POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Spatio-temporal algorithms for predicting the usage of bike-sharing systems

Supervisor
Prof. Paolo Garza

Co-supervisor
Ing. Luca Colomba

Candidate
Bianca Iacomussi

October 2021

Abstract

With the recent concerns about climate change and the tendency to promote smart mobility systems, there is an increasing interest on shared means of transport around the world, such as Bike Sharing Systems (BSS), which constitute a valid green alternative for movements within cities. Given these circumstances, many studies have been carried out by the scientific community to improve the service of Bike Sharing Systems focusing on customer satisfaction, by finding optimal locations for bike stations, studying mobility patterns in the cities of interest, enhancing bike redistribution among stations, offering customers predictions in the next minutes about bike availability in the stations. This thesis aims at contributing at the last two questions by proposing predictive models to forecast in the short term the number of bikes available and free slots in each station of the Barcelona BSS, for example in 20 minutes or an hour. The proposed solution exploits stations' behavior in the previous time instants expressed in terms of the number of available bikes, the number of free slots and the dimension of each station, whether the station is full or empty, as well as the spatial relationships between the stations by means of some spatial-temporal features. The situation of nearby stations was taken into account by computing how many neighbors stations were full or empty and which was the average station occupation in the neighborhood in the previous timestamps. Also a distinction between working days or holidays was made which has been helpful in the analysis. A windowing technique, the sliding window method, made it possible to consider the observations in the previous instants of time for all the features, transforming multivariate time series into tables, thus making the forecasting problem a supervised learning problem such that it was possible to use regression algorithms in the task, like linear regression and Support Vector Regression. Three versions of this predictive system were also tested: 1) it was decided to combine each regression algorithm with the baseline, by using regression in non-stationary moments and the baseline model, that predicts the current value in the future, in stationary moments. In this setting, the variance threshold that allows to distinguish stationary windows from non-stationary ones was a parameter to consider; 2) another test was done by predicting the number of free slots in each station, rather than the number of bikes available; 3) it was considered the addition of a new feature to identify weekly patterns, which consists of the difference in the number of bikes in the previous week in the same time slot.
To evaluate model performances, R2 and RMSE metrics were compared with those of the baseline, exceeding it. Further comparisons were also made between the different versions of the prediction system with the aim of understanding the effects of changing the algorithm used, the variance threshold, the size of the sliding window and the forecast horizon.

Table of Contents

Capitolo 1

Introduction

The increasing interest in the fight against climate change and greenhouse gas emissions is bringing governments around the world to promote sustainable and green smart mobility systems in cities with the aim of reducing traffic, air and noise pollution levels and promoting a type of mobility accessible to all, thanks to business models inspired by sharing economy.

Among the main smart mobility solutions, in recent years Bike Sharing Systems (BSS) have become increasingly popular. Since the first Bike Sharing System was launched in Amsterdam in 1965 [1], this type of shared transport service has rapidly spread and has recently been present in major cities around the world.

Using of a shared bike encourages the use of public transport allowing a reduction of CO2 emissions and the improvement of public transport options, as well as solving the problem of the first and last mile by providing the missing link between bus stops or public transport subways and users' departure points and destinations.

Bike Sharing Systems are also advantageous from an economic point of view for users, since they offer a service at affordable prices and avoid the costs of owning and maintaining their own bike as well as the possible theft of their bike. Last but not least, cycling helps to maintain a healthy lifestyle.

On the other hand, bicycles are mainly suitable for short trips and people are more exposed to adverse weather conditions when cycling. Therefore, the use of bicycles strongly depends on the weather, but not only: it also depends on the topography of the place because people may have more or less difficulty in tackling with uphill, downhill or flat roads.

The Bike Sharing System consists in a bicycle rental service for transport within the city that provides bicycles at unattended stations. Users can borrow a bicycle from a dock at one station, and then return it to another station at the end of the ride. The stations, equipped with a number of slots for docking bikes, are scattered around the city in order to be easily accessible on foot and to ensure efficient coverage.

Certainly the location of the bike stations is one of the key factors contributing to the acceptance and success of a Bike Sharing System, together with the satisfaction of the demand for bicycles and free slots in each station of the circuit.

Indeed, a commonly observed problem in Bike Sharing Systems is the imbalances in the spatial distribution of bicycles in stations over time, due to the one-way use of bikes in travel and short rental times. From the user's point of view there are two situations that cause frustration: firstly the impossibility of finding a bicycle in the desired station to start the journey and secondly the impossibility of leaving the bicycle at the destination when the arrival station is full. When one of these situations happens, users who need free space or a bicycle have to choose between waiting at the station, going to another station or taking other means of transport, with the consequent dissatisfaction that results. Therefore, a BSS should in all ways prevent these conditions from occurring by adopting appropriate measures, aimed at counteracting the problem.

One solution could be to use predictive models to know in the short term the number of bikes available and free slots in each station of the system, for example in 20 minutes or an hour. Predictions, applied to the optimization of resources, could serve to improve the BSS and increase customer satisfaction, because they would allow users to be informed in advance about the best places to pick up or leave the bikes and at the same time could improve the redistribution of bikes from full to empty stations, which is usually done by trucks or trailers and is very expensive.

Thanks to the sensors placed in the station docks, the Bike Sharing Systems record cyclists' rental activities and are able to have an overview in real time of how many free bikes and slots are present in each station. Having a large amount of such measurements available in the past, it will be possible to provide future predictions, using machine learning algorithms.

The thesis work, developed in Python, focuses precisely on this topic: predicting the number of bikes in each station in certain future instants of time, from 20 minutes ahead to 60 minutes ahead, thanks to prediction algorithms. It is therefore a question of making predictions on spatial-temporal data in the context of bike sharing, by means of regression algorithms, such as linear regression and Support Vector Regression, which, thanks to a windowing technique, can be used for this purpose.

An overall view of this experimental thesis can be given by the description of the organization of the thesis document, which is structured in six chapters. Chapter 1 (*Introduction*) consists of this short introduction on bike sharing systems and thesis work. Chapter 2 (*Background and Related Work*) provides a general overview of time series analysis using machine learning, what a regression problem is and what are the most commonly used algorithms, as well as presenting the current state of research on Bike Sharing Systems applications. Chapter 3 (*The Barcelona use case*) explores the context in

which the analyzes are carried out, namely the BSS in Barcelona, and provides an exploratory data analysis of the dataset used. Chapter 4 (*Problem statement and proposed solution*) describes the problem to be addressed and presents the proposed solution to solve it, with a theoretical explanation of the algorithms and metrics used. Chapter 5 (*Experiments*) presents the experiments conducted on the dataset and the experimental results obtained by applying the proposed model as well as some of its variants. Finally, chapter 6 (*Conclusions*) presents the conclusions based on the experimental results, describes the limitations of the study and provides some suggestions that could help to make future improvements to the project.

Capitolo 2
Background and Related Work

The core concept of the thesis is to understand how to analyze and forecast spatial-temporal data using machine learning. This chapter provides some basic fundamentals to get the knowledge of time series analysis using machine learning, to figure out what is a regression problem and what are the most commonly used algorithms, as well as to present the current state of research on bike-sharing systems applications.

2.1 Background

Spatial-temporal data are multivariate time series, that means a set of two or more time-dependent variables [2], in which spatial dependencies exist between neighboring time series [3].
To better understand this definition one may have some basic knowledge of what a multivariate time series is, which is explained in the next paragraph.

2.1.1 Time series

A time series can be defined as "*an ordered sequence of values of a variable at equally spaced time intervals*" [4] or equivalently as "*a sequence of historical measurements of an observable variable at equal time intervals"* [5].
Times series are classified based on the number of variables into univariate and multivariate: a time series that contains values of a single variable is called univariate, when it includes values of more than one variable it is defined as multivariate [6].

One can classify time series also depending on the type of data; so there will be continuous or discrete time series. A discrete time series has a discrete set of values, which are measured at discrete points of time, at regular time intervals. Time intervals are also called the granularity of the time series and they can be annually, monthly, weekly, daily, hourly and so on. Some examples of discrete time series are annual sales, revenues, city population, as well as the number of bikes and the number of free slots under analysis.

Instead, continuous time series have observations measured at every instance of time, like in the case of temperature readings or in audio signals [6].

Time series can be decomposed into four components: trend, seasonal, cyclical and random components. The trend is an overall, persistent long-term movement and characterizes the overall time series. Seasonality component is repeated over time and consists into regular and periodic fluctuations, while the cyclical is the repeating of swings or movements in a longer time period. In addition the random component contains the residual fluctuations caused by unpredictable factors, that is the noise [6].

Two tasks can be addressed when dealing with timeseries: classification and forecasting. However, in this thesis we will just talk about forecasting, which is our goal.
Time series forecasting consists in forecasting the future having a knowledge of the past, so to use a model to predict future values based on previously observed values [7]. Then, to build a forecasting model we would like that two conditions are satisfied, i.e. that numerical information about the past is available and that it is reasonable to assume that some aspects of the past patterns will continue into the future [8].

Several methods are commonly used for time series forecasting: first statistical models such as ARIMA and exponential smoothing models and second machine learning and deep learning algorithms like regression algorithms (random forest regressor, SVR, linear regression, etc.) and recurrent neural networks.
When using machine learning algorithms, usually feature engineering and time windows are applied on time series, so that the time series forecasting problem is converted into a supervised machine learning problem, as discussed in the methodology chapter.

2.1.2 Regression task

The kind of supervised learning problem we addressed is called regression task, which is explained in this sub-section, first giving a definition of what a supervised learning problem is.
The task of supervised learning is to search through the space of possible hypoteses a function h that approximates the relationship f between some input features and an output target variable [9]. A training set of N input-output pairs is given as $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)$, where each target y_i has been generated by a function $y = f(x)$ which is unknown and we would like to approximate f by means of the hypothesis h. To evaluate the accuracy of a hypothesis we use a test set of samples, distinct from the training set. We would like to find a hypothesis that fits the training data well and at the same time it generalizes well on novel examples (the test set) [9].
We distinguish among two types of learning problems. When the target y is a categorical variable, so it can assume some distinct values referring to a category or a class, the task is called classification. When the target is numerical, the task is called regression [9].

Several machine learning algorithms are currently used for solving a regression problem. The most common are linear regression, polynomial regression, ridge and lasso regression, decision tree, random forest, support vector regression. In particular, some of them have been employed in machine learning analysis in the field of BSS, as reported in the related work paragraph. In this thesis, we focused our attention on linear regression and Support Vector Regression, which are presented in a details in the methodology section.

2.2 Related Work

Many efforts and studies have been carried out by the scientific community to improve the service of bike-sharing systems focusing on customer satisfaction. The main goals of researchers in the field of BSS are to find optimal locations for bike stations, to enhance bike redistribution among stations, to offer customers predictions in the next minutes about bike availability in the stations and to study mobility patterns in the cities of interest.

In literature many attempts for forecasting traffic flow are based on time-series models such as autoregression (AR), moving average (MA), autoregressive moving average (ARMA), and autoregressive integrated moving average (ARIMA) models and their variants [10, 11, 12, 13, 14, 15]. However, in the last twenty years, machine learning models have attracted attention and they have become serious competitors to classical statistical models in the forecasting community [5]. Spanning from machine learning algorithms like SVR [10, 16], random forest [17, 18], linear regression [16, 19], to a deep learning approach with the use of recurrent neural networks and LSTM [20, 18, 21, 22, 3], all these models have been employed as well to obtain predictions. In fact, recently, there have been several attempts to design deep learning models for time series forecasting problems [3].
Furthermore, many studies adopted some clustering techniques to group stations in order to find spatial patterns [14, 23, 24, 25, 3], made use of sliding window method for applying regression algorithms [26, 17, 27], proposed some dynamic network approaches [17, 25] to spatially characterize the system or they integrated information about the context such as weather conditions [24, 17, 22] or location factors [14, 24, 25]. Often these techniques are combined, resulting in hybridization and ensemble algorithms that are now becoming popular [27].

Many researchers in the BSS field found that spatial and temporal mobility patterns exist in bike-sharing data. In particular, relevant are the studies by Kaltenbrunner et al. [11] and Froehlich et al. [23] on the Barcelona bike sharing system. In details, Froehlich et al. individuated daily routines at lunch and dinner and observed that people tend to use the system more during the working days than in the weekend. They discovered that the spatial layout of the city influences the movement patterns and the spatial patterns differ from place to place. They clustered stations based on their behavior in some timeslots, to get

stations with similar temporal patterns and observed that close stations tend to behave similarly and stations at the margins of the bike-sharing network are the least active.

Zheng et al. [24] proposed a forecasting model to redistribute bikes among different stations in the New York Bike Sharing System. They considered influence factors like time, weather and station location and clustered neighboring station that have a similar pattern. Then on each cluster they used a multiple factor regression model with ARMA error to predict the number of bikes.

Vogel et al. [14] helped in design and management of BSS decisions by finding suitable location planning of bike stations. They discovered that there are spatial-temporal dependencies between rents and returns of bikes at stations.
Given this, they applied cluster analysis both on stations and on customers, finding groups stations with similar activity patterns and customer segments with different rental behavior. Clustering stations has been done exploiting some location factors derived from the surroundings, such as the access of stations to public transport, the population living in the area, if it was a housing or commercial area.

Zhou et al. [25] started clustering stations into groups according to their connectivity, geographical location, and transition pattern. Then to predict the traffic flow between stations, they proposed a temporal link prediction method based on the dynamic traffic flow network technique.

A dynamic network model was also proposed by Yang et al. [17], working on the Hangzhou BSS. They used a probabilistic spatio-temporal dynamic network model to simulate the spatiotemporal movements of bikes within the system, and estimate the number and time of check-in at different stations. Then they extracted some offline features like time factors (day of week, time of day, weekday and holiday) and meteorology conditions (such as humidity, visibility and wind speed) and online features (real-time bike availability) and structured them in a table with the sliding window method. So they applied a customized random forest algorithm to model and predict the users' check out behaviors. Combining check-in and check-out predictions they were capable to estimate the number of available bicycles at each station in a given future time period.

The sliding window method was also adopted by Hota et al. [27] that used it toghether with Weighted Moving Average as data preprocessing. The latter smoothes the time series and assigns a weight factor to each value in the time series data such that the highest weight is given to most recent data. They adopted a Radial Basis Function Network for providing forecasts.

On the other hand, Zhang et al. [19] employed a multiple linear regression model to predict the trip demand and the ratio of demand to supply at bike stations on data from Zhongshan's public bike system. They considered spatial correlations between neighbor stations by

means of a spatial regression model and evaluated the influence of built environment variables (accessibility, cycling infrastructure, public transport facilities, and land use characteristics) on the predictions. Separate models were created for weekdays and for weekends/holidays, and for morning and evening peak hours than for off-peak hours.
They found that both travel demand and demand/supply ratio in bike stations were positively influenced by population density, length of cycle lanes and secondary roads, and different types of land use near the station, and they were negatively affected by the distance from the city center and the number of other neighbor stations.

Several efforts to design deep learning models for BSS forecasting have been made, in particular implementing RNN, LSTM and some hybrid solutions with CNN.
Sardinha et al. [20] proposed a new recurrent neural network layering able to incorporate both historical and spatial sources of context for providing forecasts. Their architecture consists in a sequential composition of two components: the first is a LSTM that receives multivariate inputs with information about the context and returns the forecasted series as the output. The second component is another LSTM (or a gated recurrent unit GRU) that takes as input the forecasted series from the first LSTM and prospective sources of context along the horizon of prediction and provides the final forecasts.

Wang and Kim [18] focused on the short-term forecasting for bike station usage in the city of Suzhou and predicted the short-term available number of bikes in docking stations having one-month historical data. They found that the random forest algorithm was suitable for the problem, reaching good performances and being advantageous in terms of training time; the deep learning approach, implemented by means of LSTM and GRU models, has shown to perform better for long term predictions.

Another study on LSTM was carried out by Pan et al. [22] which trained a deep LSTM model with two layers to predict bike renting and returning based on historical data from New York BSS and meteorology data. They also considered the importance of several influence factors such as historical and future weather conditions, date and day of week.

Asadi and Regan [3] proposed a deep neural network model applied to the clusters of similar time series data, that contains a multi-kernel convolutional layer, designed to maintain the network structure, and extract short-term and spatial patterns, followed by a convolution-LSTM component to capture long-term trends, and a pretrained denoising autoencoder to have robust prediction in the existence of missing data.

A combination of a LSTM and a CNN was thought by Deng et al. [21]. CNN models has been adopted for traffic flow prediction, transforming the time series analysis problem into the task of image-like analysis by Deng et al in [21]. So, a CNN can learn spatial features and it can be combined with a LSTM component that learns the temporal features.

Capitolo 3

The Barcelona use case

3.1 Bicing, the bike sharing system in Barcelona

In this work, research and experiments were conducted on data of the Barcelona bike sharing system, Bicing, in 2008.
Bicing is an urban transport service limited to the metropolitan area of Barcelona which offers the possibility to rent bicycles to move around the city. Its equipment is manufactured and designed by Clear Channel Outdoor which is also the system provider and manages the bike sharing system in collaboration with B:SM, the public transport management company in Barcelona [28].

Figure 3-1 Bicing station and bikes in Barcelona [28].

In order to use the service, you must first register to the system by providing your personals and credit card data and by paying a fixed amount for the annual subscription, then any other costs are charged directly to the RFID card which allows you to rent bicycles. Each card has an RFID chip, a unique ID and number, linked to the corresponding registration information. To rent a bicycle, the user can remotely see which stations have bicycles and which are empty, via the Bicing portal [29], that shows the stations in a map of the city of Barcelona, with the number of free slots and used slots for each station.

At the station of interest, you swipe the contactless RFID card to be personally identified by the system, which then unlocks a bike from the dock. The bike can be returned by simply placing it in an empty slot at any Bicing station, when arrived at destination, then it is recognized automatically and is then locked into place[1] [30].
If no bikes to be rented are available at the station, the display will list other stations nearby that do have bikes available. If the station where the user wants to return a bike is full, he or she can touch the card on the pillar of that station and an extra 10 minutes will be credited, to give time to go to the next available parking point [28]. Unavailability of bikes and unavailability of free slots to return the bike, are two critical situations that may cause users unsatisfaction and should be avoided by the bike sharing system.

To monitor the availability of slots in each station, and also to control journey time and maintenance of bicycles, Bicing uses a simple mechanism. Each bicycle is identified by a unique number and the computer system records each time it is rented or returned to a station[2]. In fact, in each station, in addition to the physical locking mechanism, the support frame has a small passive RFID chip mounted to it, which communicates with the system when the bikes are removed and parked, to confirm the unique ID of the bike [28].
All this information is used to redistribute bikes among stations by means of specialized vans, to compute usage statistics, to repair damaged bikes and to make forecasts on the future distribution of bikes in the stations.

Figure 3-2 A specialized van moving bicycles from highly loaded stations to empty ones [31].

[1] This procedure to unlock a bike is the one adopted in 2008. Nowdays, things are slightly changed, as explained below.
[2] At the beginning they thought to insert a GPS in all bikes in order to immediately locate them at all times, but this was not put into practice because of the high cost [28].

Many Bicing stations are placed near point of interests in the city, like bus stations, historic buildings, hospitals, shopping centers, and so on. Each Bicing station can hold up to 20 bicycles, but some of them are double-installed, with at most 40 bikes [28].
Contrary to what one might think, Bicing is a service designed just for the citizens of Barcelona, not for tourism or recreational use, as there is a wide range of bicycle rental companies for any visitors to the city and Bicing was thought as a means of transport, complement to the city's conventional public transport [32].

The first 30 minutes of the rental are free and the subsequent 30 minute intervals have a price for a maximum of 2 hours. The use of bicycles beyond that time limit is discouraged with a penalty rate per hour; if a user continuously use the rented bicycle for more than 2 hours, this might result in the cancellation of membership [30]. As a consequence of this price policy that encourages users to make only short trips, only a small percentage of users borrows the bikes for more than 30 minutes [28].

3.1.1 Bicing in 2008 and today: a comparison

When its installation began in March 2007, Bicing was considered a new form of public bicycle transport in Barcelona. At the beginning (July 2007), the plan provided for the installation of 100 stations with 1500 bicycles available, however today the system has grown considerably thanks to the success of the platform. In the first two months, 30,000 users had already signed up, which was expected for the first year; this was favored by an effective promotion of the service and a city culture and by mature infrastructures for bike sharing [28].

A major expansion of Bicing happened in the summer of 2008, when over 150,000 users subscribed to the BSS, which was expanded to 6,000 bicycles and each bicycle was used on average almost 8 times a day [28].
Today it seems that the network of stations has grown compared to 2008: Bicing has 517 stations and the availability today of about 7000 bikes. However, the number of users dropped to 131,454 units in March 2021, and usage also dropped: on average, each bike is used about 6 times a day. [33]

The two main differences among Bicing in 2008 and today are the possibility to rent also electrical bicycles, the introduction of the app Smou and the mechanism to unlock bicycles.
In 2008 you could just rent mechanical bikes, while today you can rent both, electric bikes and mechanical bikes.

In the past you interacted with Bicing on the website, by means of their call centers and at the stations when you pick the bicycle. Nowadays, you can also use the app Smou to see how many electric and mechanical bikes are available in each station, to unlock a bicycle and to do reservations for mechanical bikes.

