was a two-sided the t-Test, with the following hypothesis:

Hypothesis 0. *Method 1 and method 2 have the same expected values*

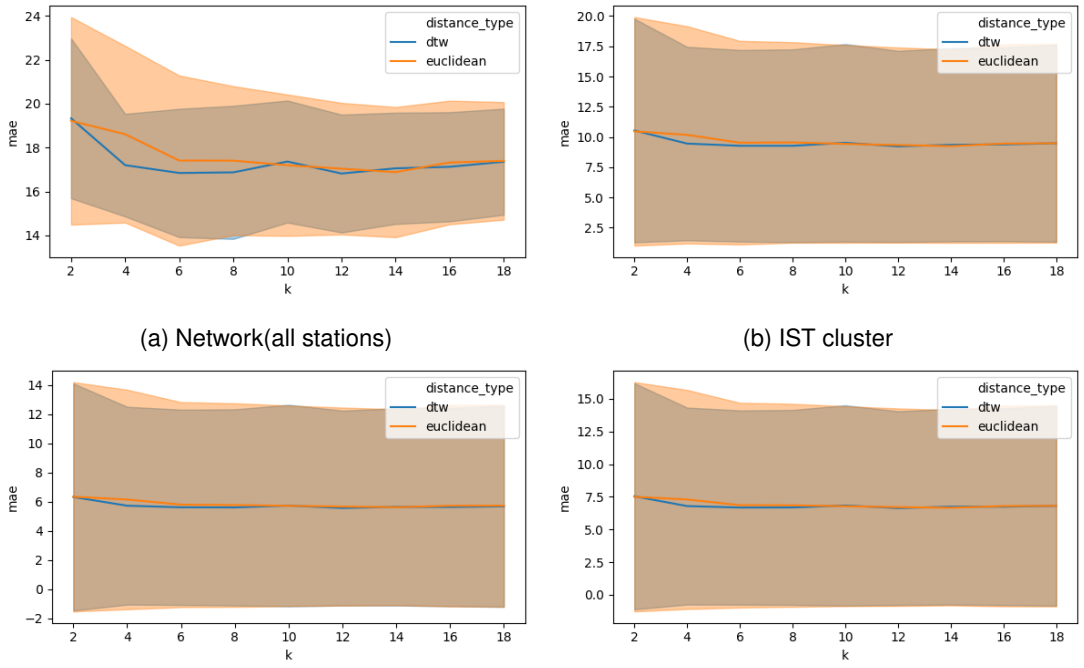
Hypothesis 1. *Method 1 and method 2 have different expected values*

The T-student test is a statistic test over the modeling/forecasting residues produced from two methods when the residues approximately follow a normal distribution. Since our residual values did not follow a normal distribution, we can not apply the t-student test to compare them. Wilcoxon rank is an alternative method to compare two samples that the distribution cannot be assumed to be normal.

5.3 Comparison of state-of-the-art forecaster

5.3.1 Distance-based forecaster

Figures 5.8 and 5.7 illustrate the performance of the KNN method at predicting the check-ins and check-outs, respectively, where the bound on each sub-figure is the standard deviation.



(a) Network(all stations)

(b) IST cluster

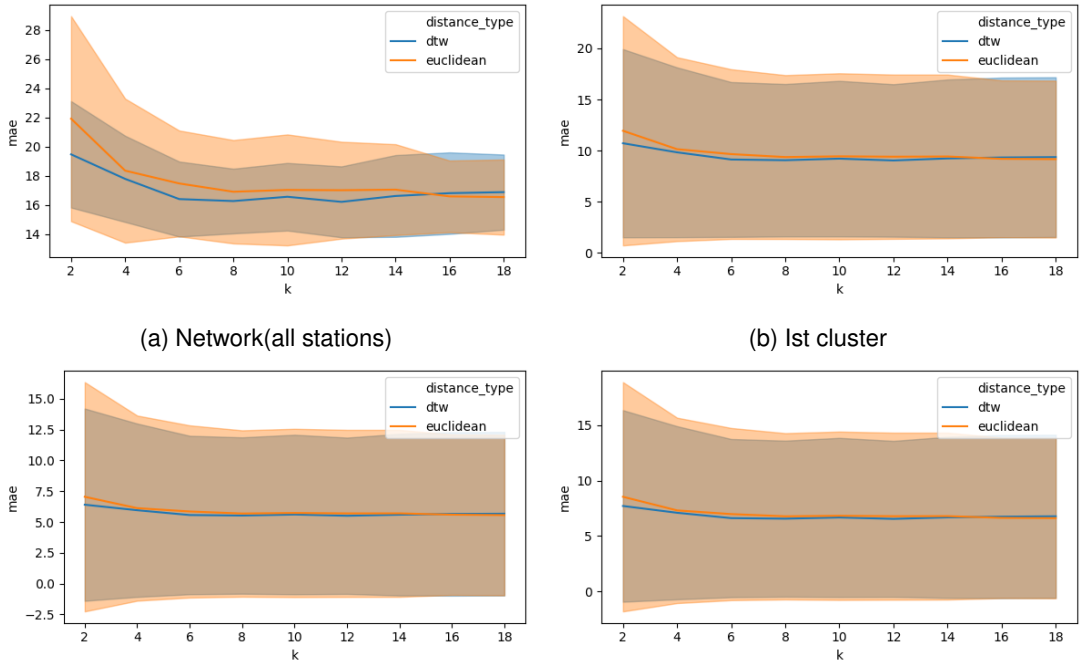
(c) Oriente cluster

(d) Saldanha cluster

Figure 5.7: Predicting the check-out demand using KNN method with DTW and euclidean distances

Both distance metrics, DTW and euclidean, have a similar performance at predicting the check-in demand. For smaller K, the KNN using DTW distance has a slightly smaller MAE value. Around the

10th neighborhood, the difference in performance between the two types of KNN decreases. For must of the Ks, the KNN-DTW has a smaller standard deviation when comparing with KNN-Euclidean.



(a) Network(all stations)

(b) Ist cluster

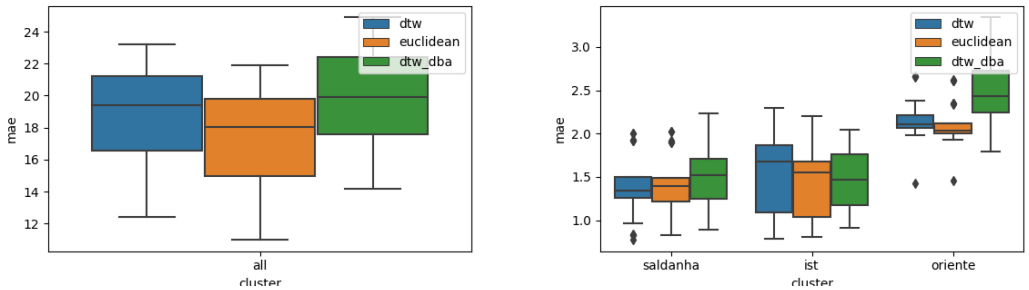
(c) Oriente cluster

(d) Saldanha cluster

Figure 5.8: Predicting the check-in demand using KNN method with DTW and euclidean distances

The performance pattern along the number of Ks between the check-ins and check-outs are identical. On the one hand, Knn-DTW has a better performance for Ks smaller than 15. On the other hand, for K larger than 15, the Knn-euclidean has a slightly better performance.

Figures 5.9 and 5.10 summarize the results of forecasting the check-in and check-out demand in the different granularity levels. For the following box-plots, an outlier is any data point outside the range 1.5 * interquartile range (IQR).



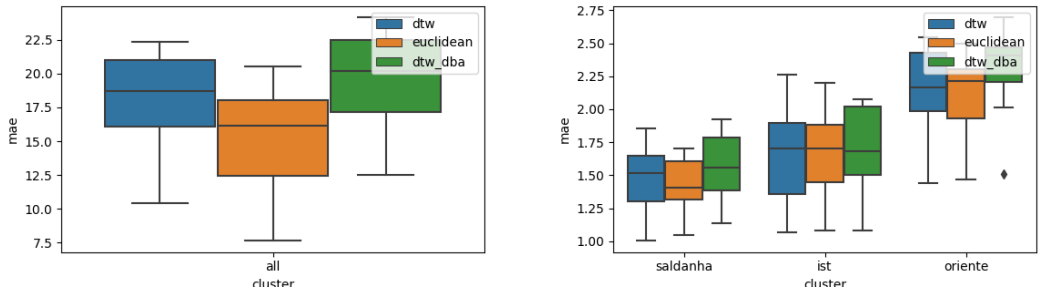
(a) Network(all stations)

(b) Clusters

Figure 5.9: Predicting the check-out demand using barycenter method with DTW, euclidean, DTW dba distances

Forecasting check-out demand has better performance using the euclidean distance instead of DTW or DTW DBA. The Euclidean-barycenter and DTW-barycenter produce more outliers them DTW-DBA-

barycenter from the clusters.



(a) Network(all stations) (b) Clusters

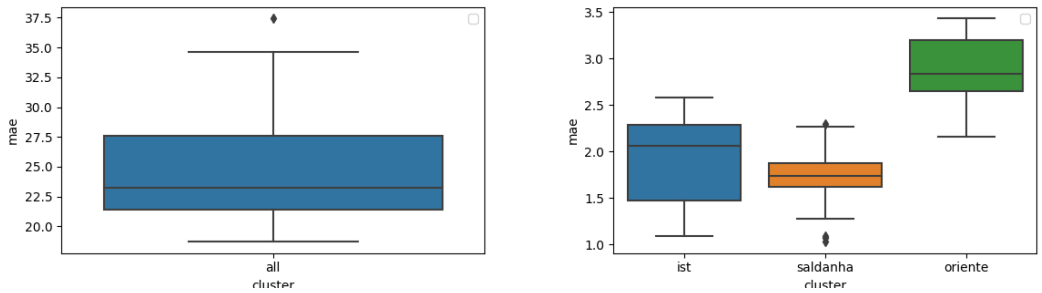
Figure 5.10: Predicting the check-in demand using barycenter method with DTW, euclidean, DTW dba distances

Overall for predicting the check-ins, a euclidean barycenter outperformance the other versions of the barycenter. This result suggests that lag between the time series is not an important factor for predicting the check-in demand.

