POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

An open machine learning framework for residential water consumption estimation

Tesi di Laurea Magistrale in
Computer Science and Engineering -
Ingegneria Informatica

Author: **Loris Panza**

Student ID: 967915
Advisor: Prof. Andrea Francesco Castelletti
Co-advisors: Prof. Dr. Andrea Cominola (TU Berlin), M.Sc. Wenjin Hao, M.Sc. Siling Chen (TU Berlin)
Academic Year: 2021-2022

Abstract

Water management is a critical topic in various parts of the world due to different environmental and anthropic factors affecting future water security. Globalization and demographic pressure are increasing water demand in highly urbanized regions, making water management in such areas critical for sustainable development. Despite infrastructure investments and technological advances, managing a city's water resources remains a complex task due to various issues. One of the most significant issues is certainly the lack of reliable data on residential water consumption, which limits our knowledge on current and future water demands. Such data is essential for designing and implementing effective water management policies and programs, from reducing any losses to identifying high-demand areas or developing water conservation programs. Unfortunately, many countries lack adequate monitoring systems, and even when they exist, the collected data is often incomplete or unreliable. On the other hand, public water consumption data collected through censuses and surveys have low spatial resolution and temporal frequency, limiting the accuracy of analysis and progress in sustainable water resource management. This thesis proposes an innovative approach to address the lack of data: the daily water consumption of residential structures is estimated using only public data and machine learning techniques. After a careful analysis of different Convolutional Neural Networks and training techniques, two architectures were chosen: one for filtering Google Street View images of building's facades and another for estimating the relative height. Combining that information with the building area and socio-demographic data, XGBoost reaches the best performance in predicting daily water consumption among various machine learning algorithms evaluated. The work demonstrated the validity of the machine learning techniques developed in the methodology overcoming the accuracy of approximative formulas and highlighting the potential of public data in supporting sustainable water management.

Keywords: Water Demand, Open Data, Sustainability, Machine Learning, Deep Learning

Abstract in lingua italiana

La gestione dell'acqua rappresenta un tema critico in varie zone del mondo dovuto a diversi fattori ambientali ed antropici. All'incapacità di accedere e distribuire l'acqua, sono soprattutto i paesi a basso reddito che non hanno la capacità economica di effettuare gli investimenti necessari per infrastrutture adatte. La globalizzazione e la crescente pressione demografica aumentano la domanda nelle regioni più urbanizzate, rendendo la gestione dell'acqua più rilevante ai fini di uno sviluppo sostenibile. Nonostante i progressi tecnologici, la gestione idrica di una città rimane ancora un compito complesso dovuto a problematiche di diversa natura. Tra queste una delle più significative è la mancanza di dati affidabili sui consumi d'acqua a livello residenziale. Essi sono fondamentali per progettare e implementare politiche e programmi efficaci per la gestione idrica: dalla riduzione di eventuali perdite all'identificazione di aree ad alta richiesta. Purtroppo, molti paesi non dispongono di sistemi di monitoraggio adeguati e i consumi idrici pubblici rilevati tramite censimento e sondaggi hanno una risoluzione spaziale e una frequenza temporale bassa che limita la precisione dell'analisi e i progressi nella gestione sostenibile delle risorse idriche. In tale contesto, il lavoro di tesi propone un approccio innovativo nella risoluzione della mancanza di dati: si è riusciti a stimare il consumo d'acqua giornaliero di strutture residenziali utilizzando dati pubblici e tecniche di Machine Learning. Dopo un'analisi di diverse Reti Neurali Convoluzionali e tecniche di addestramento, sono state selezionate due architetture: una per filtrare le immagini Google Street View raffiguranti edifici ed un'altra per estrarne la relativa altezza. Unendo tale informazione con l'area dell'edificio e dati socio-demografici, XGBoost raggiunge le migliori prestazioni nella stima del consumo d'acqua giornaliero tra i vari algoritmi analizzati. Lo studio svolto ha dimostrato la validità delle tecniche di Machine Learning superando l'accuratezza di formule approssimative ed evidenziando la potenzialità dei dati pubblici nel supporto ad una gestione sempre più sostenibile dell'acqua.