In order to reserve a bicycle, you perform the login on Smou, you select the station where you want to reserve the bike and you select the booking option. Once the booking has been accepted it cannot be canceled and you have 5 minutes to retrieve a mechanical bicycle at the selected station. Once at the station you can choose which bike to unlock, which was not possible in the past.
To unlock it is necessary to pass the card through the anchor of the chosen bicycle, without using the totem as before, or by scanning the QR code of the bicycle from the Smou app [32].

Another change is about the schedules. In 2008 the service was accessible seven days a week, 365 days a year and worked according to these schedules: from Sunday to Thursday the service works all day except from midnight to 5am: during this time users can only return bikes and not borrow them. On Fridays and Saturdays, the full service operates 24 hours a day [28]. Instead, nowadays Bicing works 24 hours a day, all days [32].

Obviously prices had increased over the years. During 2008, the annual subscription to Bicing was 24 €, then some usage fees were applied when using the bike for more than 30 minutes. In fact the first half an hour was free and to take the bike longer the cost was 0.30 € every 30 minutes. The maximum time you could keep a bike was 2 hours, after which there was an increasing penalty fine. After 24 hours a further 150 € fee was incurred if the bike was not returned.
Today, you have two rates to subscribe to the service: Flat Rate if you plan to make many uses during the year or Pay Per Use Rate (occasional) if you will use it less frequently [34].

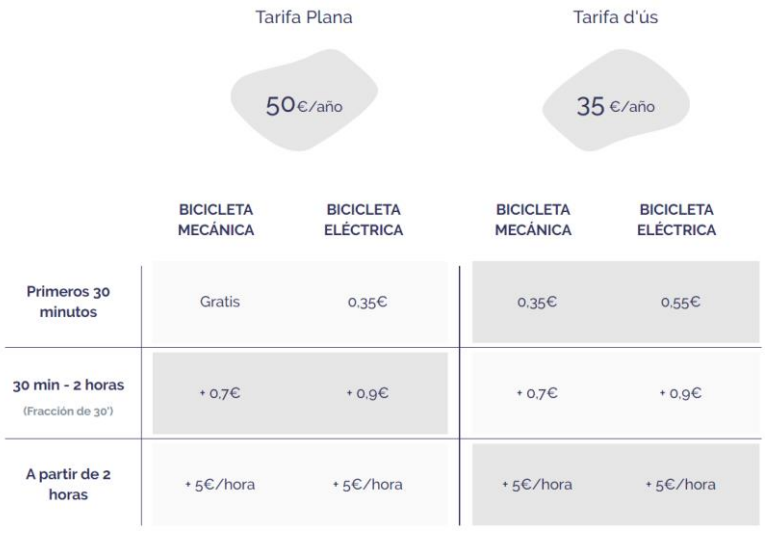


Figure 3-3 Prices of Bicing service in 2021 [34].

In the figure above, you can notice that prices are notably increased and they are differenced by type of rate and type of bicycle. With the Flat Rate, the annual subscription

is doubled with respect to the one in 2008. The first 30 minutes are free only for mechanical bikes and Flat Rate, while regarding the extra charges, they are tripled in the case of electric bicycles with respect to 2008.

3.2 Data exploration

The dataset consists into two comma-separated values files: *stations.csv* and *register.csv.*

- *Stations.csv* file contains 3301 rows and in each row has information about a bike sharing station as "id, longitude, latitude, name", where the *id* is the unique identifier of the station, *longitude* and *latitude* are the geographical coordinates of the station and *name* is the common name of the station. Some rows contain missing values: mainly some names are missing. Looking at the coordinates, you can see that the stations are in some important European cities: not only Barcelona, but also Paris, Rome, Zaragoza and so on.
- *Register.csv* file has 25319028 entries, all referring to stations in Barcelona. Each row contains information about a measurement in a station in a certain instant of time and it is formatted as "station, timestamp, used_slots, free_slots". *Station* is the id that identifies the station, *timestamp* is the time at which the measurement was done and it is expressed in the format YYYY-MM-DD hh:mm:ss, *used_slots* is the number of available bicycles and *free_slots* is the number of available slots in the station.

In order to take only the stations of Barcelona, a filter was applied on stations in s*tations.csv* file such that only those within a distance of 50 km from the coordinates of Barcelona (41.3825°, 2.176944° [35]) were considered. The distance was computed as the Haversine distance, which gives the great-circle[3] distance between two points on a sphere given their longitudes and latitudes[4] [37], that is the shortest distance between two points on the surface of a sphere [39].

After this filter, you get 776 stations without missing values, but you have only 290 distinct names for the stations. Maybe some names are repeated by mistake, some stations have the same name also in real life or there are some repetitions for the stations (also the

[3] A great circle of a sphere is "*the intersection of the sphere and a plane that passes through the center of the sphere*" [36]. Any diameter of any great circle corresponds to the diameter of the sphere, so all great circles have the same center and circumference [36].

[4] It is computed as $D(x,y) = 2\arcsin\left[\sqrt{\sin^2((x1-y1)/2) + \cos(x1)\cos(y1)\sin^2((x2-y2)/2)}\right]$, where x and y are the two points between which you want to calculate the Haversine distance, x1 is the latitude of the first point and x2 is the longitude of the first point. Then, to get the distance expressed in kilometers, it is necessary to multiply by the mean Earth radius R (6371 km) [38]. Applying the Haversine formula to the Earth is an approximation because the Earth is not exactly a sphere: the Earth radius R varies from 6356.752 km at the poles to 6378.137 km at the equator [37].

coordinates are repeated). Out of curiosity I tried to investigate the stations called Pg LluÃs Companys[5]: they have the same name, but different ids. So, I looked for the ids on the map of availability on the Bicing portal [29] and I found that some stations have different names actually and that the repeated coordinates refer to stations which do not exist.

Id	Longitude	Latitude	Exists	Actual name
5	2.180214	41.391072	Yes	PG. LLUIS COMPANYS, 11 (ARC TRIOMF)
6	2.180508	41.391272	Yes	PG. LLUIS COMPANYS, 18 (ARC TRIOMF)
7	2.183183	41.388867	Yes	PG. PUJADES, 1 (JUTJATS)
1438	2.180214	41.391072	No	-
1439	2.180508	41.391272	No	-
1440	2.183183	41.388867	No	-

Table 3.1 Comparison of different stations with the same name.

However, to get the definitive list of stations to be considered in the analysis, I joined the stations of Barcelona already filtered with the *register.csv* data . Finally, I got 284 stations: they were the first ones, with id from 1 to 284, and they are the ones shown in the map below.

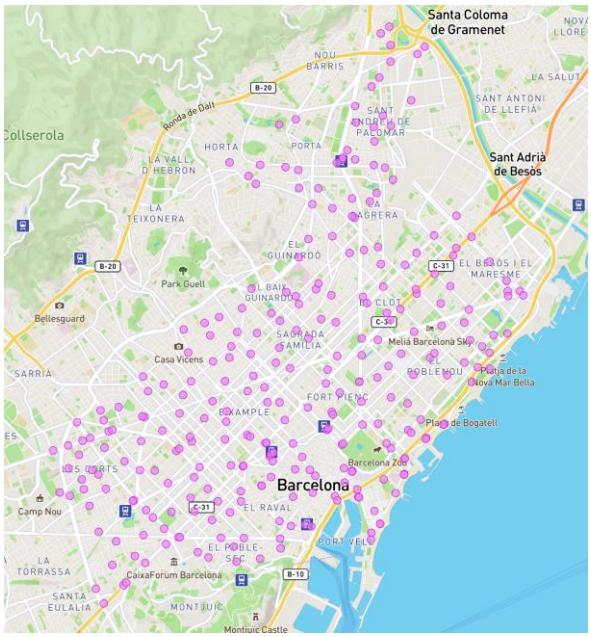


Figure 3-4 Map of the stations considered in the analysis, obtained through https://studio.mapbox.com/ website.

[5] The actual name is Pg. Lluis Companys, but in the dataset it is misspelled as Pg LluÃs Companys. Also the names of other stations are misspelled because of some problems in the encoding: some special Spanish characters are not correctly displayed in UTF-8 encoding.

Regarding the register of measurements in *register.csv*, it has been saved in a dataframe. It was noticed that timestamps go from 2008-05-15 12:01:00 to 2008-09-30 23:58:00, so we are considering the period from May 2008 to September 2008, and that are separated by 2 minutes. So, the data granularity, or the time interval between the data rows is every 2 minutes.

	station	timestamp	used_slots	free_slots
0	1	2008-05-15 12:01:00	0	18
1	1	2008-05-15 12:02:00	0	18
2	1	2008-05-15 12:04:00	0	18
3	1	2008-05-15 12:06:00	0	18
4	1	2008-05-15 12:08:00	0	18

Figure 3-5 First five measurements rows in the register dataframe.

You can see from the picture above that the first timestamp (2008-05-15 12:01:00) probably is a mistake because all the other timestamps are taken in even minutes, like 12:04, 12:06, 12:08 and so on. Moreover, if measurements are taken every 2 minutes and timestamps go from 2008-05-15 12:01:00 to 2008-09-30 23:58:00, you should have 28320196 measurements in total, but you have just 25319028. This big difference is due to the fact that there are many missing values in the time series and mainly many missing values are in the night, from midnight to 5 a.m.[6]
So, in order to have equally spaced timestamps, I created a complete range of timestamps from 2008-05-15 12:02:00 to 2008-09-30 23:58:00 where every timestamp is every 2 minutes. This range was then repeated for each station, since every station is independent from the others. Finally, a new big register was created so that for each station you have the complete range of timestamps, and the index of this dataframe is the column of timestamps such that time instants could be easily ordered in temporal order.

Considering the columns of the register, the number of available bicycles in a station (used_slots) and the number of available slots (free_slots) may not be enough to do a complete analysis of the stations behavior and to provide accurate forecasts, so two other features were added.
First, I computed a column called total_slots, which has been obtained as the sum of used_slots and free_slots. Total_slots could be useful because the stations can have a different number of slots in total and the dimensions of a station may have an impact on its behavior. Second, also the percentage of used slots has been added in order to have an idea about the occupancy of a station in each time instant, independently from the total number of slots, that is to have not only an absolute measure like used_slots, but also a relative one.

[6] Until the 1 July 2008 there is no data from midnight to 5 a.m., while from the 2 July to the 30 September 2008 we have measurements 24h/24, so even in the night.

To have a visual idea about the dataset, some plots can be shown to see the trend of the features, like pictured in the figure below.

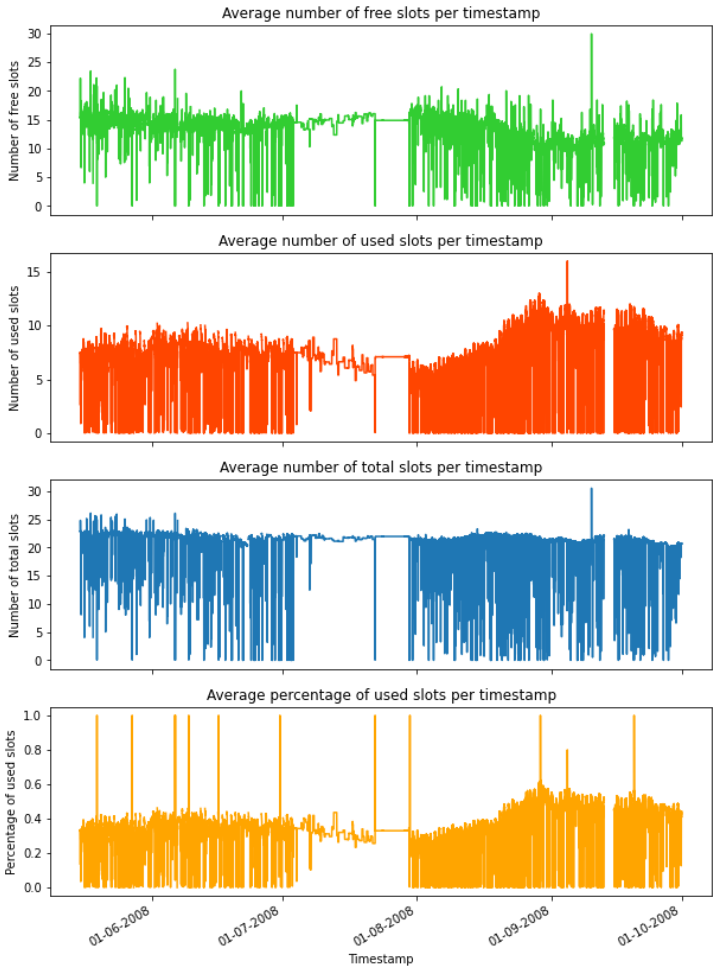


Figure 3-6 Average number of free/used/total slots per timestamp and average percentage of used slots per timestamp. The average is computed across all the stations.

One can spot that in some timestamp total slots is 0, which cannot be possible in practice: it would mean that on average the stations have no slots. Another strange fact is that there are extreme values, in particular looking at the percentage of used slots: it is weird that on average the percentage is 1, because it would mean that all stations are full all together in those time instants. Then, it is noticeable that in August and September the average number of used slots, as well as its percentage, increase. Moreover, in the middle of September there is a period with many missing values one after the other, that in the plot is represented as an interruption of the colored lines. However, what is still clear from the graph is an uncommon behavior in the month of July, which should be better investigated.

It seems that the trend in July is different from the one in the other months, so, for understanding it, I plotted the number of used, free and total slots and the percentage of used slots for one random station separately for the months of June and July and I made some comparisons.

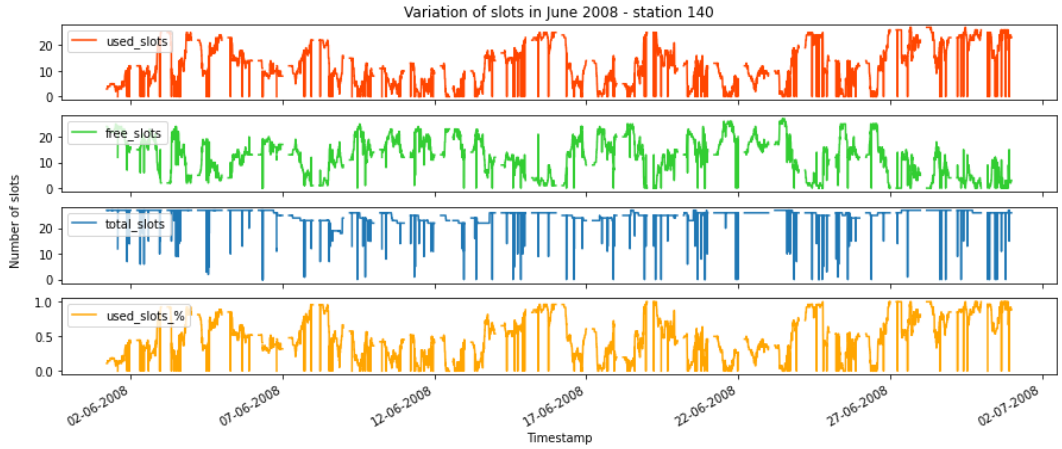


Figure 3-7 Variations of used/free/total slots/percentage of used slots for station 140 in June

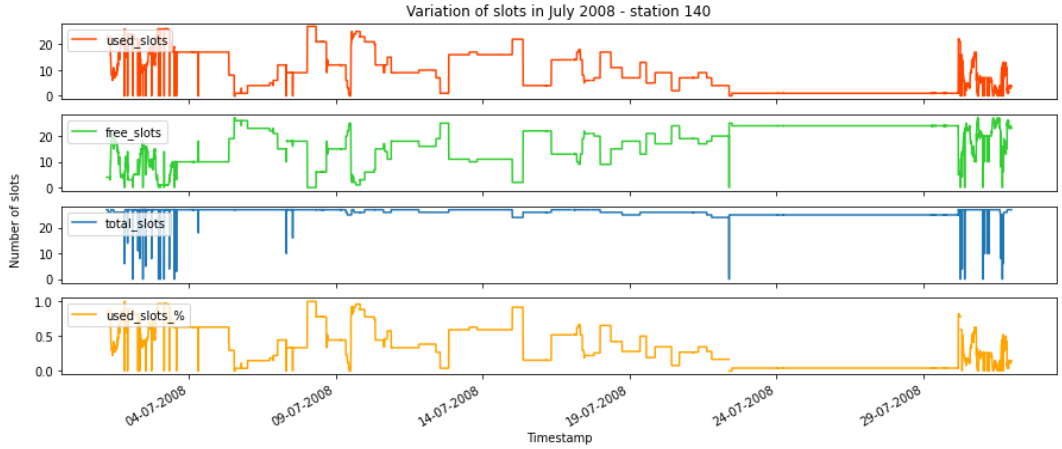


Figure 3-8 Variations of used/free/total slots/percentage of used slots for station 140 in July

The time series in June have more variations and more missing values than in July: in June they are discontinuous and swinging. Instead, in July there are not much variations and for some periods the measures are constant, in particular for the last week of month.
Since this behavior is similar also for other stations, one can suppose that in July 2008 the system was not working properly, maybe they were doing some maintenance. Actually, after investigating on the Internet, I interestingly found on a Bicing blog that the bike sharing service was working but the information provided on bicing.com has actually been erroneous since around 03/07/2008 because the bicing.com map was broken. Updates were

occasionally made, but the data was mostly static. The map was displaying the same information about the stations since 22/07/2008 at 09:30. Fortunately, it was back to work on 31 July 2008 [40].

3.2.1 Feature plots

Histograms and box plots may help in looking at the distribution of the data, so for each feature I plotted these two types of graphs, as shown in figures 3-9 and 3-10.

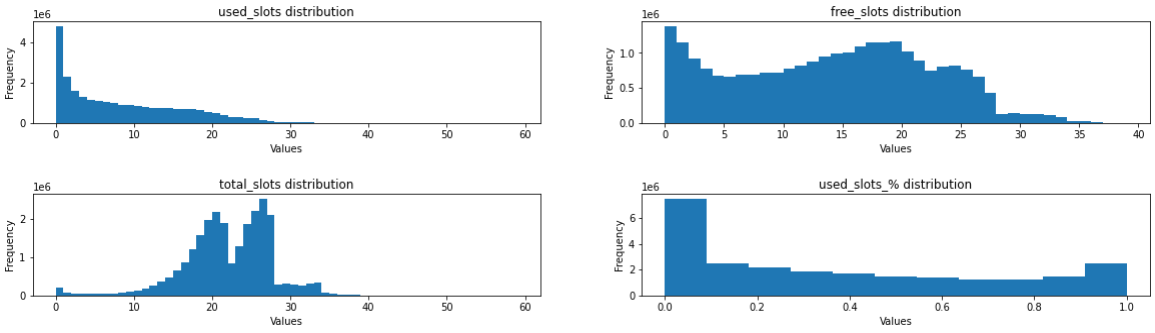


Figure 3-9 Histograms for used, free and total slots and for percentage of used slots. Each bin correspond to one possible value for discrete features (used, free and total slots). For the percentage of used slots, which is continuous, the range of possible values [0,1] was divided into 11 bins.

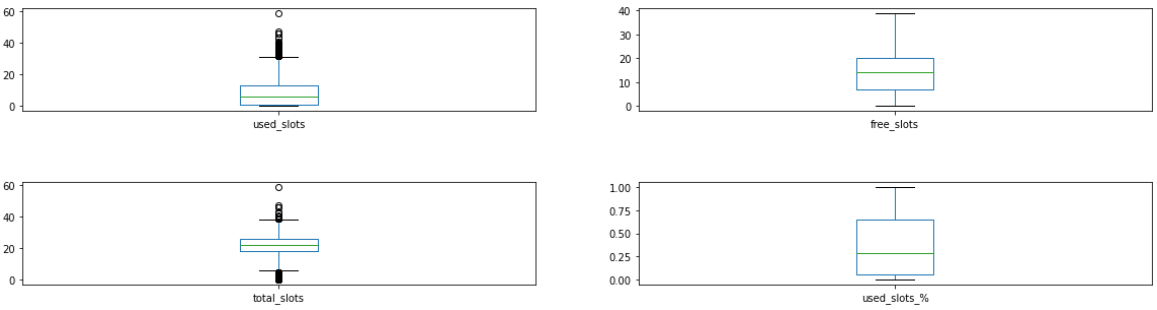


Figure 3-10 Box plots for used, free and total slots and for percentage of used slots.

The distribution of used slots has a shape that reminds an exponential, with mode in 0, which can be spotted in the histogram. Regarding used slots, there are some outliers, visible as points in the first box plot; values greater than 40 are clearly unfeasible since even the double-installed station can hold up to 40 bikes.
This is true also for extreme values of total slots: they are considered outliers. From the box plot, it is notable that values below 10 for total slots may be considered outliers, however it is weird that the median is 20 and there are many values below 20: in theory stations have

20 or 40 slots, so probably some sensors in the stations are out of service. Total slots has some values that are 0: these rows should be removed because it cannot be possible: probably the station was offline in that moment, or all its sensors did not work in that moment or for other reasons. The two peaks in total slots distribution are probably due to the fact that stations can be double-installed with 40 slots or just installed with 20 slots.
Free slots histogram has many 0 values (the mode is 0) but also other values are frequent (1 and between 15 and 20). There are no values exceeding 40, the range of values is from 0 to 40, as you can see in the box plot.
Used slots percentage distribution reflects a bit the used slots distribution since you have a peak on 0, but also a percentage equal to 1 is frequent (the second most frequent percentage). The range of values from 0 to 1 is correct, since it is a percentage and it is observable that the median is close to 0.25.