Many outliers in the check-out forecast suggest that in the test instance were an abnormal number of the user removing bikes from the dock-stations.

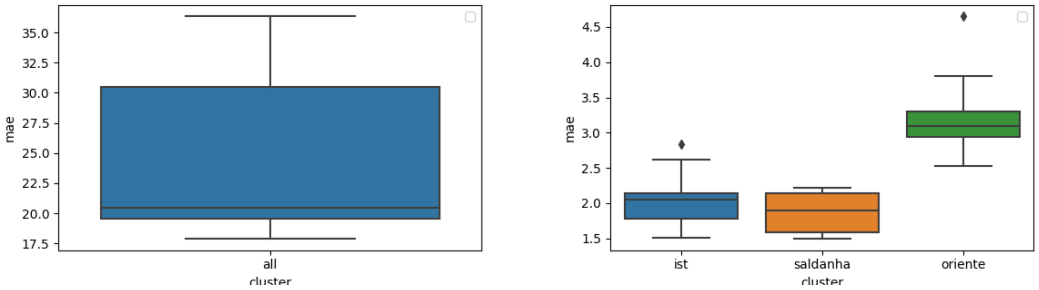
5.3.2 Classical forecasters

This section explores the main results of the holts-winters method for predicting the check-ins and check-outs tasks. For all the box-plots, the holts-winter models were optimized.



(a) Network(all stations) (b) Clusters

Figure 5.11: Predicting the check-out demand using Holts-winters



(a) Network(all stations) (b) Clusters

Figure 5.12: Predicting the check-in demand using Holts-winter

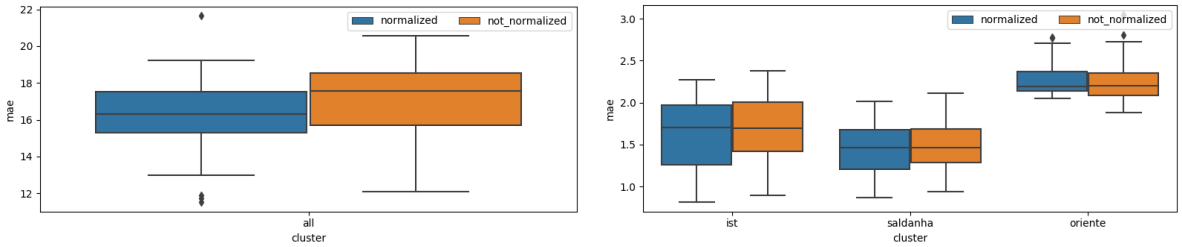
5.4 Neural network forecasting

5.4.1 Scaling

Deep learning neural network models learn to map between the input variable and the output variable. A dataset with large input features can lead to large weight values leading to an unstable model.

To minimize the impact of having features with a different unit, we decided to scales the feature using the min-max scaling method, more detail in section 4.2.5.

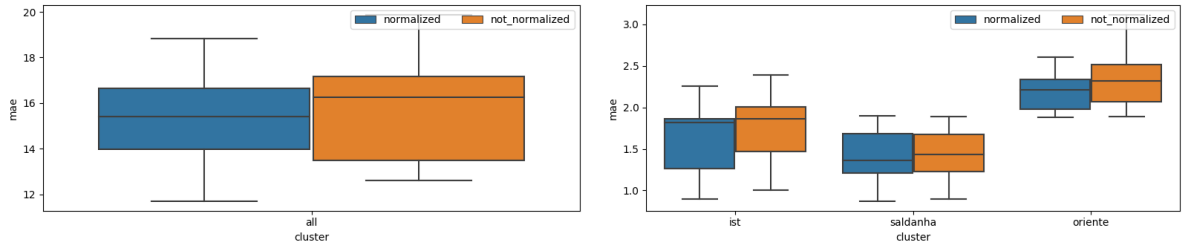
Figure 5.13 exposes the effect of re-scaling the input and output variables. The boxplots below were obtained using an LSTM with a univariate time series, a check-out time series.



(a) Network(all stations) (b) clusters

Figure 5.13: Normalization effect on estimating the check-outs

The use of scaled data improved the network, IST, and Saldanha forecast models. In the Oriente cluster, the unscaled model out-performed the scaled model. In most cases, scaling the check-out variable can lead to a better model.



(a) Network(all stations)

(b) clusters

Figure 5.14: Normalization effect on estimating the check-ins

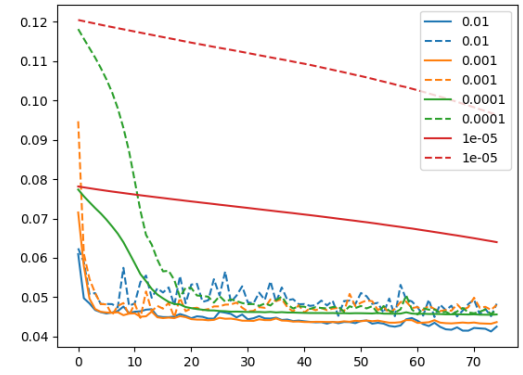
Figure 5.14 displays the performance of training an LSTM using a scaled and unscaled check-in time series. The forecast performance difference between the scaled and not scaled check-in values is very similar to the forecast check-out difference.

Overall, scaling the check-in and check-out can improve the forecast performance. However, the difference between the two types of data is not statistically relevant.

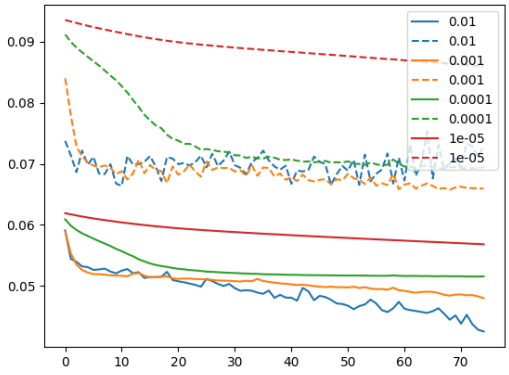
5.4.2 Learning rate

Deep learning neural networks are trained using the stochastic gradient descent algorithm. The gradient descent algorithm is an optimization method that estimates the error gradient for a state and then updates the model's wights using back-propagation. The learning rate is a positive scalar that determinate the amount that the wights are updated during training.[47]

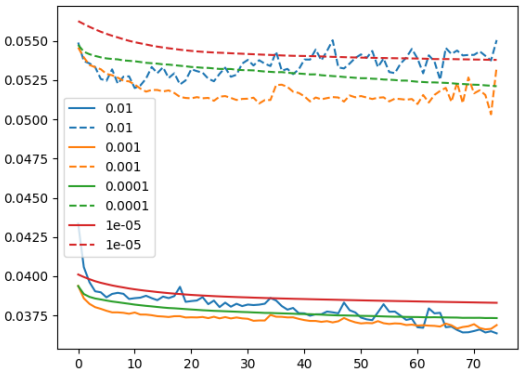
In the following Figures, 5.15 and 5.16, it is detailed the variance in performance using different of the learning rates.



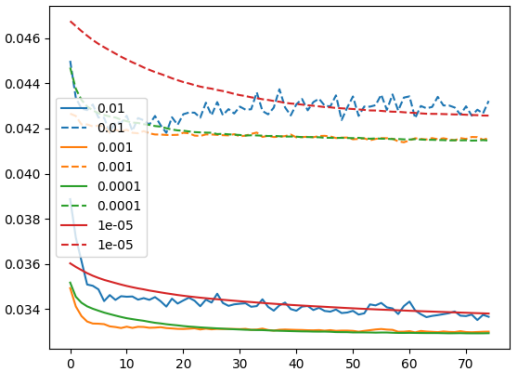
(a) Network(all stations)