Parole chiave: Consumo Idrico, Dati Pubblici, Sostenibilità, Machine learning, Deep learning

Contents

1 | Introduction

Although the global availability of freshwater is sufficient to meet present and future water needs, it is not uniformly distributed across space and time. In several areas (e.g., Djibouti, Ethiopia, Kenya), freshwater resources are insufficient to satisfy domestic, economic, and environmental demands. In these regions, the lack of clean water significantly constrains human health and productivity, thereby affecting economic development and environmental sustainability. Future projections foresee an increase of global water demand by 55% in 2050 (Connor, 2015). This increment will further intensify the existing pressure on natural resources and ecosystems (Hussey and Pittock, 2012). Furthermore, climate change is likely to exacerbate water scarcity at a global level by increasing the frequency and intensity of such extreme events. High water demands are usually concentrated in urban areas, which can experience water stress due to their high population density and water-intensive activities. As urbanization and global population growth persist, water usage in residential buildings is expected to increase, posing infrastructural and operational challenges to utilities and municipalities (Lambin and Meyfroidt, 2011). In recent years, the focus of urban water management has thus shifted from merely developing infrastructure in response to urban growth to implementing sustainable and cost-effective approaches for combined urban water supply and demands (Diaz et al., 2016; Brown et al., 2009).

Knowledge of when, where, and how people use water is essential to model water demands and inform demand management strategies. However, lack of data on residential water consumption at the household level often limits our knowledge on past and current water demands with high spatial and temporal resolution. A primary source of difficulties is the absence of water consumption monitoring infrastructure (i.e., water meters) which, when available, consists of analog sensors that do not automatically log and transmit data on water consumption. The absence of standardization in data collection and reporting is another source of issues. In many countries, data gathering on residential water consumption is sporadic and inconsistent, preventing comparison across regions or time periods. Furthermore, water consumption data are often stored by water utilities, but are hardly accessible for research purposes, when available (Di Mauro et al., 2021). Finally, census or

periodic household surveys are run with low temporal frequency, precluding policymakers and researchers from relying on up-to-date information for designing effective water demand management interventions. Without reliable data, it is difficult to identify the most significant drivers of water consumption, identify water consumption patterns and magnitudes, and evaluate the effectiveness of different water-efficient technologies and water conservation practices.

This thesis addresses the above challenges by developing an open machine learning framework for residential water consumption estimation. We estimate residential building daily water consumption relying exclusively on publicly available data. As input features for our machine learning framework, we obtain three types of data after carefully examining the variables affecting home water usage, along with the state-of-the-art techniques to acquire them from publicly available data. The first is a Google Street View (GSV) image of the building of interest, which captures the exterior characteristics of the building and offers relevant information about its extension. The relevant GSV photos are classified using a pre-trained Convolutional Neural Network (CNN), and those that reveal recognizable buildings are then fed into a second CNN regression model to determine the corresponding building height. Second, the area of each building is extracted through a public shapefile retrievable through the Milan geoportal (GeoportaleMilano, 2012). It contains spatial information about the study area, such as the location of the buildings, their shapes, areas, and footprints. By incorporating this information in addition to the height, the model can account for differences in water usage attributed to building size, which is particularly relevant in densely populated urban areas where structure dimensions can vary widely. Socio-demographic changes have become essential factors in understanding the increase in water demand, particularly in the domestic sphere, as these factors represent and reflect the household size, water use behaviour, and water-saving practices (Matos et al., 2014). As a third source of data, socio-demographic information regarding the district where the building is placed is fed into a machine learning model for the daily water consumption estimation. Five different regression algorithms were evaluated: Linear Regression, Polynomial Regression, K-Nearest Neighbour, Random Forest and Extreme Gradient Boosting.

This thesis showcases the viability of employing machine learning and deep learning methodologies for estimating water consumption at the building level and discusses the pros and cons of utilizing openly accessible data sources for similar studies.