Capitolo 4

Problem statement and proposed solution

The aim of the work is to solve a forecasting problem for spatial-temporal data, focusing on the application of bike sharing systems.

In few words, the issue to be addressed consists in providing h-step-ahead one-shot forecasts of the number of bikes available in each station in a certain future time instant. It means that at time t, one would predict the number of bikes at time t+h, where h is the forecast horizon and the number of bikes for time t+h is given as a point forecast.

A crux of the problem is the type of data we are working with. Spatial-temporal data are multivariate time series, that means a set of two or more time-dependent variables [2], in which spatial dependencies exist between neighboring time series [3]. Multivariate time series *per se* is a type of data which it is difficult to work with, then adding spatial dependencies makes the problem much harder. So, the problem becomes challenging due to unique spatial relations among neighboring time series, short-term and long-term temporal patterns [3].

A brief overview of the proposed solution is simply explained below and pictured in this chart. In the next paragraphs more detailed explanations will be given.

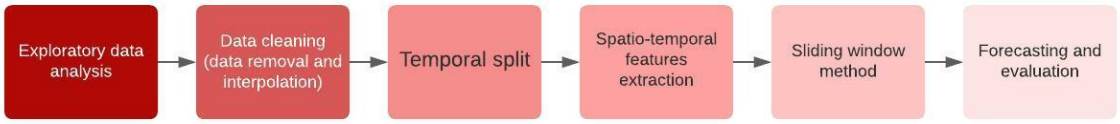


Figure 4-1 Proposed solution: the principal steps.

We would like to predict the number of bikes available in each station for some time in the future, exploiting the past measurements.

First you need to take a look at the dataset provided to know which data we are working with. Since **exploratory data analysis** has shown that data is indeed dirty, **data cleaning** operations serve to remove noise, such as removing anomalous observations and filling in missing values with data interpolation. So the dataset can be **split** into two parts: the first and most consistent will be the training set, which contains the first observations in chronological order and is used to teach the forecast model, and the test set, which contains the latest data and serves to make predictions.
Predictions can be made by exploiting information on the number of bikes available in the past, as well as the number of total and free slots in a station in the past and some other **features extracted** to describe the spatial and temporal relationships between neighbor stations in the past. All this information is structured as a table, in which the rows are sorted by time and each row contains the last w past values of each necessary information, where w is the size of a time window that indicates how much in the past one looks to predict in the future. We can also choose for what time in the future the forecasts will be prepared, i.e. what the forecast horizon will be for the predictions. Data transformation into table format is performed with the so-called **sliding window method** and is performed separately for the train and the test sets.
Finally, **forecasting** is made by some regression algorithms, linear regression and Support Vector Regression, which are fitted on the training table described above. The predicted numbers of bikes are compared with the actual values in the test table by calculating some performance metrics (the coefficient of determination and the Root Mean Squared Error), which help to understand the goodness of the predictions.

4.1 Data cleaning

As evidenced by the data exploration, the dataset contains dirty and noisy data. Time series are notoriously hard to clean and here there is also a complication due to the spatial nature of bike sharing data. In general, you can follow the same sort of rules you would in regular data analysis, except that you always need to be aware of the spatio-temporal nature of the data. Also outliers can be difficult to find in this context, so a characterization of the context is needed [41]. In particular some problems appeared in the dataset that should be mitigated:

- Measurements in July 2008 cannot be trusted since the Bicing map was broken in that period. Even if the out of service period was from the 3rd to the 31th July and in the first two days of month the system worked properly, it has been decided not to trust also the first two days of the month.

- There are sudden and large variations in the number of available bicycles (used slots) and in the number of total slots between subsequent timestamps. It is very unlikely that in 2 minutes there could be a change of 10 or 20 bikes, mostly for the total slots. However, it was noticed that there are sudden drops to zero for used and

total slots. These drops could be linked to a malfunction of the map because for example it was found a drop on used slots, which becomes equal to 0, on 2008-06-13 at 07:54:00 in 227 stations out of 284. It is strange that all sensors in 227 stations broke in the same moment, however it cannot be excluded that in other cases the drops are due to sensor malfunctions, or in other cases even to the movement of bikes with trucks from highly loaded stations to empty ones.

- In some cases, the total number of slots is outside the limits imposed by the domain: total slots must be between 1 and 40, but we have seen that this is not always the case since there are zeros or values greater than 40. However, even values well below 20 are quite strange, because stations can be double-installed with 40 slots or installed with only 20 slots.

- The schedules pose the problem of whether or not to consider the night in the analysis. In fact, from Sunday to Thursday from midnight to 5am, bicycles can only be returned and not borrowed, while on Fridays and Saturdays, the full service operates 24 hours a day, so even in the night. Despite that, until the 1 July 2008 there is no data from midnight to 5 a.m. in the register.

- There is one station for which the number of available bikes does not vary over time, as well as the percentage of used slots. Furthermore, for some stations there are many missing values, with respect to the majority of stations[7].

- Since the beginning, there was a conspicuous number of missing measurements, considered as a whole and not with respect to the single station.

Given these issues, it is evident that data is really dirty and some data removal and interpolation techniques had to be applied, which are described in the following paragraphs.

4.1.1 Data removal

The data removal steps, that try to reduce the problems present, are listed here in the order in which they were done.

- First, I removed the stations without variations over time. In this case, there was just one: station 221. For it, there were no changes in used slots and in the percentage of used slots.
- Then, I tried to remove the big consecutive variations. To do that, I deleted rows for which the station was completely full and then empty in the subsequent timeslot, and vice versa (before empty and then full). This was done looking at the percentage of used slots: for instance, if in two consecutive rows you have 0,1 or 1,0 as

[7] Actually, all the stations have Nan values because in some moments there is no data for the whole system (e.g. for a few days in September).

percentage of used slots, you delete that rows. To enforce the removal of big changes, another condition could be satisfied in order to delete the variations. I also delete the rows for which there is a change in the number of used slots and total slots greater than threshold (3) bikes, between consecutive timestamps.

- After that, I deleted rows for which used slots or total slots was greater than 40 or smaller than 10. The latter threshold was suggested by an article [42] where they worked on a very similar dataset[8] and deleted all the elements of the timeseries where the total number of slots in the station was below a certain threshold, that is 10 [42].
- Next, I recreated a dataset with all timestamps, by merging the existing one with the complete range of timeslots (one timeslot every 2 minutes). This step was done because you deleted some rows and you need to interpolate missing values, so you need to have them as missing. If you do not perform the re-adding of the missing timeslots you cannot know which measurements are then missing. The two following steps are not influencing the interpolation, so even if you are cancelling some other rows, it is not necessary to repeat this step.
- Then, since measurements in the night are missing in May and June and for this reason an analysis of the behavior in the night in the weekend was not possible, it has been decided to cut all the rows whose timestamp was from 00:00 to 4:58 a.m.
- Finally, the stations with many missing values were deleted. To do this, the rows related to the stations with a number of Nan greater than 0.15 * (number of measurements stations should have) were cancelled. In the end 272 out of 284 stations remained.

4.1.2 Interpolation

When missing values cause mistakes, at least two ways exist to deal with the problem. We may just consider the portion of data after the last missing value for making our predictions, assuming that there are enough observations to produce meaningful forecasts. As an alternative, we may replace missing values with estimates [8]. A combination of both was implemented in this thesis.
Missing values are present in all the features, that is used, free, total slots and percentage of used slots. When a row is missing, it means that all the four measurements together are missing, but overall the number of missing values in the percentage of used slots is greater than those of the other features because there are divisions by 0, when total slots is 0. Since total slots and the percentage depend on used and free slots, I interpolated only used and free slots, separately, and derive the others by recomputing them later.
It is worthy pointing out that all interpolation operations were done separately for each station, to avoid issues related to duplicated timestamps and overlaps.

[8] The dataset contained the measurements of used and free slots for Bicing in 2008.

Then, some assumptions have been made to simplify the problem and to use some unusual interpolation methods as well.
First it was supposed to fill only short sequences of null values: a threshold was set on the admissible number of consecutive Nans which was 2. In practice, sequences of 3 or more consecutive Nans are not interpolated, but are left as missing and discarded later. This is linked to the fact that we want to avoid too much uncertainty in the interpolation; in fact, having two unknown values in consecutive timeslots causes a lack of information for a maximum of 6 minutes, since the known values before and after the Nans have a distance of 6 minutes. Taking a higher threshold would mean increasing the uncertainty range and also decreasing the goodness of the interpolation. In the end, considering 6 minutes as an uncertainty interval was considered acceptable, rather than 8 or 10 minutes, which was too much, while with 4 minutes of uncertainty, sequences of a single missing consecutive value interpolated, which is too stringent.
Second, it is expected that in consecutive timeslots, or better in 2 minutes, the number of bikes in a station does not vary too much: there would likely be some changes of 1 or 2 bicycles. Possible sudden changes probably due to bike relocations with trucks were ignored and likely previously deleted with the data removal.
Another consideration is about which data points to use in the interpolation. To simplify computations, it was decided to take the value before and the value after the sequence of missing values. To do this, the timeseries is divided in blocks depending on the presence of null values. A block can contain only not null values or only null values. Then, for the blocks with nulls it is checked the condition if the size of the block is smaller than 2, because you want not to interpolate long sequences of Nans. So, you keep only sequences with few nulls and for them you construct a small dataframe that has the sequence of nulls preceded by the value before the nulls and followed by the value after the nulls. Since this might not be immediately clear, I am providing an example to better explain the method.

timestamp	station	used_slots	free_slots	total_slots	used_slots_%
2008-05-15 12:02:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:04:00	1	2	4	6	0.333333
2008-05-15 12:06:00	1	4	1	5	0.8
2008-05-15 12:08:00	1	5	2	7	0.714286
2008-05-15 12:10:00	1	6	3	9	0.666667
2008-05-15 12:12:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:14:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:16:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:02:00	2	3	3	6	0.5
2008-05-15 12:04:00	2	3	4	7	0.428571
2008-05-15 12:06:00	2	5	6	11	0.454545
2008-05-15 12:08:00	2	NaN	NaN	NaN	NaN
2008-05-15 12:10:00	2	NaN	NaN	NaN	NaN
2008-05-15 12:12:00	2	3	2	5	0.6
2008-05-15 12:14:00	2	NaN	NaN	NaN	NaN
2008-05-15 12:16:00	2	2	4	6	0.333333

Figure 4-2 Example of a small version of the dataset before the interpolation.

In the figure above, there is a small version of the dataset before the interpolation. At a first sight, you see that when a row has a missing value, all the four features are null. Suppose you want to interpolate the first column, that is used slots. First, you consider each station separately. It is observable that for the first station there are two sequences of missing values: the first contains one Nan and it is at 12:02, the second has 3 consecutive Nans. These sequences will not be interpolated because the first is at the beginning of the timeseries, so it is not possible to take the previous not null value in order to do the interpolation, then the second sequence has 3 consecutive null values, so it is discarded because of too many consecutive missing values. For the second station, the two sequences of Nans will be interpolated, since they have 2 and 1 consecutive missing values and no other particular conditions occur. After that, you construct a dataframe that will contain the values for doing the interpolation: for instance, from the sequence at 12:08 and 12:10 the dataframe will be:

Timeslot	Used_slot
2008-05-15 12:06:00	5
2008-05-15 12:08:00	Nan
2008-05-15 12:10:00	Nan
2008-05-15 12:12:00	3

Table 4.1 Example of dataframe obtained from a sequence of null values.

In this case, the first and the last data point of the dataframe will be those between which interpolation is done.

Three types of interpolation were tried, before deciding which one fitted best on the data: linear, mean and nearest interpolation. For each of them, I reported a brief explanation and the application on the previous example, as shown in figures 4-3, 4-4 and 4-5.

- Linear interpolation is "*a method of curve fitting using linear polynomials to construct new data points within the range of a discrete set of known data points*" [43]. In this case, the interpolation was done between two known points given by the coordinates. Having two known points as (x_0, y_0) and (x_1, y_1), the linear interpolant is the straight line between these points [43]. In this way, you find the interpolated value of y by taking it on the segment in correspondence of the x you want. The analytical formula is the following: $y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}$
  Since we are interpolating a discrete quantity, it was necessary to round the float number obtained to the smallest integer.
  This method works quite well only if the interval on the x-axis is small, since we are supposing that the number of used slots and free slots have little changes in subsequent timeslots. If on x-axis we have the time, discretized in timeslots which are every 2 minutes, and we consider a small period of time, we are supposing that the values y0 and y1 are similar and not too far from each other.

timestamp	station	used_slots	free_slots	total_slots	used_slots_%
2008-05-15 12:02:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:04:00	1	2	4	6	0.333333
2008-05-15 12:06:00	1	4	1	5	0.8
2008-05-15 12:08:00	1	5	2	7	0.714286
2008-05-15 12:10:00	1	6	3	9	0.666667
2008-05-15 12:12:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:14:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:16:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:02:00	2	3	3	6	0.5
2008-05-15 12:04:00	2	3	4	7	0.428571
2008-05-15 12:06:00	2	5	6	11	0.454545
2008-05-15 12:08:00	2	4	4	8	0.5
2008-05-15 12:10:00	2	3	3	6	0.5
2008-05-15 12:12:00	2	3	2	5	0.6
2008-05-15 12:14:00	2	2	3	5	0.4
2008-05-15 12:16:00	2	2	4	6	0.333333

Figure 4-3 Example of linear interpolation on a small version of the dataset.

- Mean interpolation consists in computing the arithmetic mean between the ordinates of the points. Even in this case the interpolation was done between two points, with coordinates (x_0, y_0) and (x_1, y_1). All the missing values between x0 and x1 are filled with the same value y, which is computed as $y = \frac{y_1 + y_0}{2}$.
  Since we are interpolating a discrete quantity, it was necessary to round the float number obtained to the smallest integer.
  In the end, you get a sort of step function if the two points have different ordinates, or a constant function if the points have the same value for the y.

timestamp	station	used_slots	free_slots	total_slots	used_slots_%
2008-05-15 12:02:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:04:00	1	2	4	6	0.333333
2008-05-15 12:06:00	1	4	1	5	0.8
2008-05-15 12:08:00	1	5	2	7	0.714286
2008-05-15 12:10:00	1	6	3	9	0.666667
2008-05-15 12:12:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:14:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:16:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:02:00	2	3	3	6	0.5
2008-05-15 12:04:00	2	3	4	7	0.428571
2008-05-15 12:06:00	2	5	6	11	0.454545
2008-05-15 12:08:00	2	4	4	8	0.5
2008-05-15 12:10:00	2	4	4	8	0.5
2008-05-15 12:12:00	2	3	2	5	0.6
2008-05-15 12:14:00	2	2	3	5	0.4
2008-05-15 12:16:00	2	2	4	6	0.333333

Figure 4-4 Example of mean interpolation on a small version of the dataset.

- Nearest interpolation simply takes the not null value that precedes the sequence of Nan values and assigns it to all the missing values of the sequence you are considering. Again, it is based on the assumption that variations in a little time should not be too noticeable. The result is a constant trend for the interpolated measurements.

timestamp	station	used_slots	free_slots	total_slots	used_slots_%
2008-05-15 12:02:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:04:00	1	2	4	6	0.333333
2008-05-15 12:06:00	1	4	1	5	0.8
2008-05-15 12:08:00	1	5	2	7	0.714286
2008-05-15 12:10:00	1	6	3	9	0.666667
2008-05-15 12:12:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:14:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:16:00	1	NaN	NaN	NaN	NaN
2008-05-15 12:02:00	2	3	3	6	0.5
2008-05-15 12:04:00	2	3	4	7	0.428571
2008-05-15 12:06:00	2	5	6	11	0.454545
2008-05-15 12:08:00	2	5	6	11	0.454545
2008-05-15 12:10:00	2	5	6	11	0.454545
2008-05-15 12:12:00	2	3	2	5	0.6
2008-05-15 12:14:00	2	3	2	5	0.6
2008-05-15 12:16:00	2	2	4	6	0.333333

Figure 4-5 Example of nearest interpolation on a small version of the dataset.

After interpolation, the dataset still contains 16745 missing values, but previously they were 966,093. It was observed that the three alternatives, considering the average over all stations, worked in a similar way. However, the mean interpolation and the nearest interpolation seemed to give better results than linear interpolation. This was assessed by qualitatively observing the time series of some stations over a few full days. With linear interpolation some small abrupt changes in subsequent timestamps remained particularly in the used slots and total slots, while with mean and nearest interpolation abrupt changes were avoided. Between the two, the nearest interpolation was preferred as it avoids the step behavior of mean interpolation and it is not linked to inaccuracies in the approximation of the decimal values.

In the graphs below, the situation before data cleaning is compared to the situation after data removal and interpolation. It can be seen that the noise has been reduced, the peaks have been blunted, the fluctuations and swings have been attenuated.

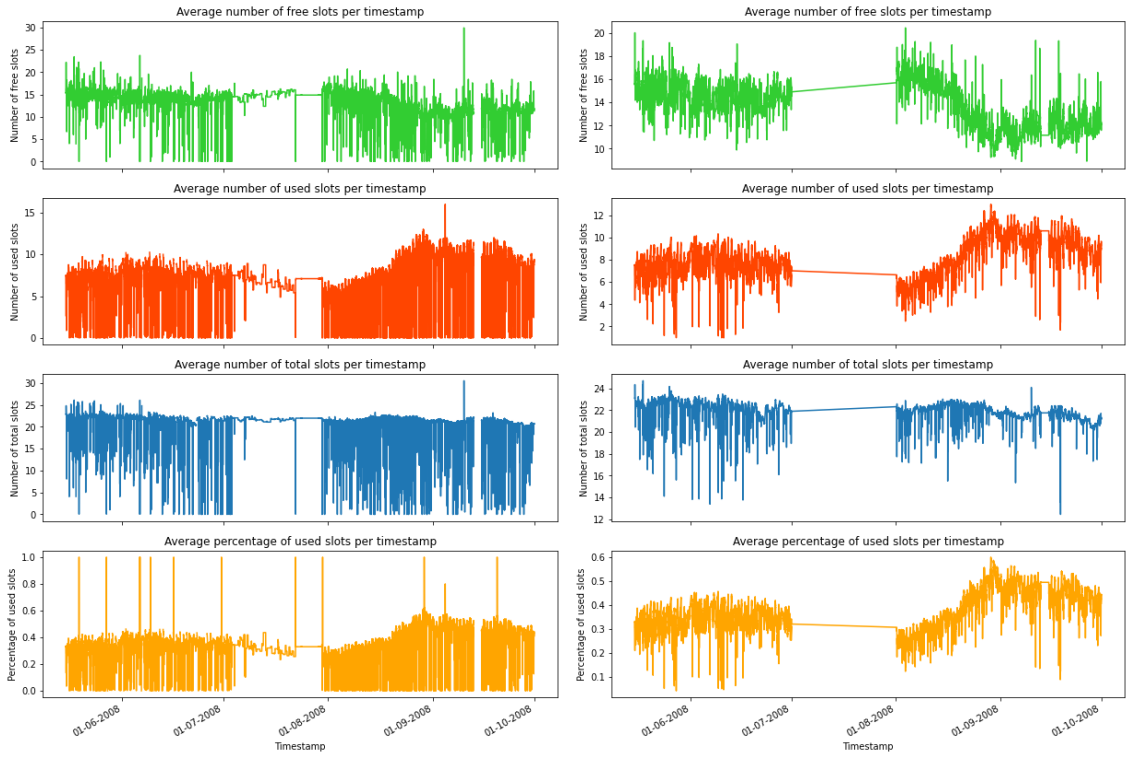


Figure 4-6 Average across all stations of free, used, total slots and the percentage of used slots before data cleaning (on the right) and after data cleaning (on the left), in the period May-September 2008.

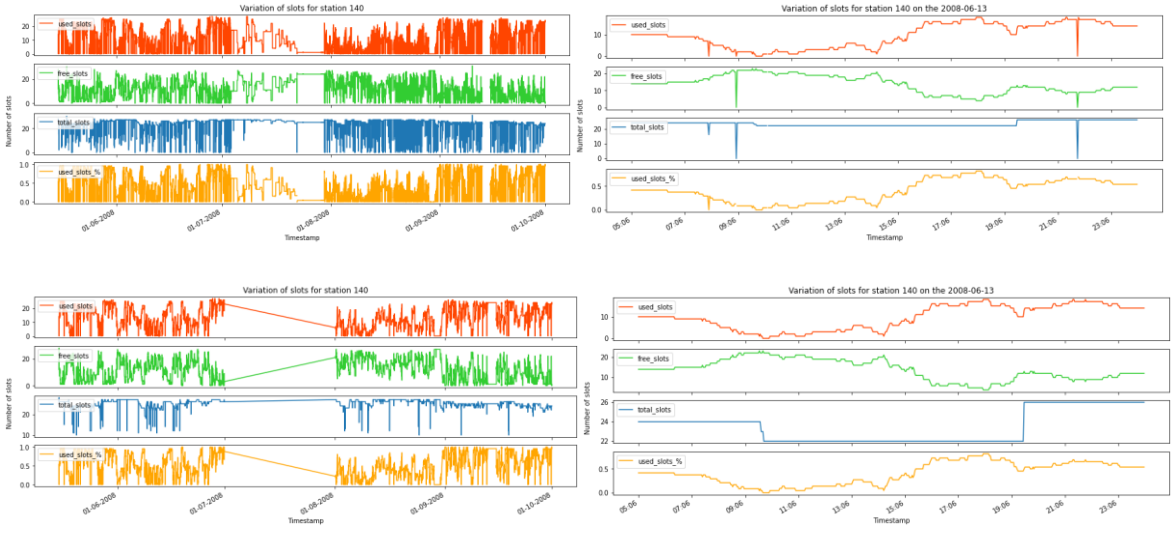


Figure 4-7 Variations of free, used, total slots and the percentage of used slots for station 140 in the period May-September 2008 (first column) and during the 2008-06-13 (second column); on the top there is the situation before the data cleaning, below the situation after.