(b) Oriente cluster



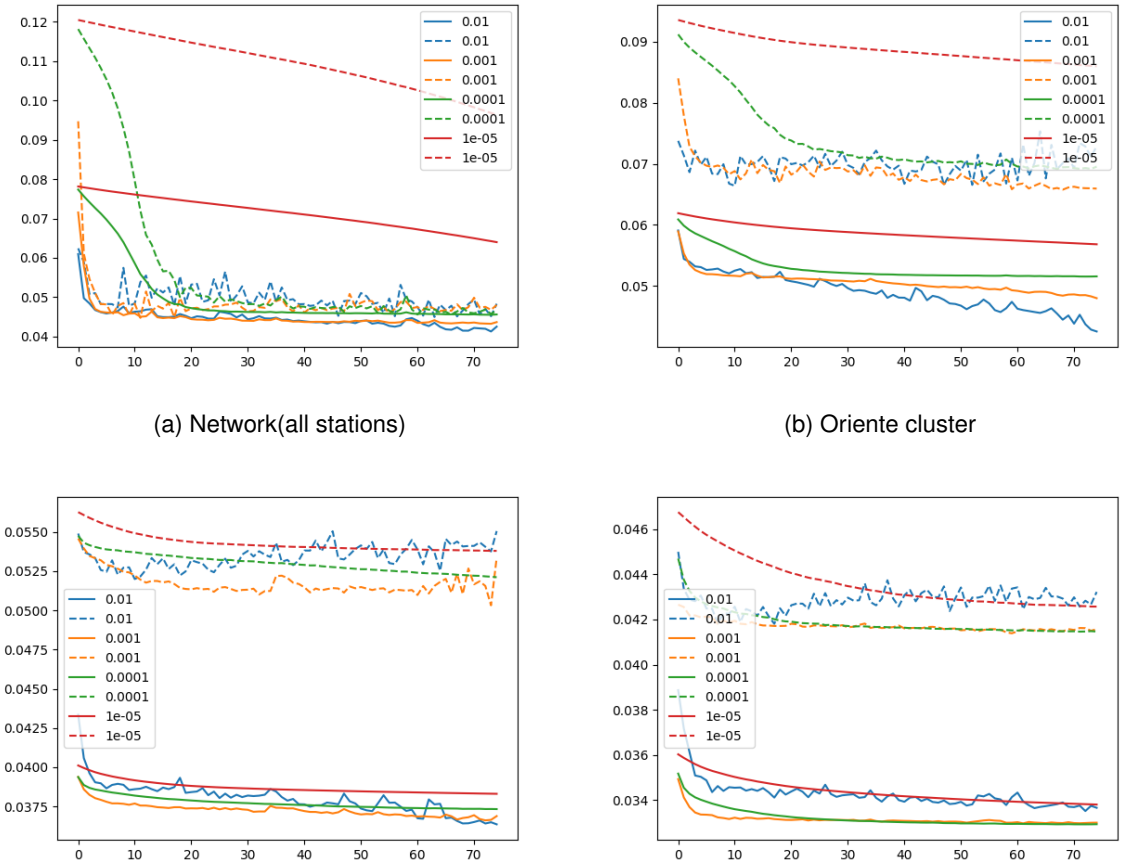
(c) Saldanha cluster



(d) IST cluster

Figure 5.15: Leaning rate effect for predicting the check-ins

The most suitable learning rate for predicting the check-in values examined is 0.01 for the train instances in most granularity levels. However, the learning rate with the highest performance for the validation instances differences a lot between clusters. The learning rate curve for the value 0.0001 is similar in Figures 5.15a and 5.15b. This similarity in the learning rate trajectory suggests that Oriente is the adequate cluster for representing all the GIRA networks.



(a) Network(all stations)

(b) Oriente cluster

(c) Saldanha cluster

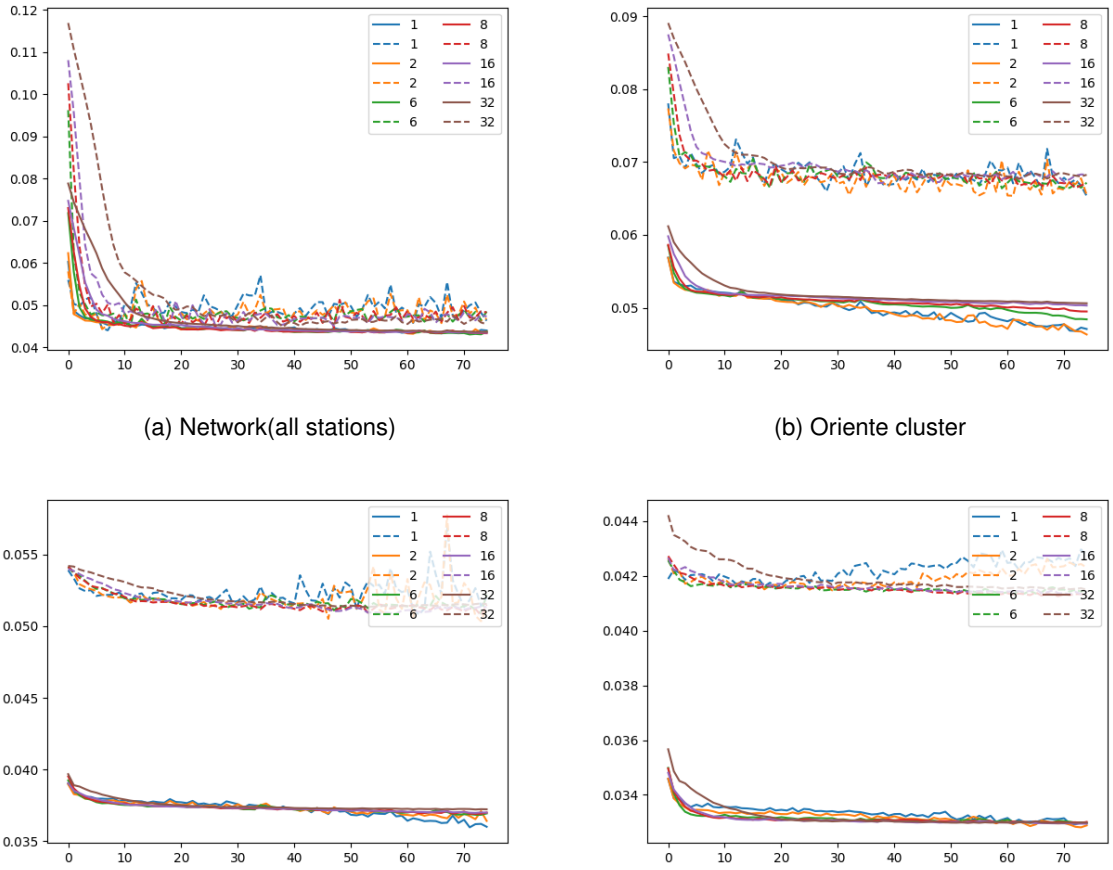
(d) IST cluster

Figure 5.16: Leaning rate effect for predicting the check-outs

5.4.3 Batch size

Batch size is an important hyperparameter for training any neural network. Batch size is the number of the instance before updating the neural network interval parameters.

Figure 5.17 assesses the average loss error along different batch sizes. This results were obtained using a C1 architecture, Figure 4.12, with 64% training instances, 16% validation instances and 20% test instances. The learning rate was 1e-3 with ADAM optimizer throughout the different batch-sizes. The C1 neural network had an early stop trigger activated when the validation loss did not improve.



(a) Network(all stations)

(b) Oriente cluster

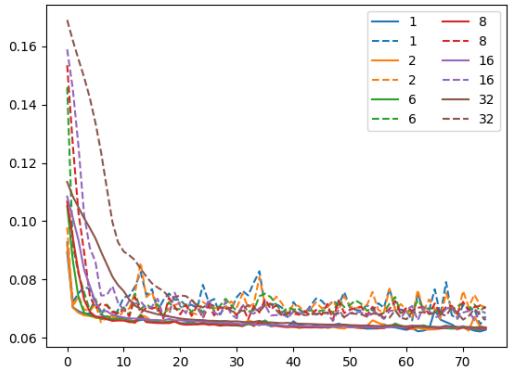
(c) Saldanha cluster

(d) IST cluster

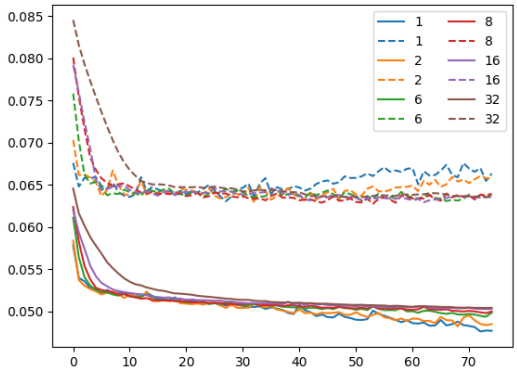
Figure 5.17: Batch size impact for predicting the check-ins

Figure 5.17a represents the GIRA network. This figure suggests that the batch size is not an essential parameter since the different batch sizes have similar performance. However, the validation loss with smaller batch sizes has greater fluctuations in the loss value.

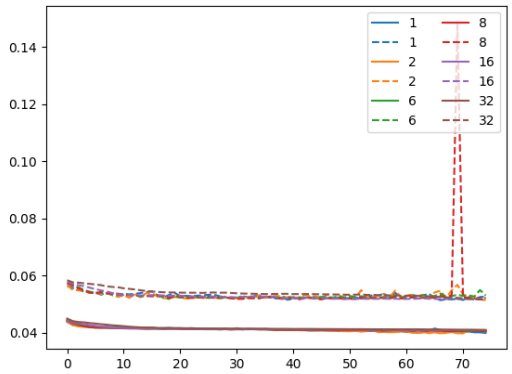
The batch size with values one and two outperforms the rest of the train data's batches sizes, Figure 5.17c. This result indicates that for the validation data, these two batches sizes should generate the best models. Nevertheless, for the validation instance, these two batches sizes produce models with considerable loss fluctuations.



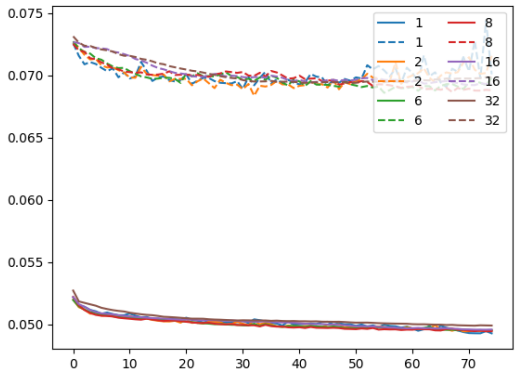
(a) Network(all stations)



(b) Oriente cluster



(c) Saldanha cluster



(d) IST cluster

Figure 5.18: Batch size impact for predicting the check-outs

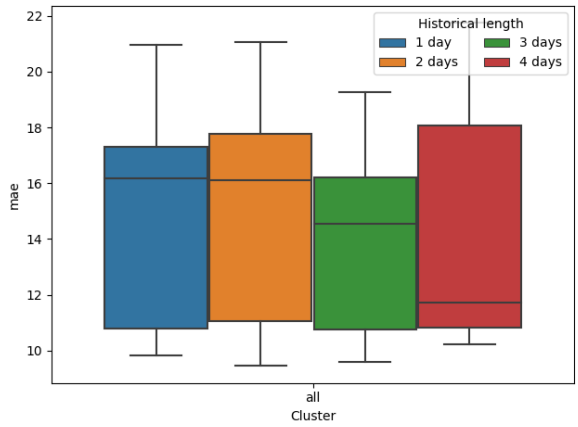
The batch size effect on forecasting the check-outs, Figure 5.18, is identical to the impact on predicting the check-ins. The batches of one and two instances (Figure 5.18b) have the lowest loss value for the train data, yet these two batch sizes produce models with high loss variation for the validation instance. This behavior is similar to Figure 5.17c.