4.2 Temporal split

When dealing with temporal data, the classical method of splitting the dataset randomly into train and test subsets does not work anymore. Splitting data randomly would cause to lose the temporal order in the time series and the temporal dependency between observations. We do not want to look at the future when training our model, but we would like to train on the past observations to predict the future.
For these reasons, a temporal split is needed, which divides the dataset into parts maintaining the temporal order. The first observations will belong to the training set and the last to the test set. This split can be performed using the temporal_train_test_split method by sktime library, a useful tool for modeling time series in Python.
It is important that the split is performed before feature extraction and sliding window method, since these operations involve the creation of time windows: if the dataset would not had been split before applying windowing, we would have had some overlaps between test and train set, in the test set there would have been some windows created using data in the train set as well.

4.3 Space-temporal extra features

In this work, an independent model has been created for each station, but it is evident that stations are not independent to each other. To avoid this discrepancy, some space-temporal features were added to the existing ones; they would help to relate stations to each other, in particular for neighbor stations, and also to remark the distinction of bike use in holidays and working days. As a further consideration, engineering features is also suggested by a tutorial by Google [44] that gives the best practices for creating tabular training data for forecasting: they say that for some raw data, you can improve model quality by engineering features [44].

4.3.1 Finding neighbor stations

First of all, for each station its neighbors were found using their coordinates. Using the latitude and longitude of the station pairs, the Haversine distance was calculated for each station pair. Nearby stations are those within a radius of distance. Hence, only stations with a distance of less than a threshold, which was fixed to 500m, were considered to be close. The result is a dictionary that contains the relationships between the stations, where the key is the id of a station and the value is a list with the ids of its neighbors. Potentially there can be stations with an empty list, i.e. without neighbors, or the same station can appear in the list of neighbors of many stations, of course. In the example of table 4.1, the former situation is represented by station 3, instead the latter is the case of station 4, which is repeated in the list of neighbors of stations 1 and 2.

Station_id	list_neighbors
1	[2 4 26 32 122]
2	[4 5 6 17 25]
3	[]

Table 4.2 Example of possible neighborhood relationships among stations

Having computed neighbor stations is useful for deriving the space-time features described in the following paragraph.

4.3.2 Details about the new features

Here, a more accurate explanation of the features is given, by listing each one and how it has been computed:

- isFull: it is a Boolean feature that is true when a station is full, false otherwise. To say whether a station is full, I established the following criterion: a station is full when the percentage of used slots is greater than a threshold (i.e. 0.9) or the number of free slots is smaller than a threshold (i.e. 2 bikes). This means that the utilization of the station is almost 100%, so we do not require it is completely full, in fact in this case action is taken a little before the actual achievement of the criticality. Adding the condition on the absolute number of bikes was done because bicycles are discrete values, so you have to evaluate whether 90% is a feasible number of bikes or not (they are too many / too few, how conservative you are in the analysis). For instance, the 10% of 2 two bikes is not feasible, the 10% of 40 bikes fits. So, checking if there are less than 2 available slots in the station, that is all the slots minus two are full can avoid senseless applications of the percentual threshold.

- isEmpty: Boolean feature, it is true when a station is empty, false otherwise. Note that a station cannot be full and empty at the same time, but it can be both not full and not empty at the same time. The requirement to mark a station as empty is that the percentage of slots used is less than a threshold (i.e. 0.1) or that the number of used slots is less than a threshold (i.e. 2 bikes). A station is considered empty with a little advance and the reason why this criterion was adopted is similar to the one explained above, but specular, as well as its implementation.

- number_full_neighbors_last_timeslots: it is the number of full stations in the neighborhood in the last minutes (before the considered timeslot). Please note that the last minutes are a parameter, which corresponds to the time window length, a concept that will be discussed later. For every station, you count how many neighbor stations were full in the previous minutes. To be clear, this is the way in which I implemented the creation of the feature: for every station, I considered its neighbors and for each neighbor I calculated a time window over the minutes

indicated shifted by one row in order to have for a row the preceding x minutes, and summed the values of isFull over the window. So, I mapped the sum of isFull using the sign function, such values greater than 0 become 1 and values equal to 0 remain 0. This is cumulated over all the neighbors of the considered station in order to count how many neighbors are full. Finally, the results for each station are put together.

- number_empty_neighbors_last_timeslots: similar to the previous one, it corresponds to the number of empty stations nearby in the last minutes (before the time slot considered). Also in this case the last minutes are a parameter, intended as before. The only difference with the first feature is that you are considering not full stations, but empty ones, so you are using isEmpty to compute the feature, instead of isFull.

- Mean_used_slots_perc_neighbors: it is the mean of the percentage of used slots in the neighborhood averaged in the last minutes (before the time slot considered). Also in this case the last minutes are a parameter, intended as before. To construct the feature, I made this procedure. For each station I computed the mean of used slots percentage for the neighbors in the same timeslots, by cumulating the percentages of all neighbors and then dividing by the number of neighbors. Then I build a time window for this dataframe, with rows of the previous minutes, then I computed the mean over the window and I shifted the window by one timeslot. Finally I put everything together, for all stations. It was decided to take the mean of a relative quantity, as the percentage of used slots is, in order to have comparable measures among stations: they have different sizes in terms of total number of slots. Using the mean of used slots or free slots in the previous minutes wouldn't have been fair because they are absolute measures.

- Holiday: it is a Boolean feature, which is true on weekends or holidays, such as Christmas, Easter, or other local celebrations in the province of Barcelona, while is false on working days. The complete list of festivities was given by a python library called holidays-es [45], that reports holidays in Spain, divided by province. The distinction between workdays and holidays was devised following the suggestions of the authors of [42], who wrote that they observed two different patterns for weekends and workdays, and that behavior on local holidays, such as that of June 24th, it was similar to that of a typical Sunday [42].

Some other spatial features have been thought of, however it has not been possible to implement them in practice. The idea was to integrate external data sources to add context information around the stations. The first feature was the number of Points Of Interest within a radius of a few meters from a station. A point of interest in the city could be a university, hospital, shopping mall, and so on. This, however, was quite problematic because if we make a model for several stations the feature that considers the POIs can be useful but if we make a model for each station it is not necessary because the feature would

be a constant: the POIs do not change in a short time. The latter option was implemented in this work, so this feature was not actually useful. The other discarded characteristics were the one that said if there was an event near the station during the day, such as festivals, football matches, concerts, and the one that said if there were movements of trucks moving bikes between the stations. But it was not possible to find a source that contained all the events together, not only for 2008 but also for today; and even the data on the movements of the trucks were not found.

4.4 Sliding window method

The sliding window method converts the timeseries prediction problem into the classical supervised learning problem, solvable in this case by means of regression algorithms.
We can restructure a time series dataset in a sort of tabular format, using the previous time steps as input variables and the future time step as the output variable [2].

The sliding window, which has a fixed size, scrolls over the data as shown in figure 4-8 by taking every time a different subsample of the past series.

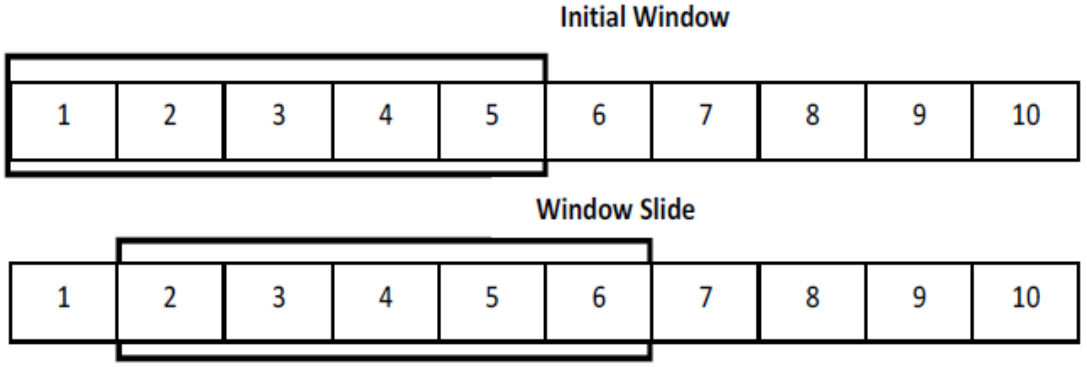


Figure 4-8 Process of sliding window, with window size of 5 [46].

A formal definition of the sliding window method can be given as follows, in the case of one-shot h-step-ahead predictions.
Consider time series data as the set of successive measurements, $\boldsymbol{X}_i = \{\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^T\}$, where $\boldsymbol{x}_i^t$ is observed values at location i and time stamp t, and T is the total number of time stamps. For a station i, the vector $\boldsymbol{x}_i^t$ contains the corresponding k features at time t, that is $\boldsymbol{x}_i^t = \{x_i^{t,1}, \dots, x_i^{t,k}\}$.
A spatio-temporal data is a collection of n multi-variate time series $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$, represented by a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times T \times k}$, where n is the number of stations. Then, a

spatiotemporal forecasting problem is cast as a regression problem. The objective of a forecasting problem is to predict $\boldsymbol{y}_{output}^{t} = x_1^{t+h}$, given $\boldsymbol{X}_{input}^{t} = \{\boldsymbol{x}^{t-w}, \boldsymbol{x}^{t-w+1}, \dots, \boldsymbol{x}^{t}\}$, where x_1 is the feature that we want to predict, w is the size of time window, and h is the forecast horizon. So, the sliding window method generates input data points $\boldsymbol{X}_{input}^{t}$ and target data points $\boldsymbol{y}_{output}^{t}$, for each timestamp in the dataset [3].

It is unavoidable to delete some rows at the beginning of the dataset since we have no previous values for constructing a full window. The number of deleted rows depends on the window size, it is two times the window size[9]. Furthermore, some values at the end of the sequence will be cancelled because we do not know the next value to predict for.

As an obvious advantage, the sliding window method permits any classical supervised learning algorithm to be applied [47], as long as the order of the rows is preserved.

4.4.1 Window size and forecast horizon

Window size and forecasting horizon are two important additional tunable parameters of the model that are set when using the sliding window method.

The window size or window width w is the dimension of the sliding window, which sets how far back the model looks during training and also for doing forecasts. It is how much past the model can use for forecasting, or the number of previous timestep considering the current one that will be used for the prediction.

Increasing the sliding window it should make the training time to increase: with a larger sliding window, the model will use more data points in training. Furthermore, the required amount of history for prediction data increases [44]. Careful reflection and experimentation is required to find a window width that results in acceptable model performance, given a certain forecast horizon.

The forecast horizon h determines how far into the future the model forecasts the target value for each row of prediction data [44]. In other words, the forecast horizon is the length of time into the future for which forecasts are to be prepared [48], or the number of periods between now and the instant of the forecast we make [49].

Below is an example of the concepts previously explained. First, a modified and short version of the data is pictured in the figure 4-9. Please, pay attention to the used_slots

[9] Some features (number_full_neighbors_last_timeslots, number_empty_neighbors_last_timeslots, Mean_used_slots_perc_neighbors) were constructed using a window approach, so they have null values in the first window_size rows. So, computing another window on them causes to have the missing values doubled.

column, which is the one that changes as it is simple to understand the mechanism, and it is the feature for which forecasts will be made.

timestamp	station	used_slots	free_slots	total_slots	used_slots_%	isFull	isEmpty	n_full_neighs_in_last_6min	n_empty_neighs_in_last_6min	mean_used_slots_perc_neighbors_6min	holiday
2008-05-15 12:02:00	1.0	0.0	18.0	18.0	0.000000	False	True	NaN	NaN	NaN	False
2008-05-15 12:04:00	1.0	1.0	18.0	18.0	0.000000	False	True	NaN	NaN	NaN	False
2008-05-15 12:06:00	1.0	2.0	19.0	18.0	0.000000	False	True	NaN	NaN	NaN	False
2008-05-15 12:08:00	1.0	3.0	18.0	18.0	0.000000	False	True	0.0	2.0	0.219643	False
2008-05-15 12:10:00	1.0	4.0	18.0	18.0	0.000000	False	True	0.0	2.0	0.219643	False
2008-05-15 12:12:00	1.0	5.0	18.0	18.0	0.000000	False	True	0.0	2.0	0.223611	False
2008-05-15 12:14:00	1.0	6.0	18.0	18.0	0.000000	False	True	0.0	2.0	0.223611	False

Figure 4-9 Example of a possible dataset before data pre-processing.

Then, suppose to build a tabular version of the dataset, by means of sliding window method, having w=3 timeslots (6 minutes) and h=1 timeslot (2 minutes). The result, $\boldsymbol{X}_{input}^{t}$, is the one of figure 4-9.

timestamp	used_slots_lag_seq_3__0	used_slots_lag_seq_3__1	used_slots_lag_seq_3__2	free_slots_lag_seq_3__0	free_slots_lag_seq_3__1	free_slots_lag_seq_3__2	total_slots_lag_seq_3__0	total
2008-05-15 12:08:00	0.0	1.0	2.0	18.0	18.0	19.0	18.0	
2008-05-15 12:10:00	1.0	2.0	3.0	18.0	19.0	18.0	18.0	
2008-05-15 12:12:00	2.0	3.0	4.0	19.0	18.0	18.0	18.0	
2008-05-15 12:14:00	3.0	4.0	5.0	18.0	18.0	18.0	18.0	
2008-05-15 12:16:00	4.0	5.0	6.0	18.0	18.0	18.0	18.0	
2008-05-15 12:18:00	5.0	6.0	7.0	18.0	18.0	18.0	18.0	
2008-05-15 12:20:00	6.0	7.0	8.0	18.0	18.0	18.0	18.0	

Figure 4-10 Example of a possible dataset after data pre-processing (sliding window method), in tabular format. Actually, it is not the complete one because there were too much columns to show, but relevant information, such as the transformation of used_slots column, is present.

Consider that the time t = 12:08 is the instant of time for which the forecast is to be made. Looking at used_slots, we can see that the first row (at instant t = 12:08) contains the three values of the slots used at instant t-3 (12:02), t-2 (12:04), t -1 (12: 06) which corresponds to the values 0,1,2. These three values make up the sliding window for t = 12:08. Since the forecast horizon is 1, the sliding window contains the three values immediately before the

time slot for which you want to make the forecast. If the forecast horizon had been 3, the same window with values 0,1,2 would have served to forecast for the hour 12:12.

In parallel, the target for the forecast, y^t_{output}, is also built, which is in this case the used_slots feature moved forward by the forecast horizon.

```
timestamp
2008-05-15 12:08:00     3.0
2008-05-15 12:10:00     4.0
2008-05-15 12:12:00     5.0
2008-05-15 12:14:00     6.0
2008-05-15 12:16:00     7.0
2008-05-15 12:18:00     8.0
2008-05-15 12:20:00     9.0
2008-05-15 12:22:00    10.0
2008-05-15 12:24:00    11.0
2008-05-15 12:26:00    12.0
2008-05-15 12:28:00    13.0
2008-05-15 12:30:00    14.0
2008-05-15 12:32:00    15.0
Name: used_slots, dtype: float64
```

Figure 4-11 Example of target for forecast.

4.5 Forecasting and evaluation

This section describes the algorithms used for predictions and the performance metrics used to evaluate the results, providing a theoretical explanation.

4.5.1 Forecasting algorithms

Since the sliding window method transforms the timeseries forecasting problem into the classical supervised learning problem, regression algorithms can be exploited to model and provide predictions. Two well-known algorithms, Support Vector Machines applied to regression and the linear regression are presented in the following subsections, as well as a dummy forecaster which was used as a baseline.

4.5.1.1 Support Vector Regressor

Support vector machine (SVM) is a popular machine learning algorithm for classification and regression, described by Vladimir Vapnik and his colleagues in 1992 [50]. Its goal is to find a function f(x) that given the input x approximates the target y, by solving a certain optimization problem.
Both linear and non-linear version of SVM exist, which are described below for the regression case (SVR).

The basic idea of linear SVR is the following. Suppose to have some training data $\{(x_1, y_1), \dots, (x_l, y_l)\} \subset \mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ is the space of the input features (e.g. $\mathcal{X} = \mathbb{R}^d$). Then, the goal is to find a linear function $f(x) = \langle w, x \rangle + b$ with $w \in \mathcal{X}, b \in \mathbb{R}$ that has at most ϵ deviation from the actual targets y_i for all the training data, and at the same time is as flat as possible [51].
Here a tacit assumption is made: there is indeed a function f that approximates all pairs (x_i, y_i) with ϵ precision. However, this function may not exist because the problem is not perfectly linear, so we can define an optimization problem that allows for deviations even greater than ϵ, introducing slack variables ξ_i, ξ_i^* that relax the constraint of strict linearity, as illustrated graphically in figure 4-12.

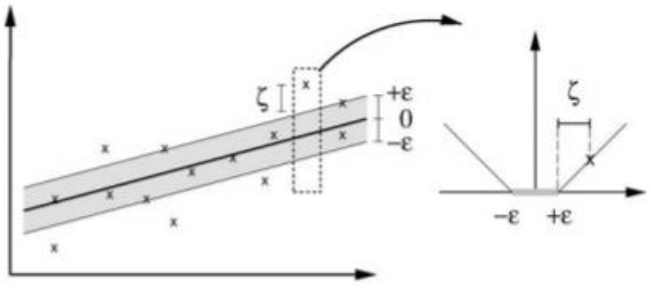


Figure 4-12 The soft-margin loss setting for a linear SVR [51]. The shaded region is called the ϵ-tube, inside which the predicted function should lie. Data points on the margin of the ϵ-tube are the Support Vectors.

Therefore, the primal optimization problem is as follows:

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*)$$

$$\text{subject to} \quad \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

The first part of the objective function is a weight decay which is used to regularize the size of the weights and penalizes large weights w, such that the weights converge to smaller values. The second part is a penalty function that penalizes errors greater than ϵ [10]. The constant $C > 0$ regulates the compromise among the flatness of f and the amount up to which deviations greater than ϵ are tolerated [51]. Errors larger than $\pm\epsilon$ are denoted with the slack variables ξ (above ϵ) and ξ^* (below ϵ), respectively [10].
Then there are the constraints that are set on the errors between regression predictions $\langle w, x_i \rangle + b$ and true values y_i.

The problem in its primal formulation is hard to solve, so the dual formulation can help in dealing more easily with the problem. To obtain the dual version of the problem, the

method of Lagrange multipliers can be adopted: first, the Lagrangian L is calculated as the sum of the primal objective function and the constraints of primal formulation, multiplied by non-negative dual variables. Then, the partial derivatives of L with respect to the primal variables are set equal to zero and substituted in the Lagrangian, obtaining a new objective function. Finally, it is no longer a question of minimizing, but of maximizing the new objective function. In the end, this is the dual formulation:

$$\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\varepsilon\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$\text{subject to} \quad \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C]$$

where α_i, α_i^* are the dual variables (or Lagrange multipliers) associated with a specific training point. The weights w and the solution f(x) can be expressed as:

$$w = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)x_i$$

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\langle x_i, x\rangle + b$$

We do not need to compute *w* explicitly and only a subset of training points serve to define the solution. In fact, for all samples inside the ϵ-tube, α_i, α_i^* vanish, so we have a sparse expansion of w in terms of x_i (i.e. we do not need all x_i to describe w). The examples that come with nonvanishing coefficients are called Support Vectors and they are sufficient to get f(x) [51]. The smaller the fraction of support vectors, the more general the obtained solution is and less computations are required to evaluate the solution on new data points.
Furthermore, it can be proven that the solution found is global because the problem formulation is convex. Also, if the cost function is strictly convex, the solution found is also unique [10].

The linear version of SVR may not work if the problem is significantly non-linear. For this reason also a non-linear version of SVR algorithm was thought, that starting from the linear version can be get by simply mapping the training features x_i with a function $\Phi: \mathcal{X} \to \mathcal{F}$ into some new feature space $\mathcal{F}$. Finding such function Φ and applying it to the data may be complex and computationally intensive, so an alternative way to use the mapping was found in the kernel trick: it can be proved that it is enough to consider only the kernel function K, without applying the mapping at all. In particular, a kernel K for a mapping Φ is a function $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that implements inner product in the feature space:
$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$.

Then, x_i are substituted by $\Phi(x_i)$ and the kernel function $K(x_i, x_j)$ is put in place of the scalar product $\langle x_i, x_j \rangle$ in the dual formulation of the problem:

$$\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ -\varepsilon\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$\text{subject to} \quad \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C]$$

So, the weights w and the solution of the optimization problem f(x) become:

$$w = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\Phi(x_i)$$

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)K(x_i, x) + b$$

The most frequently used kernel functions are the Gaussian RBF with a width of σ: $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$ and also the polynomial kernel with degree k, defined as: $K(x_i, x_j) = (1 + \langle x_i, x_j \rangle)^k$.

The difference with respect to the linear case is that w is no longer given explicitly. Furthermore, see that for non-linear SVR, thus using kernels, the optimization problem coincide to finding the flattest function in the feature space, not in input space anymore [51].

As an advantage, the optimization problem in the dual form is a convex optimization problem resulting in a global solution which in many cases yields unique solutions. However, the major drawback is that the training time can be large for data sets containing many objects [10].

4.5.1.2 Linear Regression

Linear regression is a simple and one of the most widely used models and a common statistical tool for solving the regression task. The concept is to model the relationship between some explanatory variables (called the predictors) and some real valued outcome, that is the response or target variable [52]. In particular, linear regression gives the response or target variable as a linear function of the inputs. When you have just one predictor, the model is called simple linear regression model, however this is not the case in our

application, where we will exploit the multiple linear regression, that is used when you have p > 1 predictors.

The multiple linear regression model can be expressed in the matrix form:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon}$$

where $\boldsymbol{y} = (y_1, \dots, y_n)' \in \mathbb{R}^n$ is the response vector, $\boldsymbol{X} = [\boldsymbol{1}_n, \boldsymbol{x}_1, \dots, \boldsymbol{x}_p] \in \mathbb{R}^{n\times(p+1)}$ is the input matrix, whose first column is a vector of ones and $\boldsymbol{x}_j = (x_{1j}, \dots, x_{nj})' \in \mathbb{R}^n$ indicate the j-th predictor vector. Then, $\boldsymbol{w} = (w_0, w_1, \dots, w_p)' \in \mathbb{R}^{p+1}$ represents the weight[10] vector that we would like to estimate and $\boldsymbol{\epsilon} = (\epsilon_1, \dots, \epsilon_n)' \in \mathbb{R}^n$ is the error vector [53]. In particular, $\boldsymbol{\epsilon}$ represents the residual error between the predictions and the response vector [54]. Usually $\boldsymbol{\epsilon}$ is a random variable assumed to have a Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}_n, \sigma^2 \boldsymbol{I}_n)$. This correspond to an assumption of normality and homoscedasticity of the error term and it is worthy saying that errors are supposed to be independent and identically distributed.

Without $\boldsymbol{\epsilon}$, any observed point would lie on the true regression hyper-plane[11], that is the regression hyper-plane of the entire population from which samples were taken. When including the random error term a point may fall either above the true regression hyper-hyper-plane (when the error is positive) or below the hyper-plane (when it is negative) [55]. Our goal is to obtain the predictions for each data point, that is finding a vector of fitted (or predicted) values $\hat{\boldsymbol{y}} = (\hat{y}_1, \dots, \hat{y}_n)' \in \mathbb{R}^n$. This vector is computed using an estimate of the weights and successively solving the equation of the estimated regression hyper-plane: $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}$. Then, how to obtain the estimate $\hat{\boldsymbol{w}}$?

One may compute the Maximum Likelihood Estimator for the weights and this would give an estimate. However, it was proven that maximizing the likelihood with respect to $\boldsymbol{w}$ is equivalent to minimizing the residual sum of squared residuals [54].
So, a simple method can be used, which is called the least square minimization, based on the least squares principle, which was introduced by the German mathematician Gauss [55]. According to this principle, a hyper-plane fits well to the data if the deviations from the observed points to the hyper-plane are small. The goodness of fit is measured by the sum of the squares of the deviations: the best case is having the sum of squared deviations as small as possible.

[10] In statistics, the weights are commonly named coefficients. Here the machine learning definition was presented.
[11] In the case of multiple linear regression, it is not a line but a linear hyper-plane in $\mathbb{R}^{p+1}$, that corresponds to a line in more than three dimensions.

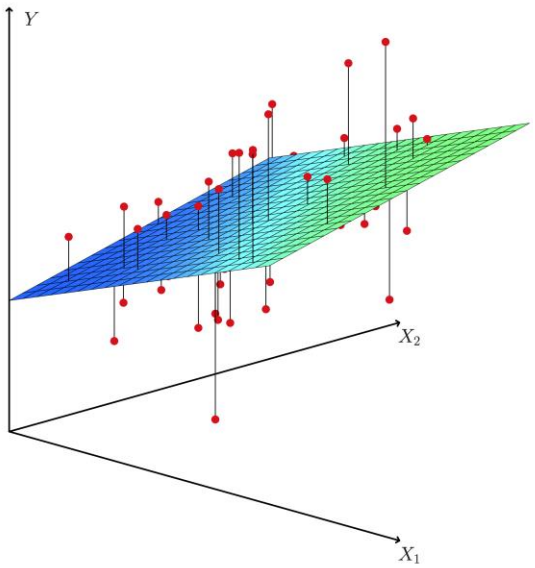


Figure 4-13 The regression hyper-plane in 3-dimensions [56]. The vertical lines between the hyper-plane and the red dots (the data points) represent the residuals.

In practice, an optimization problem can be formulated in order to minimize the residual sum of squares between the predicted targets and the actual targets:

$$\min_{\boldsymbol{w}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

Where $\hat{\boldsymbol{e}} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}} = \boldsymbol{y} - \hat{\boldsymbol{y}}$ are the residuals.
The solution of this problem is called the ordinary least squares solution and it is expressed as:

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

The ordinary least squares solution is based on the assumption that all the predictors in $\boldsymbol{X}$ are independent. When this assumption is violated, a situation of multicollinearity can happen, which we would like to avoid, if possible. Multicollinearity is observed when the predictors are highly correlated and the columns of the input matrix $\boldsymbol{X}$ are approximately linear dependent. As a consequence, the least squares estimate becomes really sensitive to random errors in the observed target [57].

Furthermore, the method is really sensible to outliers: when they are present, performances could drop since the squared error penalizes deviations quadratically and the outliers, being far from the possible fitted regression hyper-plane affect more the fit than points near the hyper-plane [54].

These two observations, together with the assumptions made earlier about the error term and the strong assumption that the relationship between each predictor and the response is linear, given the other predictors, suggest that the linear regression algorithm has some application limitations. If the relationship between the input matrix $\boldsymbol{X}$ and the response variable is not linear, underfitting may occur and low accuracy may be obtained [58]. However, to reduce this inconvenience, it is possible to apply some data transformations to the inputs, through any function, which can be a logarithmic or quadratic function, and so on. In fact, the model must be linear in the parameters (the weights) and not in the inputs.
The great advantages of linear regression are its simplicity and ease of understanding and execution: it has a lower time complexity than many other machine learning algorithms [58].

4.5.1.3 Baseline

You must have a baseline for reference. The simplest one is a dummy forecaster, also suggested by the authors of [11], that predicts the current state of the station (the number of available bikes or free slots) for any time in the future. So if I have to predict what will happen in 10 minutes and there are currently 5 available bicycles, the system will predict that 10 minutes later there will still be 5 bicycles available. In some moments, when the system is very stopped, it could work well or even better than a prediction algorithm.

4.5.2 Performance metrics (R2 and RMSE)

Some metrics are needed to evaluate the performance of the forecasting model based on the type of problem, that is, to perform regression for time series forecasting. However, even though regression analysis constitutes a large chunk of machine learning and computational statistics domains, no consensus has yet been reached on a unified preferred rate for evaluating regression analyzes [59]. Then a selection of evaluation metrics was made as there are many possible ones, each expressing slightly different information. The metrics used in this thesis are the coefficient of determination R2 and the Root Mean Squared Error.

4.5.2.1 R2

R^2, the coefficient of determination, represents the proportion of variance of the target y that has been explained by the independent variables in the model. It is useful as a regression metric since it provides an indication of goodness of fit and therefore a measure of how well unseen samples are likely to be predicted by the model, through the proportion of explained variance [60]. It is computed by means of the following formula:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where n is the number of samples, $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, that is the observed value, $\bar{y}$ is the mean of observed data, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$. The denominator of the fraction is also called the total sum of squares, while the numerator is the residual sum of squares.
R^2 is dataset dependent, since it depends on the explained variance which is also dataset dependent, so it may not be meaningfully comparable across different datasets [60]. The best case is when the predicted values exactly match the observed values: the residual sum of squares is null, so the best possible score is $R^2=1$ [61]. A score of 0 for R^2 is not desirable, since it would mean a constant model that always predicts the expected value of y, disregarding the input features [60].
To use R^2 as a performance metric, we must first bear in mind some facts: R^2 does not indicate whether the most appropriate set of independent variables has been chosen (R^2 increases as the number of variables in the model is increased), if explanatory variables are collinear, if the right type of regression model was adopted and also if there are enough data points to give robust results [61]. Despite this, some researchers suggest to focus on the ranking generated by the coefficient of determination, because it is the only metric that considers the distribution of all the ground truth values, and generates a high score only if the regression correctly predict most of the values of each ground truth category [59].

4.5.2.2 Root Mean Squared Error (RMSE)

The Root Mean Squared Error is computed as the square root of the second sample moment[12] of the differences between observed values and predicted values, that are the prediction errors:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where n is the number of samples, $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, that is the observed value.
RMSE expresses the average model prediction error in units of the variable of interest, so comparisons across different datasets would be invalid because the measure is dependent on the scale of the numbers used. In fact, RMSE is used as a measure of accuracy to

[12] The k-th sample moment, applied to a sample $X_1, \ldots, X_n$, is $\frac{1}{n}\sum_{i=1}^{n} X_i^k$

compare forecasting errors of different models on a particular dataset, since it aggregates the magnitudes of errors in predictions for several data points into a single measure [62]. It is in the range from 0 to ∞ and is a negatively-oriented score, which means that lower values are better [63]. The value 0 would indicate a perfect fit to the data, but, even if it is possible in theory, in practice it is almost never achieved [62].

RMSE penalizes large errors due to the squared term [64], because the errors are squared before they are averaged. This implies that the RMSE should be more suitable when large errors are particularly undesirable [63], but as a drawback it is more sensitive to outliers [65].

Capitolo 5

Experiments

All the experiments were run on the first 10 stations for evaluating the effects of changing the parameters of the sliding window method - window size and forecast horizon – and the prediction algorithm. Then the variations introduced in the different experiments has been compared.
Analyzing a subset of 10 out of the 272 stations remained after data cleaning may seem reductive, because it is a very small sample and there are no easy ways for saying that this sample is representative for all the other stations, since we would need extra information about the domain. However, just consider that for running this experiment took about 21 hours of execution for 10 station. Running it for more or even for all station would have been too much time consuming, it could have last for days or weeks. So due to the long execution times and the limited computational power this choice has been reasonable.

It is important to remark that a model has been built separately for each station since each station is considered a stand-alone system for simplicity and stations are related to each other by means of the neighborhood features.
To get particular and overall results, the metrics, R2 and RMSE, were computed both separately for each station and aggregating them among all stations. In details, the sum and the mean of both R2 and RMSE, respectively were given. The mean was calculated as a weighted average, since we may make a different number of predictions for each station. So, the weights are the ratio between the number of predictions made for a station and the maximum of predictions made among all stations.

Some other considerations may be reported which are about all the experiments. The split in training and test set was done separately for each station, imposing a percentage for the test set, which is the 20% of the data. So, the test set contains the last three weeks in September, while the training set has data in May, June, August and the first week in September.
In almost all the experiments we want to predict the number of bikes in station, that is used slots. In the third experiment we did a trial predicting the number of available slots in a station, that is free slots.

As the last point, predictions have been normalized before computing the performance metrics. Since the number of bikes is a discrete quantity, the predicted values have been rounded to the closest integer and since it should be non-negative, negative predictions were corrected to zero.

5.1 Effects on performances of changing window size, forecast horizon and prediction algorithm

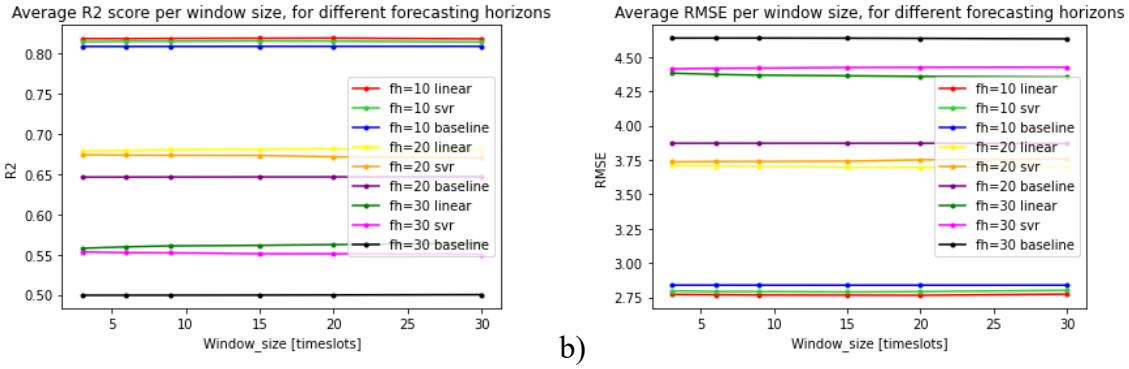
A first experiment was run on the first 10 stations for evaluating the effects of changing the parameters of the sliding window method - window size and forecast horizon – and the prediction algorithm.

All the combinations of these values of the parameters were tried:

- Window size w: 3, 6, 9, 15, 20, 30 [timeslots]
- Forecast horizon h: 10, 20, 30 [timeslots]
- Algorithm: SVR, linear regression, baseline

Please note that the unit of measurement of the window size and of the forecast horizon are timeslots; it means that for obtaining the time expressed in minutes you have to multiply by 2 because the granularity is 2 minutes (timeslots are every 2 minutes). For instance, a value of 10 timeslots for the forecast horizon means that we want to predict 20 minutes ahead.

The results are summarized in figure 5-1, where one can see the impact of changing the window size, the forecast horizon and the algorithm on the R2 and the RMSE. The plots report the weighted average of R2 and RMSE made over the stations.



a) b)

Figure 5-1 Average R2 score a) and average RMSE b) per window size for different forecast horizons and algorithms.

What is evident is that R2 and RMSE have opposite trends: when R2 is large, RMSE is small, performances are nice and vice versa. So, the two plots appear somehow specular.

The performances do not vary much as the window size increases, there is only a slight deterioration for the SVR algorithm and a slight improvement for linear regression, but they are almost imperceptible.
Then, increasing the forecast horizon decreases the performances, regardless of the values of the window size.
The difference between baseline and SVR/linear regression also increases as the forecasting horizon increases. For the same forecasting horizon, linear regression has the smallest error, SVR is in the middle but close to linear regression, finally the baseline has major error.
For instance, looking at the case with window size of 3, the discrepancy for R2 between SVR and the baseline with h=10 is 0.8140-0.808= 0.006, while with h=30 is 0.553-0.500=0.053, which is one order of magnitude higher.
A similar consideration can be made for the RMSE: in this case the difference between SVR and the baseline for h = 10 is an order of magnitude smaller than in the case with h = 30.

It may be specified that as the window size or the forecast horizon increases, the number of removed Nan values in the train and test sets increases. However this has not a relevant impact on the performances because the number of removed rows is negligible with respect to the size of the training and test sets.

Some considerations can be made regarding the execution time. It was observed that the time for computing the spatial-temporal features does not change very much despite that the window size grows. It takes about 5-6 minutes for each of the window-size values used.

It was observed that the growth of forecast horizon has no direct effect on the execution times for training and testing of prediction models because the concept of forecast horizon was implemented a simple shift on the targets.

Then, increasing the window size, the execution time[13] for training and testing for linear regression and the baseline does not change much: it takes at most 1 minute of execution. Instead, increasing the window size makes the execution time to notably grow for SVR. So, SVR leads to similar results, a little worse than linear regression and is much slower than linear regression.
This may be explained as the tendency of being slow on big input datasets for SVR. In fact, increasing the window size means adding more and more features in the table constructed with the sliding window method. For instance, consider the case with window size of 3. The features we want to transform into tabular format are 10 (you can see all of them as the column names[14] in figure 4-9). So, in the constructed table you will have 3x10=30 features, 3 for each feature. Then, think about the case where the window is 30 timeslots long: you

[13] The execution time here is intended as the whole execution time on the 10 stations (thus it is the sum of the execution times for each station).
[14] The column “station” is not considered as a feature in the analysis.

will have 30x10=300 features in the table, which is huge. Having very a high number of features in the input passed to SVR makes harder and more time consuming to solve the optimization problem, as previously written in section 4.5.1.1.
The graph below shows the trend of the average time (average on the forecast horizon) for different window values, all using SVR as an algorithm. The execution time increases almost exponentially between 3 and 15, approximately linearly between 15 and 30. It passes from 40 minutes (with w=3) to 1 hour and 40 minutes (with w=30).

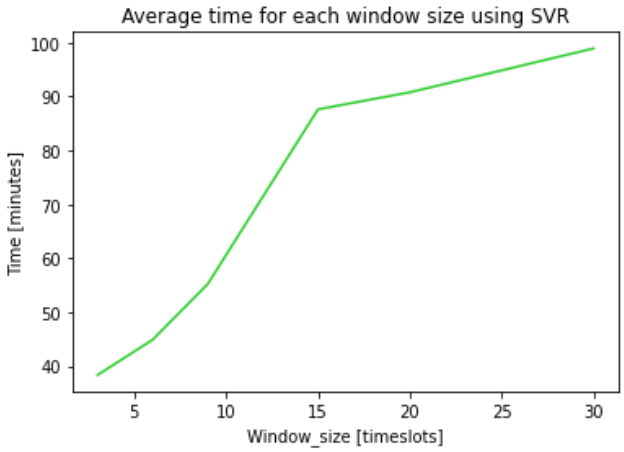


Figure 5-2 Effect of increasing the window size on the training and test time for SVR.

After that, we may want to look at the overall performances of every station, to find which are the most critical stations. To do this, we report on average the values of R2, RMSE, execution time, rows removed, total slots having grouped by station. Therefore it is an average made on the configurations of algorithm, forecast horizon and window size.
The summary of stations performances is presented in figures 5-3 and 5-4.

station	time	RMSE	R2	perc_removed_train	abs_removed_train	perc_removed_test	abs_removed_test	total_slots
6	2.306088	4.719289	0.824159	0.000951	46.666667	0.002676	32.833333	33.228251
5	2.263232	4.877183	0.763004	0.000951	46.666667	0.002676	32.833333	31.212277
9	2.409964	2.567257	0.720762	0.000951	46.666667	0.002676	32.833333	15.425076
3	2.306701	3.396897	0.709195	0.000951	46.666667	0.002676	32.833333	19.251642
1	2.335280	3.678721	0.698291	0.000951	46.666667	0.002676	32.833333	19.205122
8	2.414181	3.231381	0.687352	0.000951	46.666667	0.002676	32.833333	19.460366
10	2.469208	3.572877	0.658641	0.000951	46.666667	0.002676	32.833333	18.292833
2	2.327393	4.976845	0.583940	0.000951	46.666667	0.002676	32.833333	21.290844
4	2.325893	2.523784	0.568954	0.000951	46.666667	0.002676	32.833333	16.102284
7	2.377051	3.286242	0.513383	0.000951	46.666667	0.002676	32.833333	18.426266

Figure 5-3 Stations performances sorted by decreasing RMSE.

station	time	RMSE	R2	perc_removed_train	abs_removed_train	perc_removed_test	abs_removed_test	total_slots
2	2.327393	4.976845	0.583940	0.000951	46.666667	0.002676	32.833333	21.290844
5	2.263232	4.877183	0.763004	0.000951	46.666667	0.002676	32.833333	31.212277
6	2.306088	4.719289	0.824159	0.000951	46.666667	0.002676	32.833333	33.228251
1	2.335280	3.678721	0.698291	0.000951	46.666667	0.002676	32.833333	19.205122
10	2.469208	3.572877	0.658641	0.000951	46.666667	0.002676	32.833333	18.292833
3	2.306701	3.396897	0.709195	0.000951	46.666667	0.002676	32.833333	19.251642
7	2.377051	3.286242	0.513383	0.000951	46.666667	0.002676	32.833333	18.426266
8	2.414181	3.231381	0.687352	0.000951	46.666667	0.002676	32.833333	19.460366
9	2.409964	2.567257	0.720762	0.000951	46.666667	0.002676	32.833333	15.425076
4	2.325893	2.523784	0.568954	0.000951	46.666667	0.002676	32.833333	16.102284

Figure 5-4 Stations performances sorted by decreasing R2.

It was interesting to notice that the most critical stations are the biggest ones, so it seems that the dimension of a station has an influence on the quality of the predictions.
The stations identified as critical are stations 2, 5 and 6 that have on average 21, 31 and 33 total slots[15]. It can be seen that stations 2, 5 and 6 have the worst performances having the highest RMSEs, but a contradiction seems to be arise for stations 5 and 6, that have the highest average R2 scores. In fact, it seems strange that stations with higher RMSEs have also a higher R2, shouldn't it be vice versa? Assuming that the higher RMSE the worse the prediction and the higher R2 the better the prediction?
But remember that the RMSE expresses the average model prediction error in units of the variable of interest, while the R2 score is in percentage. So, to compare them we may compute an approximate percentage of the RMSE based on the station size. However, this percentage RMSE is not really reliable since the dimension of a station changes with time. To see in practice, consider the case of station 6.
For station 6, being the largest station, it is acceptable that the RMSE error is higher than for a small station. The discrepancy that seems to exist between the RMSE and the R2 may be linked to that. If the station has more slots, it is more difficult to make the correct prediction. That is perhaps the reason why if you order, the RMSE of station 6 is worse than the others while the R2 is better. The RMSE for station 6 is 4.7, so the prediction is wrong by 4.7 on average out of 33 slots: about the 14% of the slots are badly predicted. Then consider a small station, such as station 9 that has on average 15 slots. The RMSE for station 9 is about 2.6, therefore the error is 2.6 on average out of 15 slots, that is 16.6% slots badly predicted. In proportion it seems that the errors have similar magnitude, so high values of RMSE can be related to the high number of total slots for a station.