5.4.4 Historical and horizon lengths

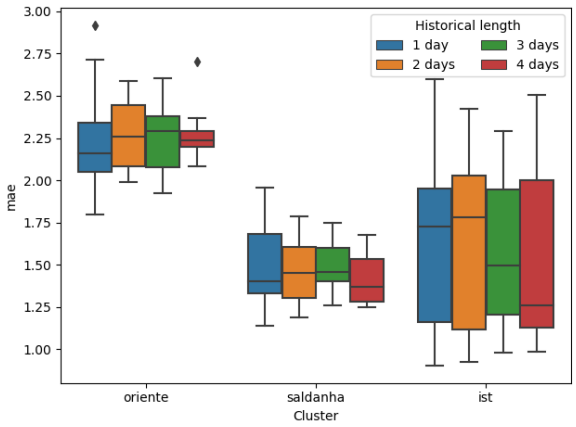
This section shows the results of varying the instances lengths, modifying the historical and horizon dimensions. A day was represented by 48 data points with 30 minutes between each data point.

Figures 5.19a and 5.19b shows the impact of varying the historical number of days. These results suggest that longer historical periods improve the forecast. This improvement can be uncertain since the test dataset has only two instances.

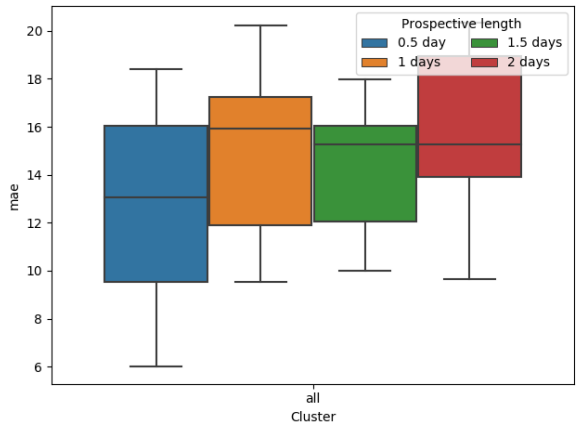
To understand the horizon length's impact on predicting the check-ins, we trained different models changing the horizon size. The effects on the forecast of check-in variables can be seen in Figures 5.19c and 5.19b. Throughout the GIRA network and the selected clusters, the lower error models were obtained using half of a day as a horizon length.



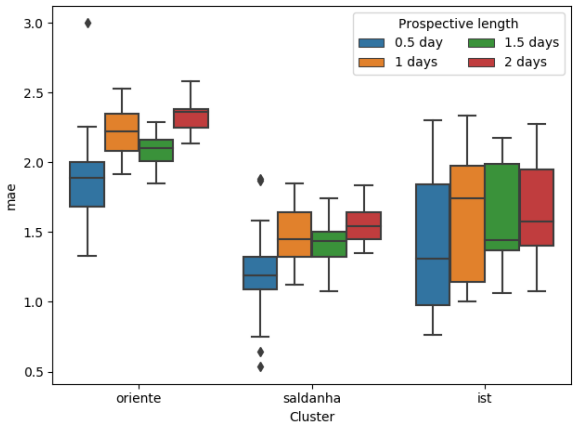
(a) Network(all stations) varying the historical length with horizon size of 1 day



(b) Clusters results varying the historical length length with horizon size of 1 day



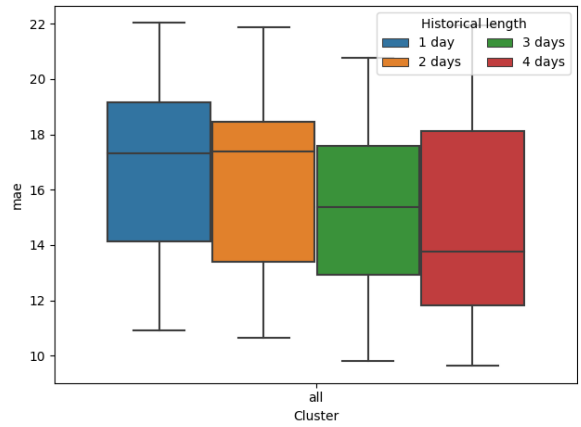
(c) Network(all stations) results varying the horizon length with historical size of 2 days



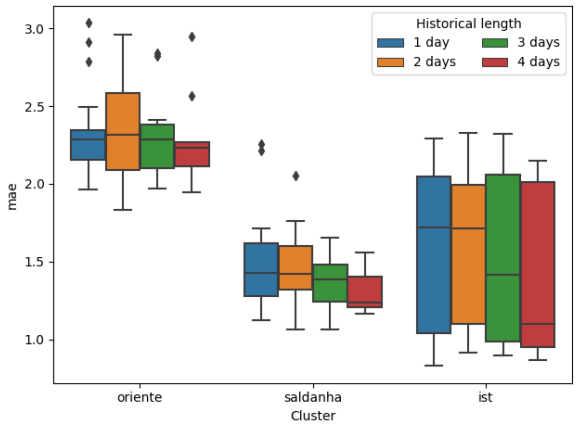
(d) Clusters results varying the horizon length with historical size of 2 days

Figure 5.19: Impact of historical and horizon lengths at predicting the check-ins

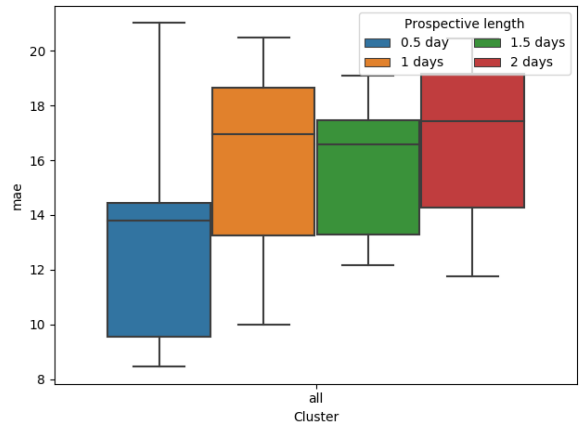
Applying the same settings used for analyzing the influence of historical and horizon length for predicting the check-in, we investigate the impact of forecasting the check-outs, Figure 5.20. The results were similar to the horizon being half a day, providing the most suitable models for predicting the check-out values.



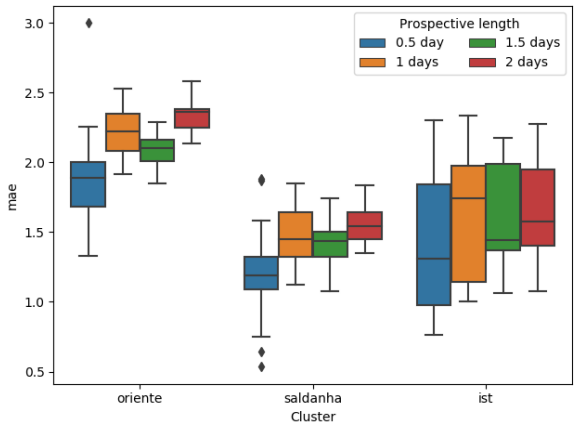
(a) Network(all stations) varying the historical length with horizon size of 1 day



(b) Clusters results varying the historical length length with horizon size of 1 day



(c) Network(all stations) results varying the horizon length with historical size of 2 days



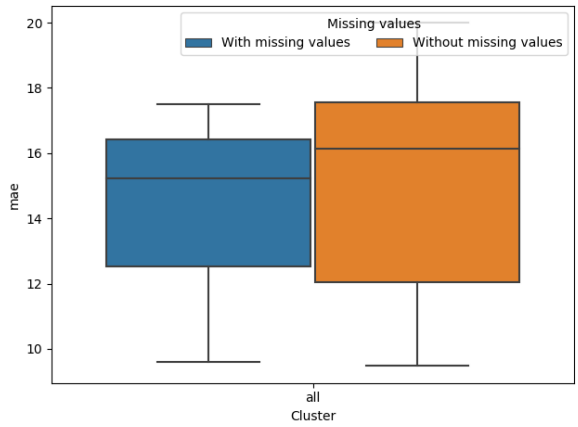
(d) Clusters results varying the horizon length with historical size of 2 days

Figure 5.20: Impact of historical and horizon lengths at predicting the check-ins

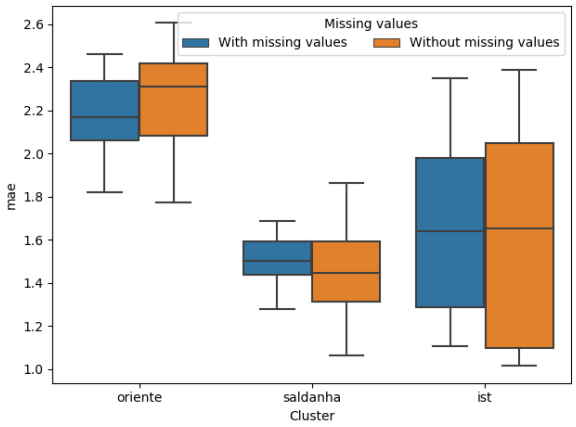
5.4.5 Missing periods and non-working hours

In this section, we study the impact of missing values and the removal of non-working periods. Throughout this section, we consider missing values for any period of 5 hours without any record.