[15] The value of total slots is an average value computed over time: in fact, total slots is not constant with time but here we need a unique value in order to make our considerations.

5.2 Distinguishing between stationary and non-stationary data

An improvement of the previous experiment was made adding the distinction between stationary and non-stationary data. The reasoning behind is that in certain periods of time the number of bikes has very little variations or even it is almost constant. In these time intervals the dummy forecaster, that is the baseline, is supposed to work better than the regression algorithms.

The idea is to identify on the training data the time periods in which the trend is on average stationary and those in which it is not. Then the regression model is trained only on non-stationary periods and in the test phase we proceed by analyzing the windows of past data. If a window is not stationary the trained model is used. If it is stationary, the baseline is used.
Then we need a way to understand if a certain window is stationary for both training and test data and we have to establish a criterion for the stationarity.

In statistics, a process is stationary when the mean, variance and autocorrelation structure do not vary over time [66]. A stationary time series looks flat over time, without trend, and has constant variance over time, a constant autocorrelation structure over time and no periodic fluctuations (seasonality) [66].
However, we do not mean stationarity here that way. Rather, we call stationary a portion of a time series that is flat and almost constant over time, therefore with a small variability: for a window to be stationary, the variance of the number of bikes (used slots) within the window must be less than a certain threshold. Asking that the variance must be zero within a window would mean that all the values in the window are equal, but this is a very strict constraint, so we relaxed it by requiring that the variance must be below a threshold to define a window stationary.
Having this in mind and varying the threshold, we would like to know how many windows are stationary or not and to understand the behavior of the prediction model.

The experiment, just like the previous one, was run on the first 10 stations. The forecast horizon and window size parameters have been varied as in the previous experiment, but now also the parameter of the threshold on variance is considered and the possible prediction algorithms are the combination of SVR with the baseline and the combination of linear regression with the baseline. So, the possible values of the parameters are as follows:

- Window size w: 3, 6, 9, 15, 20, 30 [timeslots]
- Forecast horizon h: 10, 20, 30 [timeslots]
- Algorithm: SVR combined, linear regression combined
- Threshold on variance: 0.2, 0.5, 1.0

The rows (the windows) of training and test tables, that are the output of the sliding window method, should be divided into stationary and non-stationary and aggregated in such a way

to create stationary training set, not-stationary training set, stationary test set and non-stationary test set.
To do this a column isStationary is added to the training and test tables, separately. IsStationary is a Boolean series that is true if the window at the corresponding timestamp has the variance on the used slots values below the specified threshold, false otherwise.
Then the four subsets described before are generated, by masking on the column isStationary, which is actually destroyed after that since it does not help in the forecasting phase.
So, the regression model is trained only on the non-stationary train table and predicts data in the non-stationary test table; stationary test data are predicted through the baseline. After that, the two predictions are concatenated together and sorted by timestamp, to form a single array for the predicted values of used slots. Finally execution times and performance metrics are computed and evaluated.

The results are summarized in figure 5-5, where one can see the impact of changing the window size, the forecast horizon and the algorithm on the R2 and the RMSE. The plots report the weighted average of R2 and RMSE made over the stations and over the variance threshold. The choice of averaging on the variance threshold was made because there would have been too confusing and complicated to see the results in the plots if there was an extra line for each possible value of the variance threshold. However, the impact of the threshold is studied next with apposite graphs. For the same reason of readability it was chosen to make a plot for the combination of SVR-baseline and linear regression-baseline, separately.

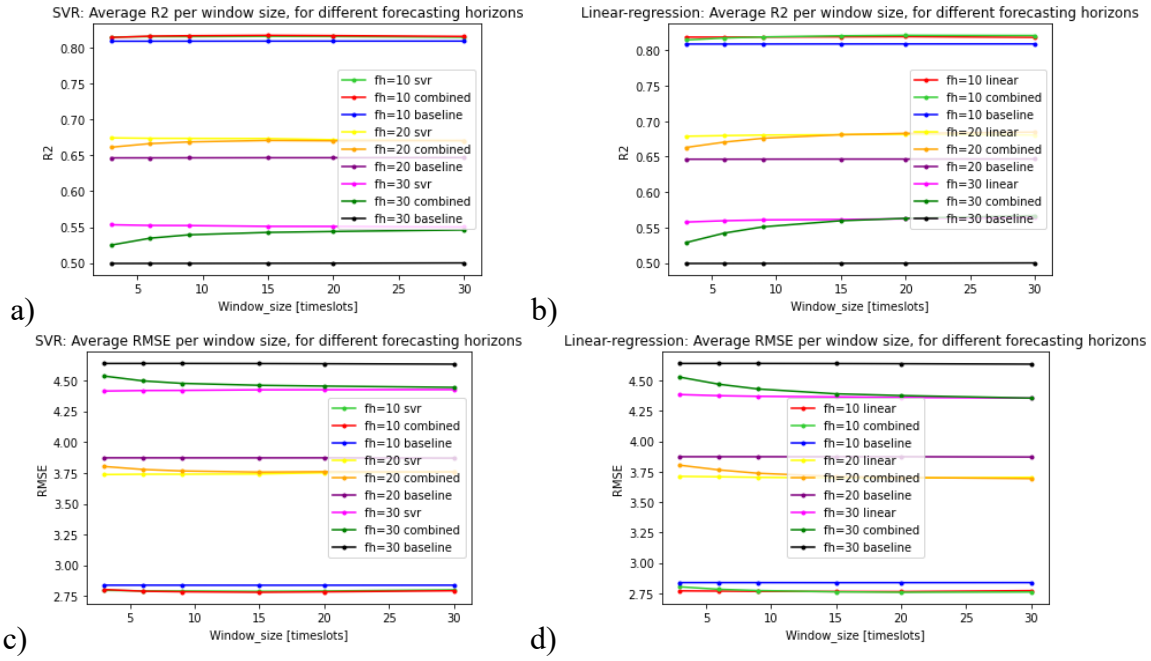
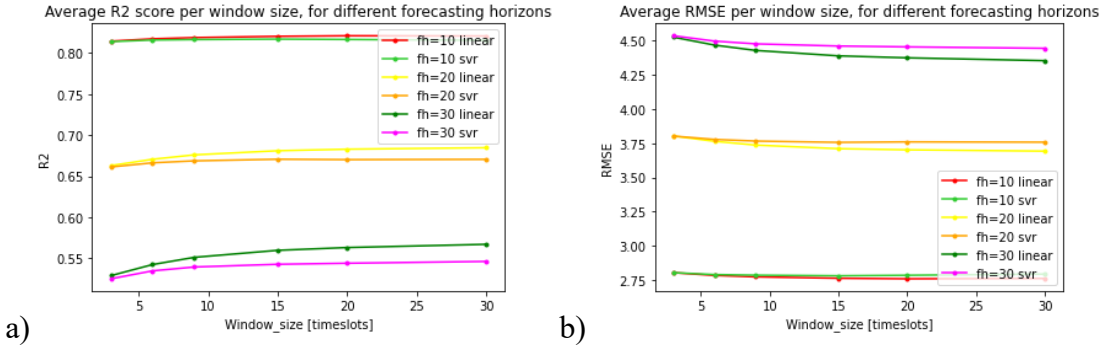


Figure 5-5 Average R2 score a) and average RMSE b) per window size for different forecast horizons for SVR-combined; average R2 score c) and average RMSE d) per window size for different forecast horizons for linear regression-combined.

The combination of the baseline and a regressor (that corresponds to the distinction between stationary and non-stationary windows) has performances in between the regressor alone and the baseline: the combined works better than the baseline but worse than the regressor alone, both for SVR and linear regression. But it can be noticed that with a large enough window the combined algorithms outperform both the regressor alone and the baseline. In any case, the results are quite similar to those obtained without making distinctions between stationary and non-stationary data.
As previously seen, performance degrades when increasing the forecast horizon, in particular for the baseline.
For forecast horizon = 10 using the regressor alone or the combined with the baseline does not change anything. If the forecasting horizon increases, the difference between the algorithms increases.
Despite that performance of the baseline and of the regressor alone is the same changing the window for the same forecast horizon, using combined the situation is slightly different: by increasing the window size, the performance of the combined improves. This may be due to a better individuation of stationary and non-stationary data or because it is more difficult to have stationary windows when windows are big, so the regressor is trained on more non-stationary data.

Then, one may compare the two combined models and see their performances in figure 5-6.



a) b)

Figure 5-6 Average R2 score a) and average RMSE b) per window size for different forecast horizons for SVR-combined and linear regression-combined.

It can be noticed that linear regression performs better than SVR, also when it is combined with the baseline (not just when used alone). The difference among the two is small, mostly with a small forecast horizon. Again we see that the metrics are not nearly constant as the

window size changes: increasing the window size, R2 increases and RMSE decreases, therefore performance improves.

One may study how performance varies when changing the threshold on variance, with the help of the graphs in the figure 5-7. The plots report the average RMSE over the stations per window size for different variance thresholds for SVR-combined and linear regression-combined, for different forecast horizons. The corresponding plots of average R2 are not described here, but they are equivalent.

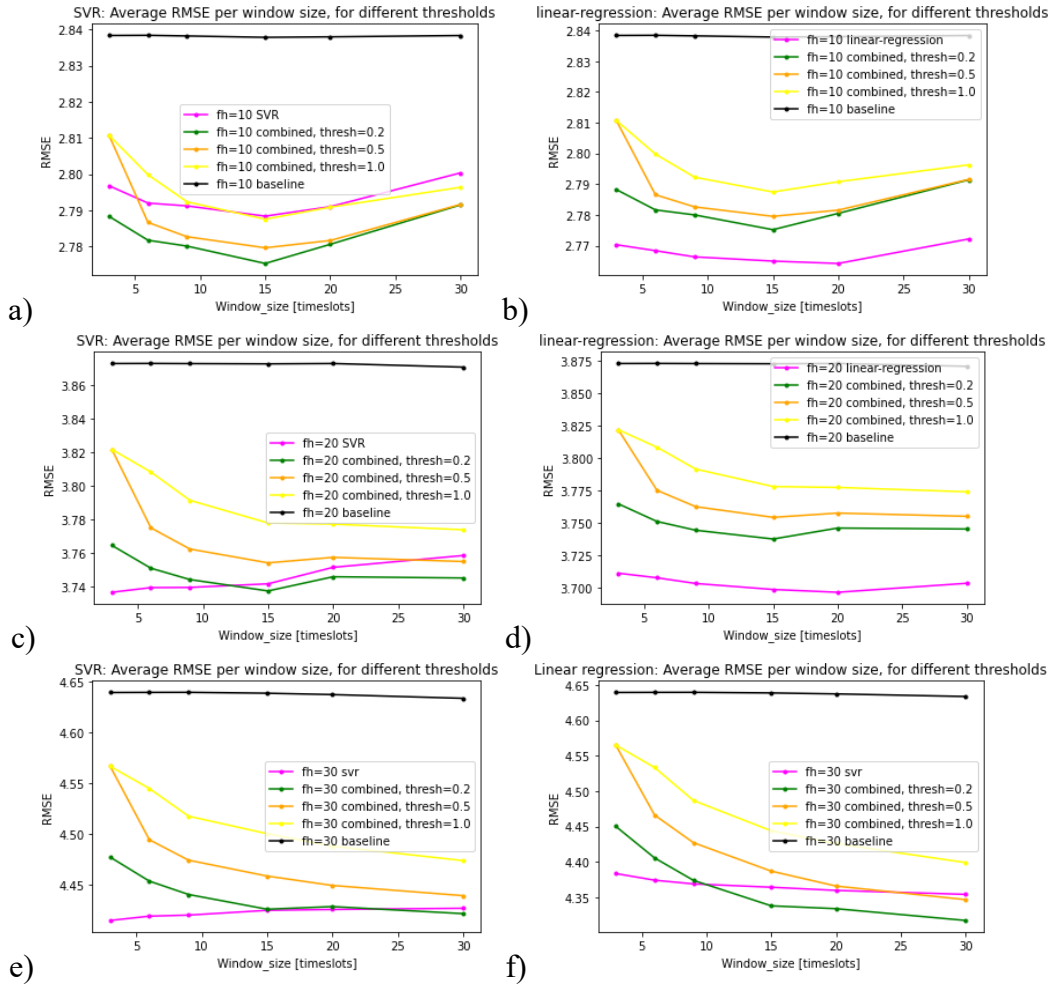


Figure 5-7 Average RMSE per window size for different variance thresholds for SVR-combined (a, c, e) and linear regression-combined (b, d, f), with forecast horizon 10 (a, b), 20 (c, d) and 30 (d, e).

It was experienced that increasing the threshold on variance, performance worsens, so a lower threshold is better since it allows to get lower RMSE and higher R2. A lower

threshold means that you are more strict when defining a window stationary, then you have fewer stationary windows and the regressor alone is used more times.

Then, it can be seen that the combined works better with variance threshold of 0.2, the value 0.5 is in the middle and 1.0 performs worst. However, with a forecast horizon of 10 timeslots, the average RMSE is practically the same with thresholds 0.5 and 1. This may be due to the fact that with large windows the two higher thresholds bring to a similar distinction between stationary and non-stationary windows.

After that, one might notice that with forecast horizon of 10 and 20 timeslots the combined works best with a window size of 15. On the other hand, with a longer forecast horizon (30 timeslots) larger windows are better: the combined has the best performances with window size of 30 timeslots.

Finally, when using SVR as regression algorithm, the combined version works generally better than SVR alone. With linear regression the regressor alone generally gives lower RMSE than the combined and then better performance. With shorter forecast horizons, the size of the window necessary to have an improvement over SVR alone is smaller.

Regarding the execution time, some considerations can be made. By increasing the window, the execution time for training and testing increases, both for linear regression combined and SVR combined. As in the previous experiment using linear regression as the regression algorithm causes a little time increment (from 0.5 minutes with w=3 to 1.1 minutes with w=30). SVR combined, instead, has a more conspicuous growth of execution time, but lower than using SVR alone. In fact now there are less data for which SVR is trained and on which it predicts: on the stationary data it is used the baseline. For instance, with a window of 30 timeslots SVR alone took 1 hour and 40 minutes, while SVR combined took little more than one hour.

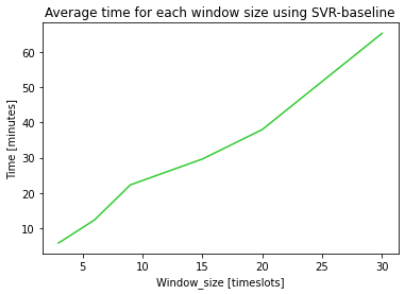
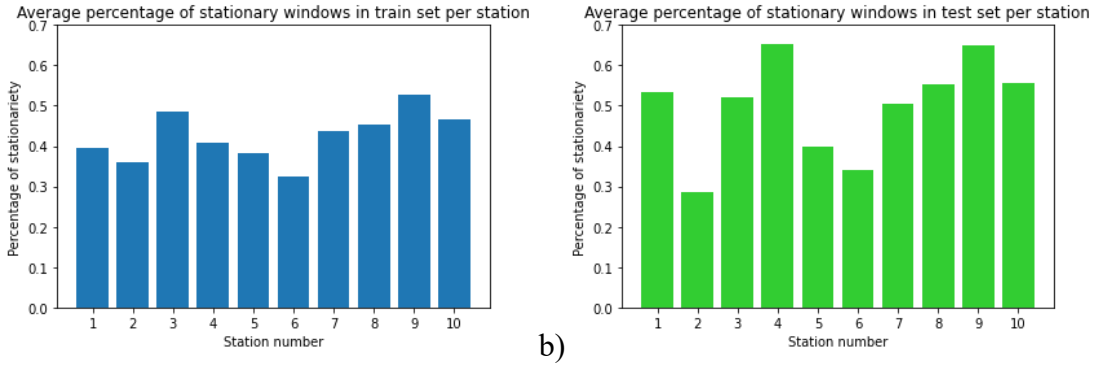


Figure 5-8 Effect of increasing the window size on the training and test time for SVR.

5.2.1 Some considerations about stationarity

The percentage of stationary windows has been calculated for each station and for each configuration of parameters, in order to see how it changes depending on the station, the window size, the forecast horizon and the variance threshold.

In figure 5-9 you can see how stationary the stations are on average (the average is made on the configurations). You notice that the percentages vary between training and test set: in general the test set is more stationary than the training set**.** However, it depends on the station: station 2 is more stationary in the training set, station 5 and 6 have almost the same percentages both in training and test set, all the other stations are more stationary in the test set than in the training. Station 4 and 9 are the most stationary in the test set, and station 4 has the greatest difference in terms of stationarity between the two sets. On average, the percentage of stationarity varies between 0.3 and 0.65.



a) b)

Figure 5-9 Average percentage of stationary windows per station in train a) and test b) set.

This difference between the two sets may be linked to the different behavior of Bicing users in different months of year. The test set contains data in September, while the training set has data in May, June, August and the first week of September. So it is possible that the way of using the service by users in the summer is different from using it in late September.

Then, we would like to study how the stationarity varies as the configuration parameters change.
First, look at the window size. It has been seen that the percentage of stationarity decreases as the window size increases. The graph below shows the percentage of stationarity having averaged on the stations and on the variance threshold and forecast horizon, separately for the training and test sets.

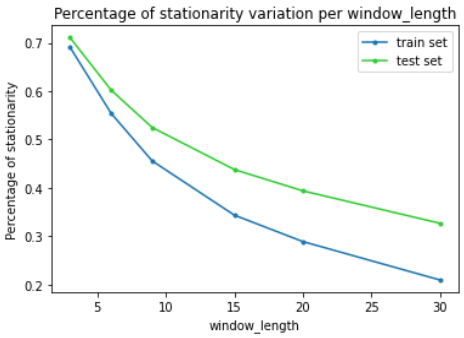


Figure 5-10 Percentage of stationarity per window size.

Then, we can see how the stationarity varies as the threshold on the variance changes. The lower the threshold on variance, the more stringent it is, so the percentage of stationary windows decreases. When the threshold is high, the number of stationary windows increases and the baseline is used more. Also in this case the percentages of stationarity are an average among the stations, the window sizes and the forecast horizons.

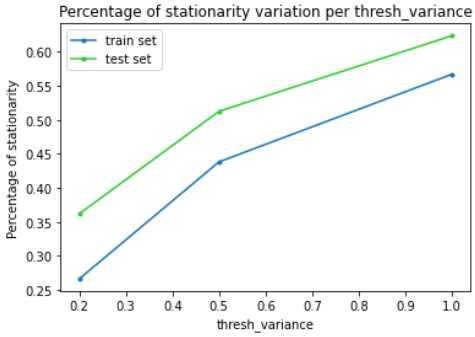


Figure 5-11 Percentage of stationarity per variance threshold.

The variation of the forecasting horizon does not affect the stationarity, for the way in which stationarity is defined. In fact, in the graph below two lines parallel to x-axis. Note that the percentages are an average over the stations, the window sizes and variance thresholds.

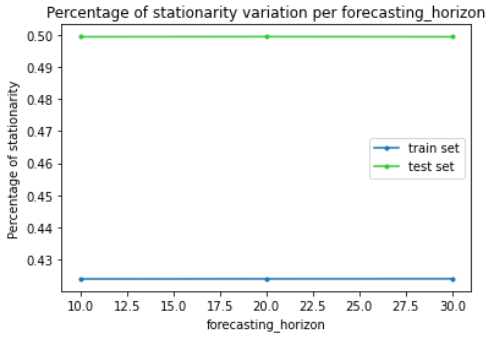


Figure 5-12 Percentage of stationarity per forecast horizon.

Note that all the three plots in figures 5-10, 5-11 and 5-12 provide a confirmation of what stated before, that is the test set is more stationary than the training set.

Also in this experiment it turned out that large stations have large RMSE but also large R2 and intuitively it is more difficult to make predictions on large stations.
Given this, we may want to study and understand better the behavior of the biggest and the smallest stations in more details, as told in the following sub-section, in order to get deeper into this question.

5.2.2 Focused analysis of critical stations

The biggest and smallest stations (station 5, 6 and 9), were analyzed in more details. Station 6 has 33 as the average of total slots, station 5 has 31 slots on average and 9 has 15 slots on average. It is interesting to study the behavior of these stations also because they are the ones with the highest R2 scores (stations 6, 5 and 9) and highest RMSE at the same time (stations 5 and 6). As it was seen in figure 5-9, station 9 is more stationary in percentage than the other two, stations 5 and 6 have low percentages of stationarity.

Some graphs were constructed in order to estimate if stationarity is assigned in a reasonable way, which have time on x-axis and the used slots series on y-axis, depicted with points. Note that a graph has been made for each configuration of:
- Station (values: 5, 6, 9)
- Window Size (3, 6, 9, 15, 20, 30)
- Threshold variance (0.2, 0.5, 1)

The forecast horizon is not considered since it does not have an influence on stationarity for the way in which it has been defined.