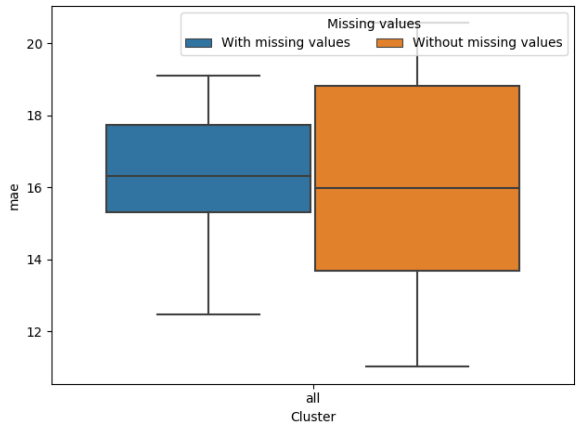
Figure 4.6 displays the forecaster's performance with and without missing values. These results were obtained using LSTM. For the majority of the models learned, eliminating the missing intervals reduced the quartile sizes. This reduction suggests that removing missing values will improve the performance of the forecasts.



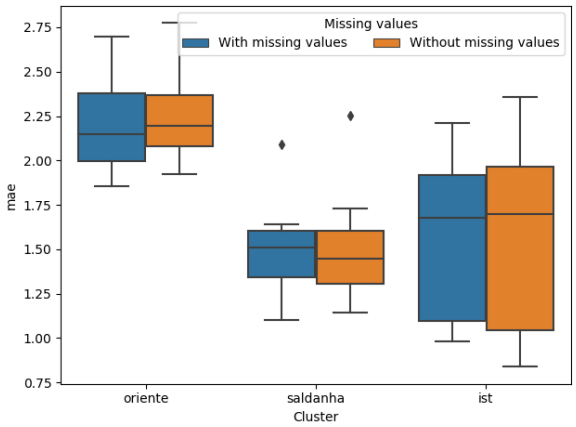
(a) Impact on predicting check-ins by removing missing values: GIRA network



(b) Impact on predicting check-ins by removing missing values: Clusters



(c) Impact on predicting check-outs by removing missing values: GIRA network

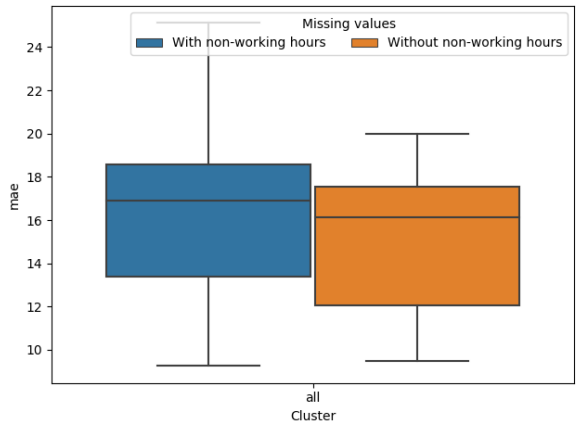


(d) Impact on predicting check-outs by removing missing values: Clusters

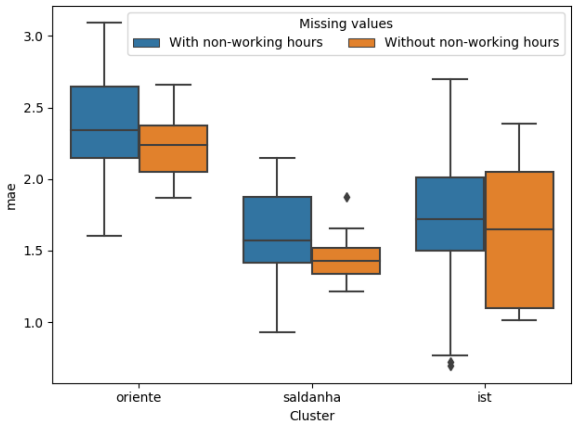
Figure 5.21: Impact on predicting check-ins and check-outs by removing missing values

To test the impact of non-working hours, we obtain for each cluster two types of LSTM, LSTM with non-working hours and LSTM without non-working hours. By eliminating non-working schedules, we reduce the number of available data points per day from 48 to 40.

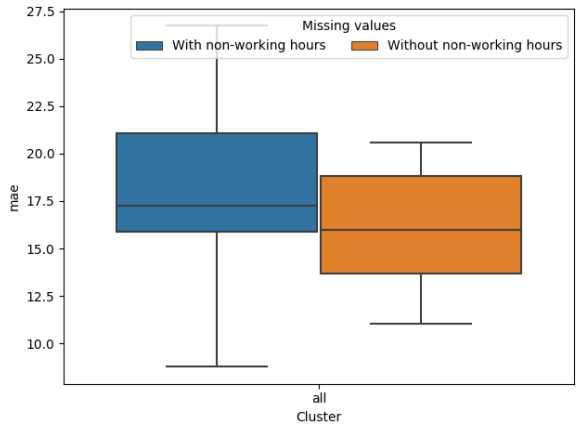
Figure 5.22 displays the results of removing non-working periods for predicting the check-ins, Figures 5.22a and 5.22b, and check-outs, Figures 5.22c and 5.22b. For the clusters Oriente and Saldanha and for the GIRA network eliminating the non-working schedules improved the model's performance at predicting the check-ins and check-outs. Although models learned using the IST cluster data did not show any signal of improvement.



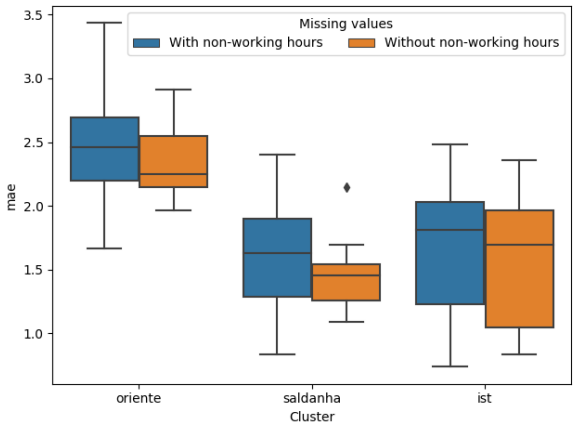
(a) Impact on predicting check-ins by removing working hours: GIRA network



(b) Impact on predicting check-ins by removing working hours: Clusters



(c) Impact on predicting check-outs by removing working hours: GIRA network



(d) Impact on predicting check-outs by removing working hours: Clusters

Figure 5.22: Impact on predicting check-ins and check-outs by removing working hours

5.5 Context impact

In this section, we describe the influence on the check-in and check-out performance using different context variables. Subsection 1 reveals the impact of combining historical context variables for predicting the check-in and check-out variables. Subsection 2 displays the results of incorporating historical and available prospective meteorology variables. Subsection3 shows the forecast performance using spatial features.

Figures 5.23 and 5.24 provide two illustrative data instances that clearly show the impact of incorporating meteorological context data on the forecasts. Figure 5.23 considers the check-in demand at the whole network level, clearly showing the indisputable role of incorporating weather record data for improving predictions. Figure 5.24 considers the check-in demand at the Oriente cluster level, showing the impact from the integration of multiple sources of context.

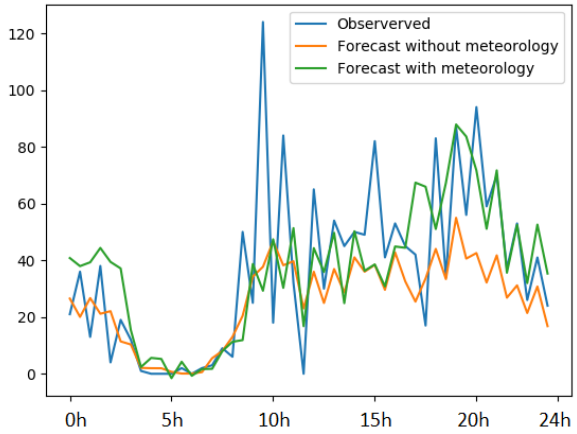


Figure 5.23: Aleatory testing data instance: true observations vs. LSTM forecasts vs. weather-aware LSTM-based forecasts considering check-ins for all GIRA stations.

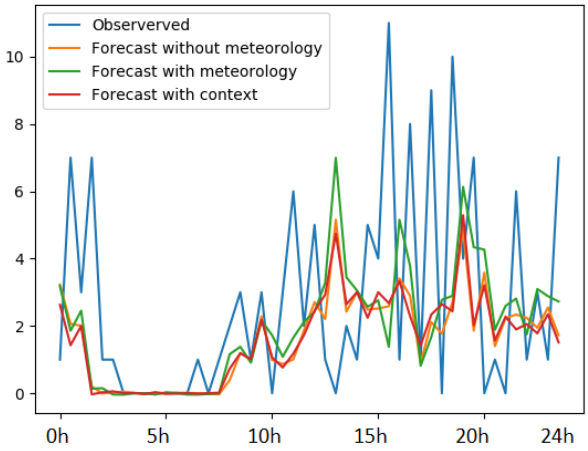


Figure 5.24: Hard testing data instance: true observations vs. LSTM forecasts vs. context-aware LSTM-based forecasts considering check-ins for Oriente cluster of stations.

Table 5.1: Comparison of forecasting errors (MAE and RMSE) of state-of-the-art predictors: *check-in demand* at 30 minutes granularity.

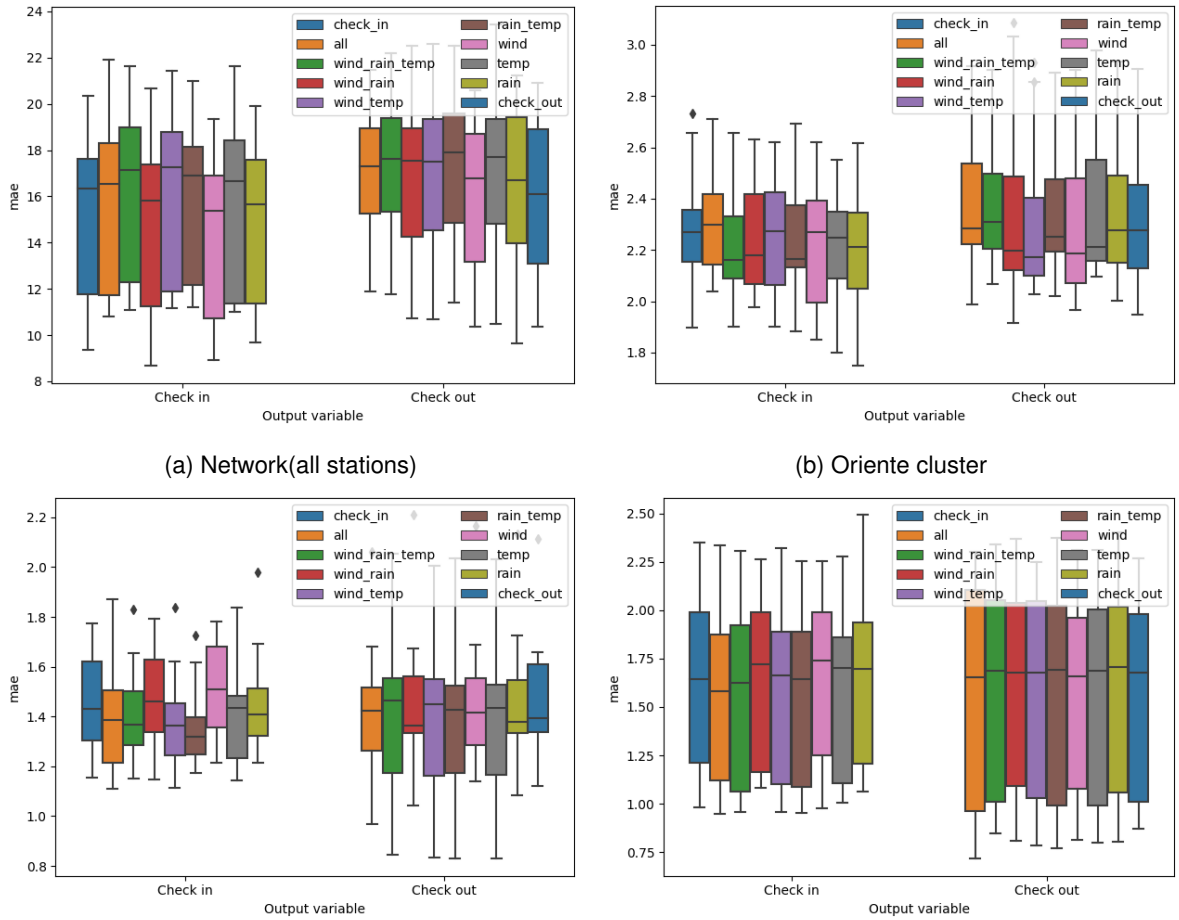
	Senario 1: all stations		Senario 2: Oriente cluster		Senario 3: single station	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
KNN - DTW	16.57± 2.27	23.52± 3.84	2.42±0.23	3.81±0.48	0.35±0.06	0.79±0.14
KNN - Euclidean	17.03±3.72	24.06±6.02	2.46±0.35	3.82 ±0.64	0.34±0.06	0.78 ±0.14
Barycenter - DBA	19.81±3.32	27.87±4.86	2.47±0.43	3.82 ±0.54	0.34 ± 0.05	0.85 ± 0.15
Barycenter - Euclidean	17.01±3.42	24.81±5.35	2.11±0.25	3.25 ±0.44	0.35 ±0.05	0.70 ± 0.21
Barycenter - DTW	18.57±3.28	26.87±4.92	2.16±0.26	3.35 ±0.46	0.36 ± 0.05	0.71 ± 0.22
Holts-Winters	24.11±6.19	33.94±14.88	3.17±0.45	4.64 ±0.45	1.01± 0.33	1.99 ± 0.73
LSTM check-in	15.07±3.41	20.87±3.4	2.26±0.23	3.45 ±0.23	0.24± 0.05	0.72 ± 0.14
LSTM check-in and check-out	14.49±3.14	20.48±3.15	2.17±0.22	3.38 ±0.21	0.24 ± 0.05	0.72 ± 0.14
LSTM day mask	13.97±3.97	20.35±3.97	2.24±0.23	3.51 ±0.22	0.24± 0.05	0.72 ± 0.14
LSTM nearby	-	-	2.23±0.26	3.38 ±0.26	0.36 ± 0.05	0.68 ± 0.14
LSTM all meteo	15.75±3.80	22.01±3.80	2.31±0.19	3.62 ±0.19	0.24 ± 0.05	0.72±0.14
LSTM all meteo and nearby	-	-	2.12±0.27	3.45 ±0.68	0.36 ± 0.05	0.68 ± 0.14

Table 5.2: Comparison of forecasting errors (MAE and RMSE) of state-of-the-art predictors: *check-out demand* at 30 minutes granularity.

	Senario 1: all stations		Senario 2: Oriente cluster		Senario 3: single station	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
KNN - DTW	17.37± 2.72	24.21±3.64	2.44±0.42	3.64±0.55	0.36±0.05	0.88±0.18
KNN - Euclidean	17.20±3.15	23.78±4.43	2.54±0.33	3.77 ±0.44	0.34±0.06	0.81 ±0.21
Barycenter - DBA	19.70±3.06	27.18±5.05	2.35±0.26	3.82 ±0.56	0.36 ± 0.08	0.81 ± 0.14
Barycenter - Euclidean	15.62±3.18	22.30±5.29	2.11±0.29	3.33 ±0.55	0.36 ±0.05	0.67 ± 0.14
Barycenter - DTW	18.25±2.95	25.85±4.96	2.16±0.31	3.48 ±0.56	0.36 ± 0.03	0.66 ± 0.12
Holts-Winters	25.00±4.94	34.78±4.94	2.91±0.35	4.21 ±0.35	3.10± 1.48	4.14 ± 1.79
LSTM check-out	15.73±1.59	22.31±3.59	2.32±0.29	3.37 ±0.29x	0.35 ± 0.05	0.71 ± 0.21
LSTM check-in and check-out	16.08±1.84	23.00±3.22	2.24±0.22	3.46±0.41	0.35 ±0.05	0.70± 0.21
LSTM day mask	16.60±4.50	23.35±4.50	2.34±0.32	3.48 ±0.32	0.36 ± 0.05	0.71 ± 0.21
LSTM nearby	-	-	2.16±0.22	3.30 ±0.44	0.35± 0.05	0.71 ± 0.21
LSTM all meteo	16.02±1.70	23.14±3.40	2.42 ±0.32	3.62± 0.49	0.36 ± 0.06	0.72± 0.23
LSTM all meteo and nearby	-	-	2.16 ±0.22	3.30± 0.43	0.35 ± 0.05	0.71± 0.21

5.5.1 Historical context

The impact of using meteorological context features was split into two segments: understand the impact of utilizing all context variables; comprehend the performance weight of rain, wind, and temperature variables at predicting check-ins and check-outs.Figure 5.25represents the influence of using various historical meteorological variables at forecasting the check-in and check-out values.



(a) Network(all stations)

(b) Oriente cluster

(c) Saldanha cluster

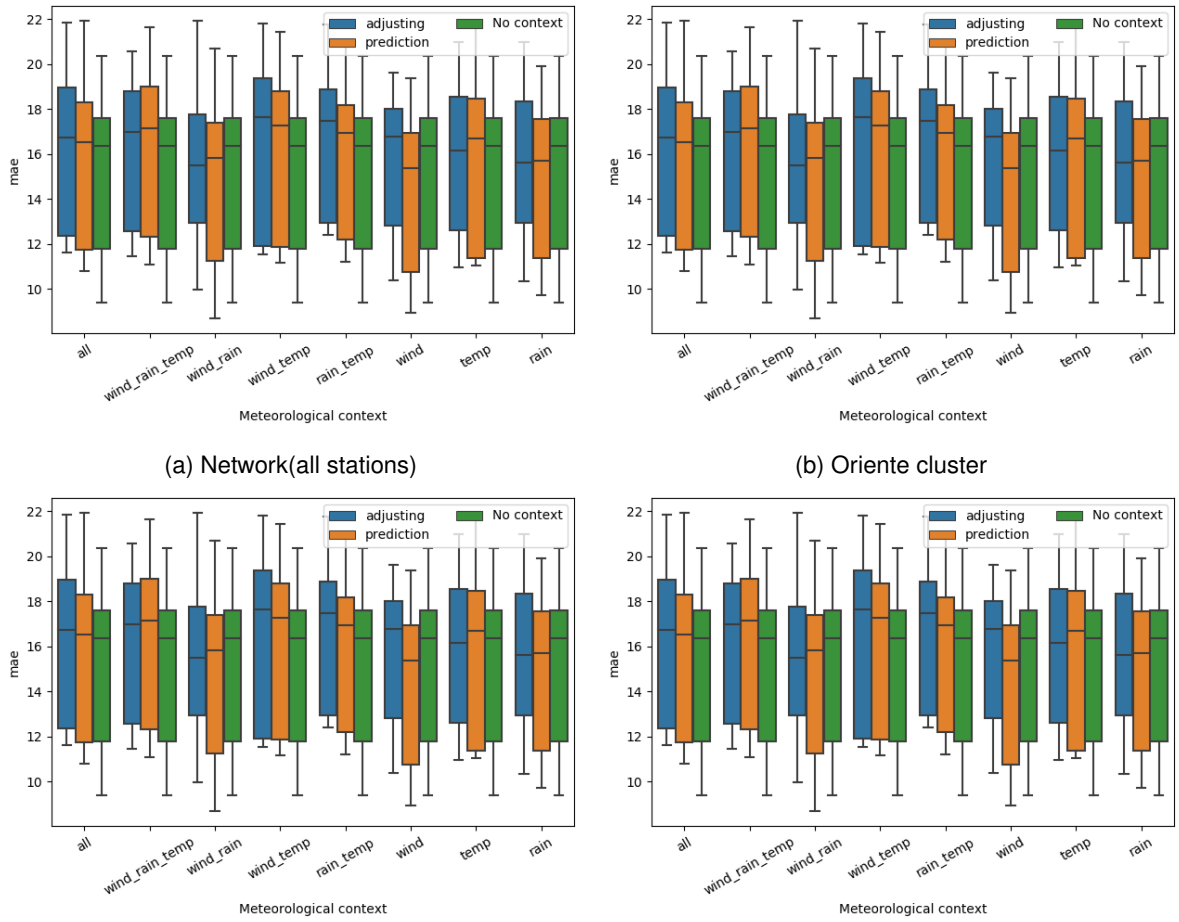
(d) IST cluster

Figure 5.25: Impact of historical meteorology at predicting the check-ins and check-outs variables