In the thesis, not all graphs can be reported, but some of them will be commented and explained.

The graphs provide a visual idea of what just said before: increasing the window size, the number of non-stationary windows increases. Already with w = 9 you can see few stationary windows; for w = 30 the windows in the graphs are practically all non-stationary. By increasing the variance threshold, the number of stationary windows increase: the threshold is less stringent (with a window of 3 timeslots there is a change only passing from 0.2 to 0.5 of the threshold, for larger windows there is a change in any case).

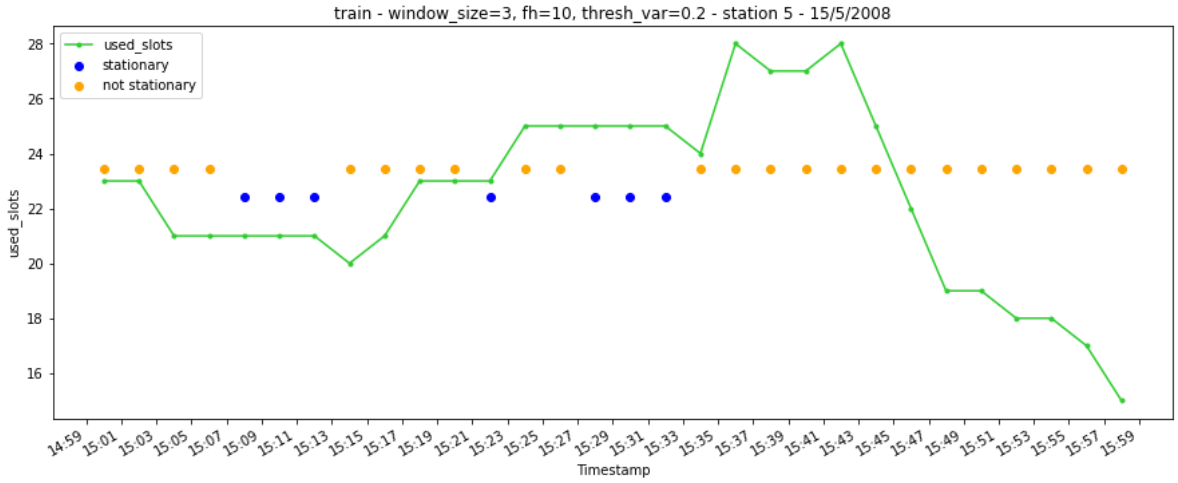


Figure 5-13 Visualization of stationary and non-stationary windows during one hour.

Figure 5-13 shows one of the mentioned graphs. The plot refers to station 5 and to a specific random moment in time (the 15th May 2008 from 15:00 to 16:00) in the training set; window size, forecast horizon and variance threshold are respectively 3 timeslots, 10 timeslots and 0.2.
In the graph the green line represents the values of used slots varying with time; the blue and orange dots mean that a window is stationary or not, respectively. To see a window, you just consider one timeslot t and the preceding two (because w=3 in this case). The values of used slots in these timeslots form a window. Then if you want to know if the window for the timeslots t is stationary or not, you see the color of the blue or orange dot at timeslot t.
For instance, the window at 15:08 has been defined as stationary: in the current timeslot (15:08) and in the two time slots before (15:04, 15:06) used slots is constant, so it is right for it to be stationary.
As a further example, at 15:14 the window is non-stationary: in the current and two previous instants used slots are 20, 21, 21. Having computed the arithmetic mean as 20.67, the variance is

$$s_{n-1}^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} = \frac{(20-20.67)^2 + (21-20.67)^2 + (21-20.67)^2}{3-1} = 0.33 > 0.2$$

so it is right that the window is not stationary as established by the threshold.

The previous plot is just a screenshot of the situation in a very short time interval. To extend the view, the same type of plot was created for a longer period: one day divided into 4 time slots of 5 hours each. Here just two of these graphs are reported, but having generated many of them some general considerations may be made.
The trend of the number of bikes varies with varying times of the day but it is practically impossible to visualize common characteristics between the times of the day by looking at

these graphs. Furthermore, it has been understood that each station is a case of its own, but also we see that for the same time and for the same station on different days the behavior is different.
Another fact that was observed is that a variance threshold of 1 is too high: clearly non-stationary moments are classified as stationary. Finally, with a low window size there are more stationary windows (given the same threshold).
To see it in practice, in figure 5-14 there is one of the plots for station 5, on 18/05/2008 from 5:00 to 10:00, with w=3. At the beginning, from 5:00 to 7:30, there are non-stationary windows alternating with stationary. The same values of used slots but with w=15 generated all non-stationary windows from 5:00 to 7:30, given the same threshold of variance (0.2), as visualized in figure 5-15.

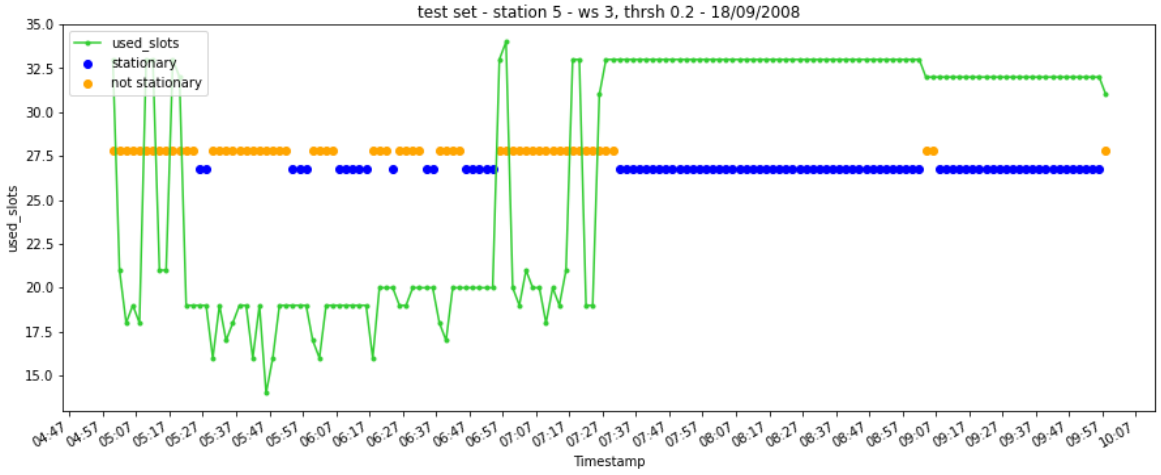


Figure 5-14 Visualization of stationary and non-stationary windows during 5 hours, w=3.

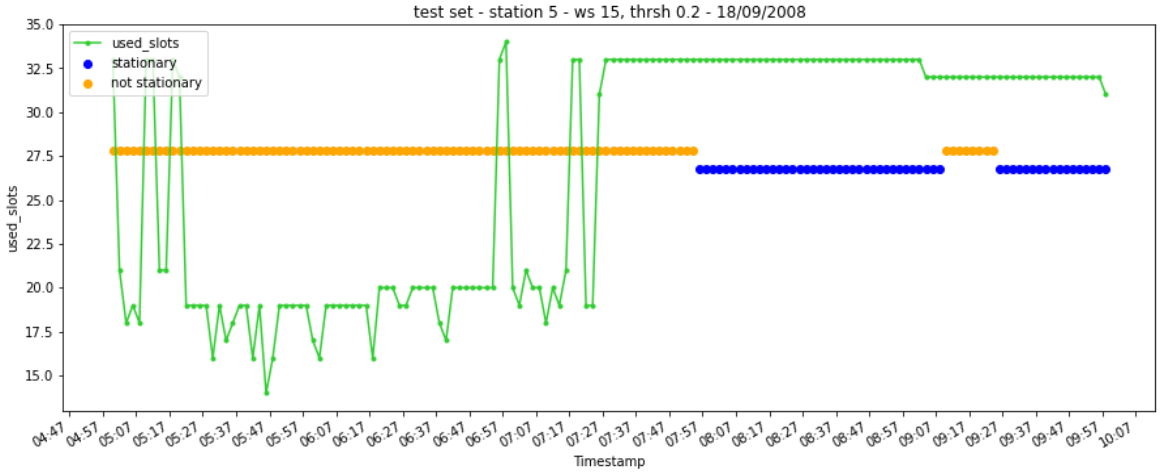


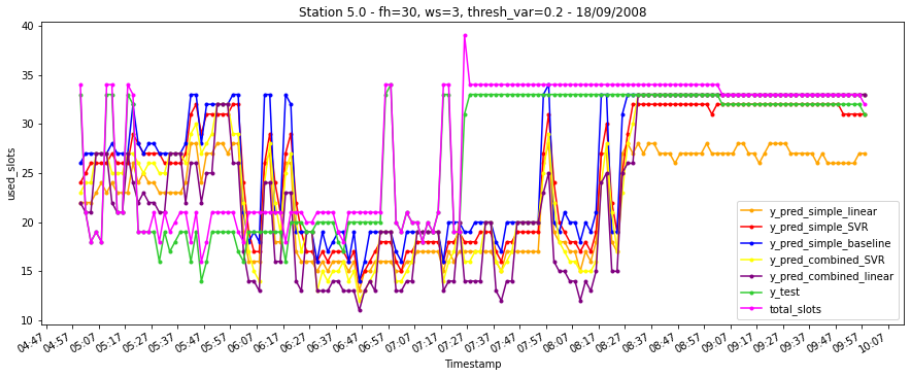
Figure 5-15 Visualization of stationary and non-stationary windows during 5 hours, w=15.

Another type of plot was constructed to understand and visualize how predictions of the various methods differ from the true values, having time on x-axis.
For various bands of the day (5-10, 10-15, 15-20, 20-24) we showed for a random day (always 18/9/2008 as in the other graphs) time variation of:
- Y_pred of linear regression alone
- Y_pred of SVR alone
- Y_pred of the baseline alone
- Y_pred of the combination of baseline and SVR (distinguishing between stationary and non-stationary data)
- Y_pred with the combined baseline and linear regression (distinguishing between stationary and non-stationary data)
- Y_test, i.e. the true values of used slots
- Total_slots

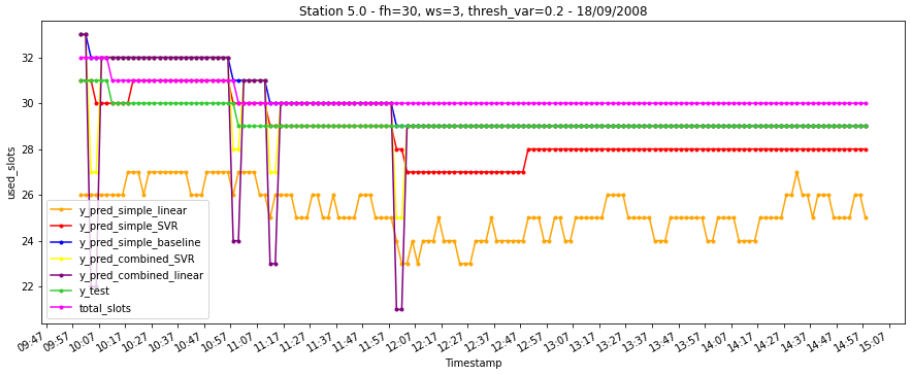
It was created a plot for each combination of station (values: 5, 6, 9), window size (3, 6, 9, 15, 20, 30) and variance threshold (0.2, 0.5, 1), having fixed the forecast horizon to 30. Analog results may be obtained with the other forecast horizons, however it was decided to focus on the h=30 timeslots since it is the most difficult situation to predict.

In figure 5-16 you can see the predictions compared to the target values of used slots, for station 5 during the 18/09/2008 (with window size of 3, variance threshold 0.2 and forecasting horizon of 30).

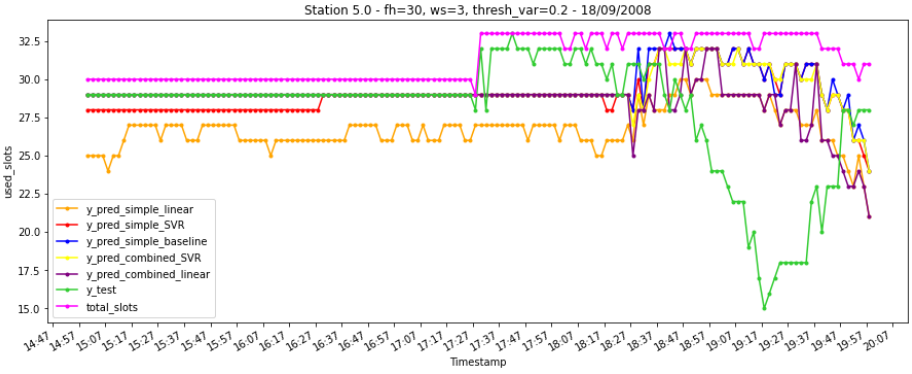
a)



b)



c)



d)

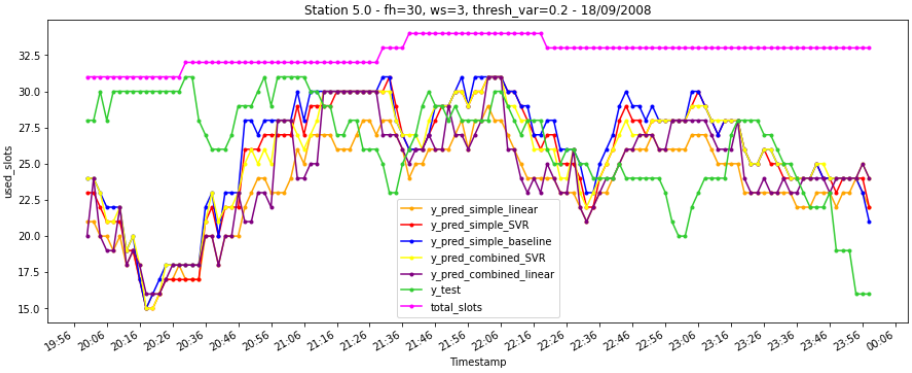


Figure 5-16 Predictions compared to the targets during one day.

It is noted in some of the graphs that y_pred_simple_linear (linear regression alone, in orange) does not predict well when it should predict a constant value (2nd graph): it oscillates and predicts a lower value. Also in other cases it is seen that the orange predicts a lower value than the others and y_test: it seems that linear regression alone is in a narrower range, it does not seem likely to make significant changes up and down, and it stays around an average value.
Combining linear regression and the baseline the oscillation is reduced but there is still a little bit. Combined_linear is fine when using the baseline, but when linear regression comes in, it goes down.

In some cases (for example, in figure 5-15-c, 5-15-d) it was observed that the prediction algorithms are translated forward with respect to y_test and are similar to each other in terms of trend. It seems that predictions arrive later than what it should be: the delay is given by the forecast horizon because algorithms predict a change when they begin to see a change in the past.
In the first graph we see that total_slots varies a lot: this is not a good sign because it means that there is a problem which is unsolvable with data analysis and machine learning, probably due to Bicing map or some sensors in the stations broken. It is impossible to provide predictions if magically the number of total slots for a station goes up instantaneously or it goes down.

To make this work, we have not just to trivially predict used_slots but we must also predict when sensor faults or Bicing map problems could verify or, in any case, we must understand when total_slots is not stable. In theory, these cases where total_slots varies so much should have been avoided with data cleaning but obviously it was not enough: total slot variations after cleaning may be due to interpolation, as there are still peaks in total slots.

5.3 Predicting the number of available slots

As it was shown in exploratory data analysis but also in figure 5-15-a, it is evident that the sudden changes in total slots are due to sudden changes in used slots, rather than in free slots. If total slots goes up a lot but free slots remains constant it means that used slots is noisy. So, it turns out that the noisy variable is used slots that causes total slots to be noisy too. Given this observation, it was thought worth trying to predict free slots instead of used slots, that is predicting the number of available slots rather than the number of bikes present in a station. We hope that performance may improve, thanks to the fact that free slots is less noisy than used slots.

Then, the same two previous experiments were run, by simply changing the target variable to be free slots, instead of used slots.
Unfortunately, predicting free slots did not lead to great performance improvements. R2 and RMSE are practically the same as before, comparing the corresponding methods and algorithms, as well as the graphs for critical stations. This might be related to the fact that free slots is complementary to used slots.
In the end, with free slots there are some fewer anomalies but not so much as to simplify the problem. Since predicting used slots or free slots does not make much difference, we kept predicting used slots in the following experiment, in order to compare the results with those of the first two experiments.

5.4 Adding the difference of previous week

Another strategy for trying to improve performance was adopted. The idea is to add another input feature to make the prediction, that considers possible correlations between the variation of the number of bikes today and in the previous week, given the same time bands and for the same station. Here a strong assumption is made: we may suppose that a station in the same day of week has a similar trend, so we assume that there may be weekly patterns.

At the moment we are considering the features in the past given by the window size. We would like to add to the feature table also a column which represents the delta of used slots one week ago. The concept is explained in more details in the following statements.

In input we have the data back of the window size. For instance, if we want to predict Wednesday at 9.30 and we have a forecast horizon of 30 minutes, we use the data of Wednesday at 9 and going back of the window size. Then we expect that the arrivals and departures of bikes on Wednesday will be similar to the arrivals and departures of Wednesday of the previous week. So we add to the information that there is already a used slots delta which is calculated on the previous Wednesday by doing used slots at 9.30 on previous Wednesday minus used slots at 9 on previous Wednesday. So we see how the number of bikes changed the week before in the same time slot, for the same station.

To see it in practice, for example, we want to make the prediction on 22/05/2008 at 13:40. The feature is calculated as the difference between used slots of 15/05/2008 (one week ago) at 13:20 (which is 6) and used slots of 05/15/2008 at 13:40 (which is again 6), assuming to have a forecast horizon h = 10 timeslots = 20 minutes. So on 22/05/2008 the feature value is the difference 6-6 = 0.
Below we have reported the used slots data from a week before (day 15/05/2008), so you can have a confirmation of the values of used slots at 13:20 and at 13:40.

```
2008-05-15 13:20:00    6.0
2008-05-15 13:22:00    5.0
2008-05-15 13:24:00    6.0
2008-05-15 13:26:00    8.0
2008-05-15 13:28:00    8.0
2008-05-15 13:30:00    8.0
2008-05-15 13:32:00    7.0
2008-05-15 13:34:00    7.0
2008-05-15 13:36:00    6.0
2008-05-15 13:38:00    6.0
2008-05-15 13:40:00    6.0
2008-05-15 13:42:00    5.0
2008-05-15 13:44:00    5.0
```

Figure 5-17 Example for showing the difference with previous week feature: the values of target variable (used slots) on day 15-05-2008, from 13:20 to 13:44.

Then, in figure 5-18 we show the calculated feature. See that at 13:40 the value of the feature is 0, as explained above.

```
2008-05-22 13:20:00    3.0
2008-05-22 13:22:00    4.0
2008-05-22 13:24:00    3.0
2008-05-22 13:26:00    0.0
2008-05-22 13:28:00    0.0
2008-05-22 13:30:00    0.0
2008-05-22 13:32:00    0.0
2008-05-22 13:34:00    0.0
2008-05-22 13:36:00    0.0
2008-05-22 13:38:00    0.0
2008-05-22 13:40:00    0.0
2008-05-22 13:42:00    0.0
2008-05-22 13:44:00    1.0
```

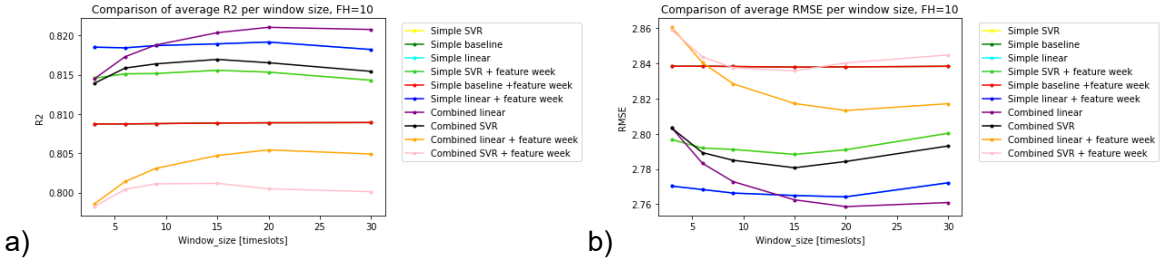
Figure 5-18 Example for showing the difference with previous week feature: the values of the feature on day 15-05-2008, from 13:20 to 13:44.

Note that the data we have in the training and test tables having introduced the difference of previous week feature is not exactly the same as in the first two experiments. This is due to the fact that we were forced to remove the rows of the first week, in the training and test tables respectively. In fact, the added feature has null values for the first week because you do not have a previous week for the first one. As a remark, notice that Nans are present also in the test set because the feature is computed after the train-test split, so training and test tables are separated and you cannot use the last data in the training set for computing features in the test set.
This caused to have high percentages of removed rows in the two sets, but in particular in the test table. In the training table we removed approximately the 16% of rows, while in the test table we deleted about the 33% of data. It seems impressive but consider that the test set contains the measurements in the last three weeks of September 2008, so removing the first week correctly corresponds to removing one third of the data. However, it was not possible to avoid this fact because of the way in which the feature was defined.

Then, the experiment was run both distinguishing between stationary and non-stationary data (combined linear regression and combined SVR) and without this distinction (linear regression. SVR, baseline), as it was made in the second and the first experiment respectively, but now adding the feature of the delta of the previous week.

It was observed that the behavior of performances varying the forecast horizon, the window size and the variance threshold is analogous to that in the first two experiments, respectively. Even stationarity is similar, despite that data is not exactly the same as in the second experiment. It was noticed that the error remains high or it does not increase compared to before.

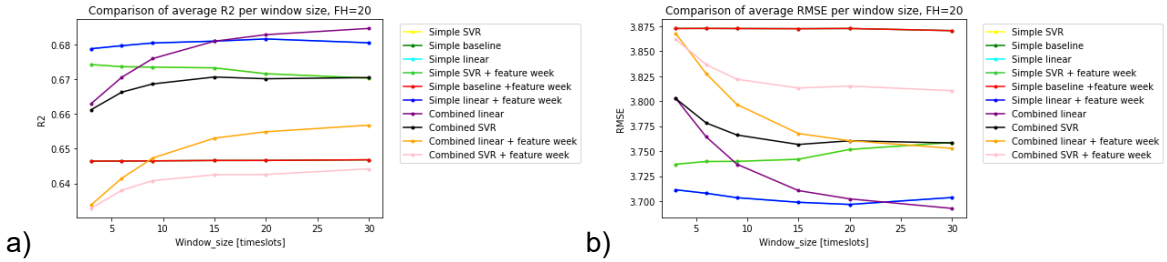
The graphs in figures 5-19, 5-20 and 5-21 compare all proven methods, divided by forecasting horizon. For the combined, an average was made on the variance threshold. A lower threshold worked better but to have no further lines in the graph and to facilitate the reading of the graph we preferred doing the average.
With h=10 timeslots we can see that the combined linear with w=20 performed best: it has the highest R2 score and the lowest RMSE. With a short window size, the simple linear performs best, with or without adding the feature of delta week. Actually, for all the algorithms in their simple version, adding or not the feature that gives the difference of used slots in the previous week in the same time slot did not lead to changes: the first three curves are superimposed on the second three. Instead, when using the combined, adding the feature week leads to worsening: RMSE increases and R2 decreases. The baseline is flat with respect to the window size, it does not change when the window size changes and it has a very high error (one of the highest), but surprisingly not a very low R2 score, in the middle compared to the others. It seems that the best values for the window size are 20 or 15 timeslots, but it depends on the algorithm used. For many of them very long windows causes performance to diminish. Overall, with a small forecast horizon of course performances are higher since the prediction problem is easier; all methods have a R2 in the range 0.79-0.82 and the error varies from 2.76 to 2.86.



a) b)

Figure 5-19 Comparison of all methods through average R2 score a) and average RMSE b), with forecast horizon 10 timeslots.

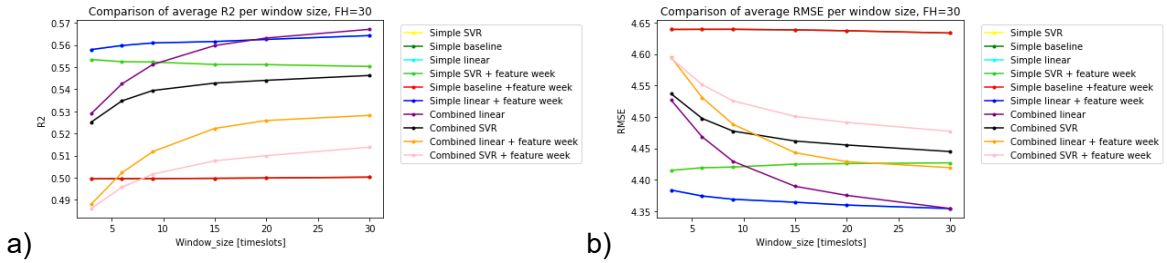
When the forecast horizon increases, the baseline worsens compared to the others. Already with h=20 the baseline becomes the worst in terms of RMSE. However it is not the worst in terms of R2 score, but the combined SVR with the feature week. It is observed that the best method is again the combined linear but with a window size of 30 timeslots. In fact, when the forecast horizon becomes longer, the window size necessary to have best performances increases for the majority of algorithms to 30 timeslots. Simple linear and simple SVR with or without the feature week seem an exception to this trend.
Overall, the performances get worse increasing the forecast horizon: all methods have a R2 in the range 0.63-0.68 and the error varies from 3.70 to 3.88.



a) b)

Figure 5-20 Comparison of all methods through average R2 score a) and average RMSE b), with forecast horizon 20 timeslots.

A forecast horizon of 30 timeslots makes prediction performance to highly deteriorate: the R2 score is between 0.49 and 0.57, while the RMSE is in the range 4.35-4.65. An error of 4 out of 20 bikes or out of 30 bikes is a high mistake.
The baseline gets so bad and the difference between the baseline and the other methods is evident in the RMSE plot: there is difference on the 2nd decimal place. With a high forecast horizon it required to have a longer window size for getting better performances. For almost all methods the best window size is 30 timeslots, except for the simple SVR with or without the feature week for which the error grows when the window size increases. Looking at the RMSE plot (figure 5-21-b) the best methods seem to be the simple linear with or without the feature week and the combined linear both with w=30. However, considering the R2 scores it can be stated that the best is actually the combined linear with w=30.



a) b)

Figure 5-21 Comparison of all methods through average R2 score a) and average RMSE b), with forecast horizon 30 timeslots.

Overall, adding the feature week did not provide improvements: in fact, the performances are the same, in the case of the regressors or the baseline alone, or worse when using the distinction between stationary and non-stationary data (so the combined versions).
Maybe the situation changes a bit from week to week and there are not evident week patterns or the difference within an hour is so small that it does not impact.
Another thing to be considered is that we use the training data in a certain period of the year, then the test data is in September. Those two moments are probably not very related and also how they are used. Having data from a year ago in the same month could help, as well as working with more up-to-date data.

Capitolo 6
Conclusions

From the research conducted it emerged that making predictions on spatial-temporal data is an extremely difficult task due to the nature of the data itself, as it is not simple and immediate to identify the spatial and temporal patterns and relate them, being also different for each case study. The problem was further complicated in the particular case of this thesis because of the use of dirty data and missing information on the domain and the surrounding context, which could have helped to better understand the contingent situations, to improve the efficiency and quality of the model, as well as some suggestions to be taken into account for future work.

On the one hand, the proposed solution turned out to be more performing than the baseline and the more the forecast horizon was increased, the more the difference between the performance of the baseline and the other algorithms increased. With a forecast horizon of 40 minutes, the baseline became the absolute worst in terms of RMSE. On the other hand, the proposed solution has proven to be effective for short forecast horizons and could be improved in the future to ensure more reliable forecasts having a longer time horizon.

It can be said that combining the regression algorithms with the baseline distinguishing between stationary and non-stationary moments has led to improvements in performance compared to the use of regression algorithms alone, with a large enough window. Certainly linear regression turned out to be better than SVR for both execution times and performance. In fact, linear combined algorithm was found to be the best method with all the forecast horizons tested, with a different window size depending on the time horizon. With a forecast horizon of 20 minutes, the best results were obtained with a time window of 40 minutes, while with a time horizon of 40 and 60 minutes, 60 minutes of time window were required. It was therefore found that with a high forecast horizon, it was necessary to have a longer window to achieve better performance.

Considering the fact that as the forecast horizon increases, forecasts become increasingly complicated and performance decreases significantly, it has been seen that in some cases the algorithms provide predictions lagged of the forecasting horizon, as they predict a change when they start to see a change in the past, as they use windows with past data. For

very close forecasts, this problem is limited but when the forecast horizon is increased this delay greatly affects performance, for example for one hour distance forecasts, where performance is significantly deteriorated. To overcome this problem, a solution could be to increase the granularity in the windows by aggregating the data, so that we can consider a larger period in the past without adding dimensionality problems.

Assuming a station on the same day of the week has a similar trend, so assuming there can be weekly patterns, it did not work. Even if the data are not exactly the same as in the other experiments, since those of the first week were not complete, it was seen that there were no improvements: the performances remained the same in the simple version of the algorithms, while they even deteriorated for the combined versions. Perhaps the situation changes a little from week to week and there are no noticeable weekly patterns or the difference within an hour is so small that it has no impact. Eventually it could be experimented the addition of other features, separately or all together, with the difference in the previous 24 hours, the difference in the previous working day or weekend depending on whether the instant to be predicted is a working day or a weekend. In this way, other specific temporal patterns could be identified.

A limitation of the research was that of having very long execution times and limited computational resources and for this reason, unfortunately, the research was forcibly conducted on a subset of BSS stations, building a model for each station. In the future, one could think of using a framework for distributed computing, such as Apache Spark, for the creation of prediction models. On the other hand, also the fine tuning of the parameters of the regression models has been put in the background because it is an expensive process in terms of time and memory and because it would have less impact than the choice of other parameters such as the window size and the forecast horizon. The parameters of the machine learning models were then set to default, as established by the Scikit learn library.

Another thing to consider is that the training data was used at a certain time of one year, approximately from May to August, while the test data is in September. These two moments are probably not very related and also how they are used. It would be interesting to use data of the same type but over a wider time interval so that measurements in the same month in past years can also be considered, as it would be useful to have more up-to-date data, since we were working on 2008 measurements, and also information on weather conditions, as they influence cycling behavior.

However, in the data analyzed, an underlying problem has contributed to making the predictions even more difficult, namely that the number of total slots, i.e. the size of the stations, changes over time. Forecasting thus becomes an almost unsolvable problem with mere data analysis and machine learning, because you cannot know when sensor failures or BSS map problems might occur, or when bikes are redistributed. The data cleaning was not sufficient to remove the variations of the used and total slots after cleaning, due to the interpolation of used slots. Indeed, sudden changes in total slots would be due to sudden

changes in slots used, rather than free ones, but even predicting free slots did not lead to changes in performance.
One solution would be to add to the stations dataset also the information on their total slots because it should be a constant number over time or in any case in the short term. In this way you would know at any moment how many sensors are not working, knowing the number of occupied and free slots in each station. To do this, however, it would be necessary to make an inspection and then work with current data or to find the information in some other dataset perhaps available on the internet. Incorporating data on the redistributions of bikes made with trucks, therefore their schedules and the number of bicycles transported in each of these schedules, would provide valuable information to improve predictions, knowing when there will be abrupt changes in the number of bikes available.

Finally, more precise forecasts could be obtained by knowing the origin and destination of individual users, but these are sensitive data that cannot be disclosed in Europe as required by the GDPR. In other countries outside the European Union, studies have been carried out using this type of data, which makes us reflect on the ethical issues of protecting the privacy of individual citizens and on certain limits that should not be crossed when using machine learning techniques on data that affect people more or less directly.

List of Tables

List of Figures

References

[1] R. Alvarez-Valdes, J. M. Belenguer, E. Benavent, J. D. Bermudez, F. Muñoz, E. Vercher e F. Verdejo, «Optimizing the level of service quality of a bike-sharing system,» *Omega*, vol. 62, pp. 163-175, 2016.

[2] J. Brownlee, «Time series forecasting supervised learning,» 2016. [Online]. Available: https://machinelearningmastery.com/time-series-forecasting-supervised-learning/ .

[3] R. Asadi e A. C. Regan, «A spatio-temporal decomposition based deep neural network for time series forecasting,» *Applied Soft Computing*, 2019.

[4] «e-Handbook of Statistical Methods,» NIST/SEMATECH, [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc41.htm.

[5] G. Bontempi, S. B. Taieb e Y.-A. Le Borgne, «Machine Learning Strategies for Time Series Forecasting,» *Lecture Notes in Business Information Processing*, 2013.

[6] T. Cerquitelli e E. Baralis, «Data Science Lab. Time series analysis: fundamentals,» DataBase and Data Mining Group, 2019.

[7] «Time Series,» Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Time_series.

[8] R. J. Hyndman e G. Athanasopoulos, Forecasting: Principles and Practice, Monash University, Australia: OTexts, 2013.

[9] S. Russell e P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Upper Saddle River, New Jersey: Pearson Education, 2010.

[10] U. Thissen, R. van Brakel, A. P. de Weijer, W. J. Melssen e L. M. C. Buydens, «Using support vector machines for time series prediction,» *Chromometrics and intelligent laboratory systems*, n. 69, pp. 35-49, 2003.

[11] A. Kaltenbrunnen, R. Meza, J. Grivolla, J. Codina e R. Banchs, «Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system,» *Pervasive and Mobile Computing*, vol. 6, pp. 455-466, 2010.

[12] C. S. Song e W. C. H. Han, «A real-time short-term traffic flow adaptive forecasting method based on ARIMA model,» *Acta Simulata Systematica Sinica*, vol. 16, n. 7, pp. 1530-1456, 2004.

[13] Z. Li, Y. Li e L. Li, «A comparison of detrending models and multi-regime models for traffic flow prediction,» IEEE Intelligent Transportation Systems Magazine, vol. 6, n. 4, p. 34–44, 2014.

[14] P. Vogel, T. Greiser e D. C. Mattfeld, «Understanding Bike-Sharing Systems using Data Mining: Exploring Activity Patterns,» *Procedia - Social and Behavioral Sciences*, vol. 20, pp. 514-523, 2011.

[15] M. V. D. Voort, M. Dougherty e S. Watson, «Combining Kohonen maps with ARIMA

time series models to forecast flow,» *Transportation Research Part C: Emerging Technologies*, vol. 4, n. 5, p. 307–318, 1996.

[16] K. Lin, Q. Lin, C. Zhou e J. Yao, «Time Series Prediction Based on Linear Regression and SVR,» 2007.

[17] Z. Yang, J. H. Hu, Y. Shu, P. C. Cheng e T. Moscibroda, «Mobility Modeling and Prediction in Bike-Sharing Systems,» in *MobiSys '16: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 2016.

[18] B. Wang e I. Kim, «Short-term prediction for bike-sharing service using machine learning,» in *International Symposium of Transport Simulation (ISTS'18) and the International Workshop on Traffic Data Collection and its Standardization (IWTDCS'18)*, 2018.

[19] Y. Zhang, T. Thomas, M. Brussel e M. van Maarseveen, «Exploring the impact of built environment factors on the use of public bikes at bike stations: Case study in Zhongshan, China,» *Journal of Transport Geography*, vol. 58, pp. 59-70, 2017.

[20] C. Sardinha, A. C. Finamore e R. Henriques, «Context-aware demand prediction in bike sharing systems: incorporating spatial, meteorological and calendrical context,» 2021.

[21] S. Deng, S. Jia e J. Chen, «Exploring spatial–temporal relations via deep convolutional neural networks for traffic flow prediction with incomplete data,» *Applied Soft Computing*, vol. 78, pp. 712-721, 2019.

[22] Y. Pan, R. C. Zheng, J. Zhang e X. Yao, «Predicting bike sharing demand using recurrent neural networks,» *Procedia Computer Science*, vol. 147, pp. 562-566, 2019.

[23] J. Froehlich, J. Neumann e O. Nuria, «Measuring the Pulse of the City through Shared Bicycle Programs,» I*nternational Workshop on Urban, Community, and Social Applications of Networked Sensing Systems*, 2008.

[24] Z. Zheng, Y. Zhou e L. Sun, «A Multiple Factor Bike Usage Prediction Model in Bike-Sharing System,» in *International Conference on Green, Pervasive, and Cloud Computing*, 2018.

[25] Y. Zhou, Y. Li, Q. Zhu, F. Chen, J. Shao, Y. Luo, Y. Zhang, P. Zhang e W. Yang, «A reliable traffic prediction approach for bike-sharing system by exploiting rich information with temporal link prediction strategy,» John Wiley & Sons Ltd, 2019.

[26] M. Vafaeipour, O. Rahbari, M. A. Rosen, F. Fazelpour e P. Ansarirad, «Application of sliding window technique for prediction of wind velocity time series,» *Int J Energy Environ Eng*, 2014.

[27] H. S. Hota, R. Handa e A. K. Shrivas, «Time Series Data Prediction Using Sliding Window Based RBF Neural Network,» *International Journal of Computational Intelligence Research*, vol. 13, n. 5, pp. 1145-1156, 2017.

[28] «Bikeoff Project – Design Against Crime, July 2008 : Bicing Barcelona, Clear Channel Public,» [Online].

[29] «Mapa de disponibilitat,» Bicing, [Online]. Available:

https://www.bicing.barcelona/mapa-de-disponibilitat.

[30] «Bicing,» Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Bicing.

[31] «https://commons.wikimedia.org/wiki/File:Furgo_bicing_bcn.JPG,» [Online].

[32] «Faqs y normas,» Bicing, [Online]. Available: https://www.bicing.barcelona/es/faqs-y-normas.

[33] «Bicing,» Bicing, [Online]. Available: https://www.bicing.barcelona/es.

[34] «Tarifas,» Bicing, [Online]. Available: https://www.bicing.barcelona/es/tarifas.

[35] «https://geohack.toolforge.org/geohack.php?language=it&pagename=Barcellona&params=41.3825_N_2.176944_E_type:adm3rd_scale:1000000&title=Barcellona,» [Online].

[36] «Great Circle,» Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Great_circle.

[37] «Haversine Formula,» Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Haversine_formula.

[38] «Haversine Distances,» Scikit Learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.haversine_distances.html.

[39] «Great Circle Distance,» Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Great-circle_distance.

[40] «Blog Bicing Watch,» [Online]. Available: https://blog.bicingwatch.com/2008/07/.

[41] J. Veenstra, «How do I clean time series data,» Quora, 2018. [Online].

[42] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina e R. Banchs, «Bicycle cycles and mobility patterns - Exploring and characterizing data from a a community bicycle program,» 2008.

[43] «Linear interpolation,» Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Linear_interpolation.

[44] Google, «https://cloud.google.com/vertex-ai/docs/datasets/bp-tabular,» [Online].

[45] «Holidays-es,» [Online]. Available: https://pypi.org/project/holidays-es/.

[46] H. S. Hota, R. Handa e A. K. Shrivas, «Time series data prediction using sliding window based RBF neural network,» *International Journal of Computational Intelligence Research*, vol. 13, n. 5, pp. 1145-1156, 2017.

[47] T. G. Dietterich, «Machine learning for sequential data: a review,» Oregon State University, Corvallis, Oregon, USA.

[48] «Glossary: Forecast horizon,» Eurostat, [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php/Glossary:Forecast_horizon.

[49] F. X. Diebold, Forecasting in Economics, Business, Finance and Beyond, University of Pennsylvania, 2017.

[50] «Understanding support vector machine regression,» MathWorks, 2021. [Online].

[51] A. J. Smola e B. Schölkopf, «A tutorial on support vector regression,» *Statistics and*

Computing, vol. 14, pp. 199-222, 2004.

[52] S. Shalev-Shwartz e S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.

[53] N. E. Helwig, *Multivariate Linear Regression*, University of Minnesota, 2017.

[54] K. P. Murphy, Machine Learning: A Probabilistic Perspective, Massachusetts Institute of Technology, 2012.

[55] J. L. Devore, Probability & Statistics for Engineering and the Sciences, Brooks/Cole, Cengage Learning, 2012.

[56] «Multiple Linear Regression,» Standford University, [Online]. Available: https://web.stanford.edu/class/stats202/notes/Linear-regression/Multiple-linear-regression.html.

[57] «Linear Model,» Scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html.

[58] «Advantages and disadvantages of linear regression,» OpenGenus IQ. [Online].

[59] D. Chicco, M. J. Warrens e G. Jurman, «The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation,» *PeerJ Computer Science,* 2021.

[60] «R2-score,» [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score.

[61] «Coefficient of determination,» Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Coefficient_of_determination.

[62] «Root mean square deviation,» Wikipedia, [Online] Available: https://en.wikipedia.org/wiki/Root-mean-square_deviation.

[63] «https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d,» 2016. [Online].

[64] S. Perera, «https://medium.com/making-sense-of-data/time-series-next-value-prediction-using-regression-over-a-rolling-window-228f0acae363,» [Online].

[65] R. J. Hyndman e A. B. Koehler, «Another look at measures of forecast accuracy,» *International Journal of Forecasting*, vol. 22, n. 4, pp. 679-688, 2006.

[66] «e-Handbook of Statistical Methods,» NIST/SEMATECH, [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm .

Acknowledgements

Colgo l'occasione per ringraziare tutte le persone che mi hanno accompagnata in questo piacevole percorso, dalle piccole comparse fino ai profondi affetti. È stata oltre che una esperienza accademica, anche una occasione di crescita personale. Nonostante le parole concise richieste dalla formalità del contesto, ci tengo a precisare che si vorrebbe dire ben di più rispetto a quanto espresso.
Ringrazio il prof. Garza e l'ing. Colomba, che con la loro gentilezza e competenza mi hanno supportata nella realizzazione di questo lavoro di tesi. Grazie per tutto il vostro aiuto e la vostra disponibilità. Se dovessi scegliere di nuovo argomento e relatore per la tesi, rifarei la stessa scelta.
Un grande grazie alla mia famiglia, a mia nonna, ai miei genitori e a mio fratello per avermi sempre sostenuto, anche nei momenti più difficili, con pazienza, dedizione e dolcezza, senza di voi non sarei qui.
Grazie a Luca… Hai reso speciale la semplice quotidianità.
Infine vorrei ringraziare tutti i miei amici. Grazie per i bei momenti trascorsi insieme e per le grandi risate.

Un abbraccio,
Bianca