The check-in forecaster with the smallest error, Figure 5.25a, is the model trained with the features of wind and wind-rain. For predicting the check-out, the model without the historical context variables has lowerest the MAE value between the multiple models. At the Oriente cluster, Figure 5.25b, the combination of the variables wind, rain, and temperature had the most promising results at forecasting the check-ins. The impact of the historical context variable at the IST cluster, Figure 5.25d, is very small. This outcome can be explained by the fact that this cluster as only two stations.

5.5.2 Historical and prospective context

In section 4.5.1, we present a neural network architecture that should be able to learn the check-in and check-out using historical and prospective data. In this section, we display the results of the proposed architecture.



(a) Network(all stations)

(b) Oriente cluster

(c) Saldanha cluster

(d) IST cluster

Figure 5.26: Impact of historical and prospective meteorology feature at predicting the check-ins values

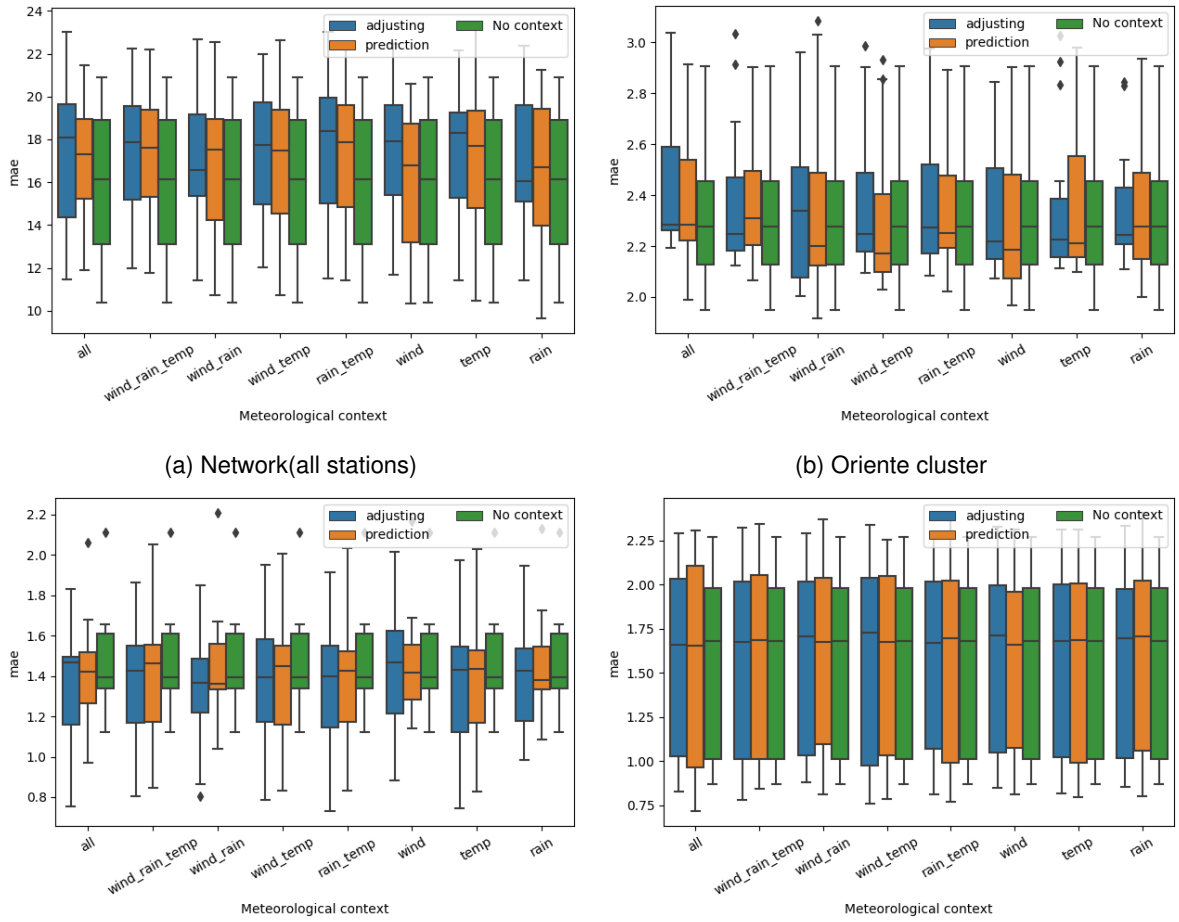
At the GIRA network level, Figure 5.26a using weather variables for adjusting the prediction made by the first layer,C1 layer in Figure 2.4, of the neural network did not improve the prediction.

In some cases, selecting weather variables for regulating the check-in results can reduce the median error slightly. For example, the rain results in Figure 5.26a.

Figure 5.27 exhibits the MAE error for predicting the check-out demand using historical and prospective weather variables.

Predicting the check-outs and check-ins values at the GIRA level had a similar outcome, i.e., in some cases, using prospective weather data can lead to a decrease in the median.

For the Oriente cluster, Figure 5.27b, using historical and prospective variables improve the prediction.



(a) Network(all stations)

(b) Oriente cluster

(c) Saldanha cluster

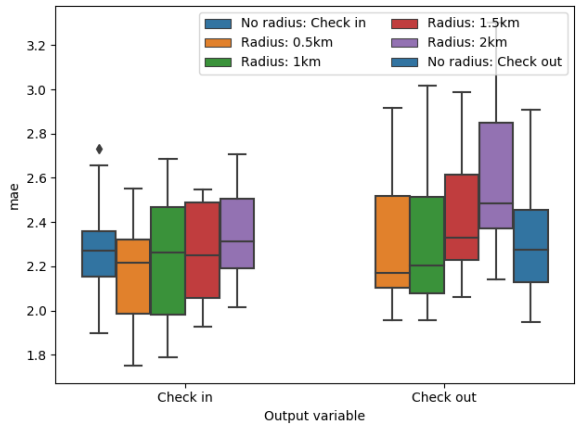
(d) IST cluster

Figure 5.27: Impact of historical and prospective meteorology feature at predicting the check-outs values

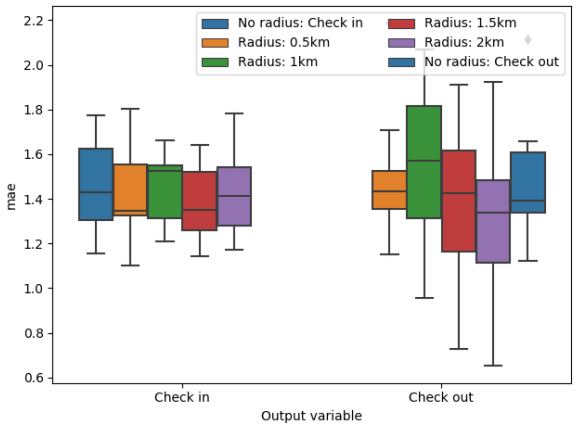
5.5.3 Spatial context

In this section, we expose the impact that stations can have in each other. The influence was studied by using a spatial mask, more details in section 5.5.3, with different radius values.

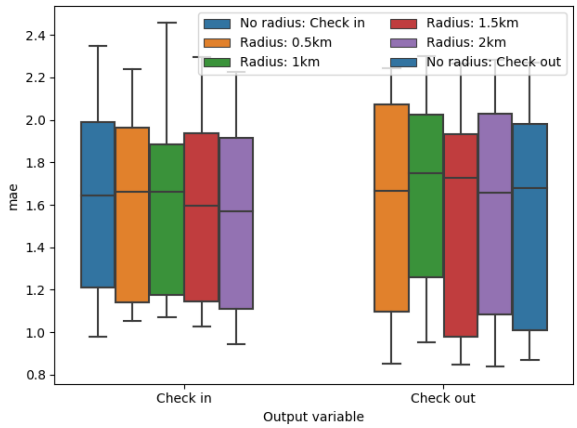
Figure 5.28 reveals the influence of spatial features. In the Oriente cluster results, Figure 5.28a, suggests that a larger radius decrease the model's performance. On the other hand, the model trained with spatial data from stations within two km of radius had the least error for the Saldanha cluster, Figure 5.28b. For most of the radius values, using spatial context improves the forecaster performance.



(a) Oriente cluster



(b) Saldanha cluster



(c) IST cluster

Figure 5.28: Influence spatial feature at predicting the check-in and check-outs

Chapter 6

Conclusions

The existing literature on predictive tasks fails to model the impact that sources of context data can have on bike-sharing demand and fails to separate the influence of both historical and prospective sources of context. In this dissertation, we provided a comprehensive analysis of the impact that context data manifests on predicting the bike demand and evaluation of the proposed context-awareness forecasters against state-of-the-art forecasters at different spatial and temporal granularities.

To support Lisbon city hall decision regarding the GIRA systems, we provide a tool that allows visualisation of GIRA's stations capacity and bike demand, as well as context-enriched visualisation of BSS dynamics.

This work focused on the GIRA study case (77 stations, and approximately 700 bikes and 700 to 1500 daily trips). With the goal of incorporate heterogeneous sources of spatial, meteorological and calendrical context, we propose a new class of recurrent neural layering for context-sensitive demand prediction, which is up to the state-of-the-art models. Firstly, we examined the components, check-ins and check-outs, of the GIRA network to find mobility patterns. Then, we analysed the performance of state-of-the-art models. Afterwards, we applied the novel neural network architecture to describe the influence of spatial, weather, and calendrical context produces on forecasting the GIRA's bike demand. Further, we examine the fundamental aspects of a neural network, such as, batch size and learning rate, and comprehend the influence that these aspects can produce.

Results provided significant evidence concerning the importance of incorporating historical and prospective context sources can offer to predict the bike demand. Additionally, the results also confirmed the unique role of meteorological conditions in forecasting the check-ins and check-outs.

To conclude, the implications of this work are various. Firstly, a complete, end-to-end dashboard, capable of giving a whole historical picture of GIRA's bike demand, secondly, a comprehensive analysis of distance and classical methods and algorithms, third, an entire review of the impact that context data has on forecasting bike demand using an original neural network architecture, and last but not least, an implementation of the chosen algorithms, KNN, barycenter and the new original LSTM architecture.

Scientific Contributions

With the development of this work, we achieved the following scientific contributions:

- Buildsys '20 (Submitted): *"Context-aware description and prediction of the demand in public bike sharing systems: on the integration of sources of situational, meteorological and calendrical context*

6.1 Future work

Four major future directions are identified:

- Apply the proposed deep learning architecture in other modes of transportation, such as METRO, and Carris.
- Extend the proposed forecast methodology to predict bike demand for mid and long terms, taking into account the available historical data.
- Refine the neural network architecture to improve the performance of incorporating the context data.
- Develop the context mask principals to distance-based approaches.

Bibliography

[1] Felix Creutzig, Rainer Mühlhoff, and Julia Römer. Decarbonizing urban transport in european cities: four cases show possibly high co-benefits. *Environmental research letters*, 7(4):044042, 2012.

[2] Mark J Nieuwenhuijsen. Urban and transport planning pathways to carbon neutral, liveable and healthy cities; a review of the current evidence. *Environment International*, page 105661, 2020.

[3] Mikael Karlsson, Eva Alfredsson, and Nils Westling. Climate policy co-benefits: a review. *Climate Policy*, 20(3):292–316, 2020.

[4] Susan Shaheen and Stacey Guzman. Worldwide bikesharing. *Access Magazine*, 1(39):22–27, 2011.

[5] Kayleigh B Campbell and Candace Brakewood. Sharing riders: How bikesharing impacts bus ridership in new york city. *Transportation Research Part A: Policy and Practice*, 100:264–282, 2017.

[6] F Richter. The global rise of bike-sharing. *www. statista. com*, 2018.

[7] Susan Shaheen, Adam Cohen, Nelson Chan, and Apaar Bansal. Sharing strategies: carsharing, shared micromobility (bikesharing and scooter sharing), transportation network companies, microtransit, and other innovative mobility modes. In *Transportation, Land Use, and Environmental Planning*, pages 237–262. Elsevier, 2020.

[8] Ines Frade and Anabela Ribeiro. Bike-sharing stations: A maximal covering location approach. *Transportation Research Part A: Policy and Practice*, 82:216–227, 2015.

[9] Yaping Ren, Leilei Meng, Fu Zhao, Chaoyong Zhang, Hongfei Guo, Ying Tian, Wen Tong, and John W Sutherland. An improved general variable neighborhood search for a static bike-sharing rebalancing problem considering the depot inventory. *Expert Systems with Applications*, page 113752, 2020.

[10] Kyoungok Kim. Investigation on the effects of weather and calendar events on bike-sharing according to the trip patterns of bike rentals of stations. *Journal of transport geography*, 66:309–320, 2018.

[11] Lei Lin, Zhengbing He, and Srinivas Peeta. Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. *Transportation Research Part C: Emerging Technologies*, 97:258–276, 2018.

[12] Youru Li, Zhenfeng Zhu, Deqiang Kong, Meixiang Xu, and Yao Zhao. Learning heterogeneous spatial-temporal representation for bike-sharing demand prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1004–1011, 2019.

[13] Álvaro Lozano, Juan F De Paz, Gabriel Villarrubia González, Daniel H Iglesia, and Javier Bajo. Multi-agent system for demand prediction and trip visualization in bike sharing systems. *Applied Sciences*, 8(1):67, 2018.

[14] Po-Chuan Chen, He-Yen Hsieh, Kuan-Wu Su, Xanno Kharis Sigalingging, Yan-Ru Chen, and Jenq-Shiou Leu. Predicting station level demand in a bike-sharing system using recurrent neural networks. *IET Intelligent Transport Systems*, 14(6):554–561, 2020.

[15] Inês Leite, Anna Carolina Finamore, and Rui Henriques. Context-sensitive modeling of public transport data. 2020.

[16] Elliot Fishman, Simon Washington, Narelle Haworth, and Armando Mazzei. Barriers to bikesharing: an analysis from melbourne and brisbane. *Journal of Transport Geography*, 41:325–337, 2014.

[17] Wafic El-Assi, Mohamed Salah Mahmoud, and Khandker Nurul Habib. Effects of built environment and weather on bike sharing demand: a station level analysis of commercial bike sharing in toronto. *Transportation*, 44(3):589–613, 2017.

[18] Huthaifa I Ashqar, Mohammed Elhenawy, and Hesham A Rakha. Modeling bike counts in a bike-sharing system considering the effect of weather conditions. *Case Studies on Transport Policy*, 7(2):261–268, 2019.

[19] Dimitrios Tomaras, Ioannis Boutsis, and Vana Kalogeraki. Modeling and predicting bike demand in large city situations. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2018.

[20] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1720–1730, 2019.

[21] Susan Shaheen, Adam Cohen, Ismail Zohdy, et al. Shared mobility: current practices and guiding principles. Technical report, United States. Federal Highway Administration, 2016.

[22] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. springer, 2016.

[23] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[24] Raj K Jain. A state space model-based method of seasonal adjustment. *Monthly Lab. Rev.*, 124:37, 2001.

[25] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[26] Charles C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

[27] Peter R Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[28] Christopher M Bishop. *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[29] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[30] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *arXiv preprint arXiv:1910.08233*, 2019.

[31] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, pages 33:1–33:10, New York, NY, USA, 2015. ACM.

[32] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. Rebalancing bike sharing systems: A multi-source data smart optimization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1005–1014. ACM, 2016.

[33] Zidong Yang, Ji Hu, Yuanchao Shu, Peng Cheng, Jiming Chen, and Thomas Moscibroda. Mobility modeling and prediction in bike-sharing systems. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 165–178. ACM, 2016.

[34] Romain Giot and Raphaël Cherrier. Predicting bikeshare system usage up to one day ahead. In *2014 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS)*, pages 22–29. IEEE, 2014.

[35] Lei Lin, Zhengbing He, and Srinivas Peeta. Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. *Transportation Research Part C: Emerging Technologies*, 97:258–276, 2018.

[36] Tien Dung Tran, Nicolas Ovtracht, and Bruno Faivre D'arcier. Modeling bike sharing system using built environment factors. *Procedia Cirp*, 30:293–298, 2015.

[37] Yi Ai, Zongping Li, Mi Gan, Yunpeng Zhang, Daben Yu, Wei Chen, and Yanni Ju. A deep learning approach on short-term spatiotemporal distribution forecasting of dockless bike-sharing system. *Neural Computing and Applications*, 31(5):1665–1677, 2019.

[38] Di Chai, Leye Wang, and Qiang Yang. Bike flow prediction with multi-graph convolutional networks. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 397–400. ACM, 2018.

[39] Longbiao Chen, Daqing Zhang, Leye Wang, Dingqi Yang, Xiaojuan Ma, Shijian Li, Zhaohui Wu, Gang Pan, Thi-Mai-Trang Nguyen, and Jérémie Jakubowicz. Dynamic cluster-based over-demand prediction in bike sharing systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 841–852. ACM, 2016.

[40] Simon Kwoczek, Sergio Di Martino, and Wolfgang Nejdl. Predicting and visualizing traffic congestion in the presence of planned special events. *Journal of Visual Languages & Computing*, 25(6):973–980, 2014.

[41] Ali Soltani, Andrew Allan, Ha Anh Nguyen, and Stephen Berry. Bikesharing experience in the city of adelaide: insight from a preliminary study. *Case studies on transport policy*, 7(2):250–260, 2019.

[42] Nguyen Thi Hoai Thu, Chu Thi Phuong Dung, Nguyen Linh-Trung, Ha Vu Le, et al. Multi-source data analysis for bike sharing systems. In *2017 International Conference on Advanced Technologies for Communications (ATC)*, pages 235–240. IEEE, 2017.

[43] Yan Pan, Ray Chen Zheng, Jiaxi Zhang, and Xin Yao. Predicting bike sharing demand using recurrent neural networks. *Procedia computer science*, 147:562–566, 2019.

[44] Terry T Um, Franz MJ Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 216–220, 2017.

[45] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[46] The GPyOpt authors. GPyOpt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.

[47] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.