Thesis outline

This thesis is organized as follows:

- Chapter 2 provides a comprehensive review of the existing state of art methods that exploit public data to retrieve information on relevant socio-demographic determinants of residential water consumption, with highlights on their strengths and limitations;
- Chapter 3 describes the open data-based machine learning framework developed in this thesis, with a primary focus on the explanation of the methods and models used from a theoretical and practical point of view;
- Chapter 4 presents the data, starting with the description of the study site followed by the water consumption time series analysis and the introduction of the other features used as targets or fed-in input for the implemented models. It also shows the training setup for the different implemented models;
- Chapter 5 describes and discusses the results of the test set and the comparison of the performances achieved by different models, as quantified with different accuracy metrics;
- Chapter 6 summarizes the findings, implications and limitations of the research, suggesting future improvement directions;

2 | State of art the review

2.1. Water demand determinants

Water is an increasingly stressed natural resource. Growing concerns about the impacts of climate change have firmly emphasized the need to plan and manage water resources judiciously to reduce waste (Bates et al., 2008). A vital component of urban water system planning is water demand. Knowledge of current and future water demands enables better operation and management of the system and fosters long-term water conservation (Cominola et al., 2021). Climate, socio-economic, psychological, institutional, and management factors all play a role in how much water a city needs (see, e.g., Toth et al., 2018; Cominola et al., 2023). However, this relationship is complicated, likely nonlinear, and unknown in many cases and high-resolution scales. It is imperative to identify the key factors (also known as determinants) among these to predict water demand and to plan and manage water resources and supply systems. Estimation of water demand of residential housing has been pursued from a variety of approaches and has been a major issue in water management and policy as well as environmental economics, as investigated by several recent studies (e.g., Matos et al., 2014; Arbués et al., 2003; Suh et al., 2012; Cominola et al., 2023). The above studies surveyed empirical residential water demand analyses concerning the main variables that can affect residential water demand such as the impact of price, income, population, interannual climate variability, housing types, apartment complex types and household composition of residential households. The most widely utilized population data worldwide are official *census data*. However, they are typically obtained with low spatial detail and low temporal frequency, making this information often not updated and much aggregated. Moreover, in some countries with poor or unstable political conditions, this data may not be available (Wardrop et al., 2018). Consequently, a lack of precise population data has become an obstacle for governors, planners, and researchers in heterogeneous urban areas. The increasing power of geographical information systems (GISs), massive remote sensing products, and multiple geospatial big data have improved socio-demographic data estimations, especially at the regional and global scale (Leyk et al., 2019). In this chapter, some articles related to the

extraction of socio-economic features from open data useful for water demand estimation will be analyzed by comparing different possible methods and sources of data.

2.2. Social sensing and remote sensing data

The use of *social sensing data* to delineate socio-economic characteristics is a growing trend in the recent decade (Liu et al., 2015). With the development of information and communications technology, various kinds of sensors produce big data such as vehicle trajectories, public transportation card records, cell phone location and call records and user profiles from the social network. These data, describing people's daily activities, have abundant spatio-temporal information and have been widely used to analyse spatio-temporal patterns of human socio-economic activities and potential interaction among different activities. Data from location-based social media are being used increasingly to investigate anthropogenic activities and their effects on the environment. The interaction of human activities with the physical environment also gives essential information for estimating human volume concentration (e.g., Zhang et al., 2019). Physical characteristics in built-up structures, greening covers, surface temperatures, and atmosphere emissions are greatly affected by and influence how people communicate, produce, and live. This unravels the possibility of estimating human activity volumes based on a glimpse of the physical environment, which is traceable through remote sensing (RS) techniques. The RS observations comprehensively capture the physical environment properties of the Earth's surface and have advantages in terms of low acquisition cost and broadly scanned coverages, including areas with low-frequency census or sparse network base stations. The objective is to help limited location data to estimate human activities using RS images, given the high correlation to complex landscape sceneries. An example of RS data is Nighttime light (NTL) data, which has been acknowledged as a valuable source to reveal human activities and have been widely applied to the estimation of population due to its advantages in large detection range and high temporal resolution (e.g., Townsend and Bruce, 2010; Zeng et al., 2011). Recent studies investigated the relationship between NTL and socio-economic indicators and demonstrated the correlation between light intensity and the population density and relative consumption at the national and urban scales (Keola et al., 2015; Elvidge et al., 1997; Amaral et al., 2006). However, using the nighttime light data alone to estimate the population may cause overestimation problems due to excessively high light radiance in specific types of areas, such as commercial zones and transportation hubs. In this light, it is worthwhile using various data sources to improve the accuracy of population estimation (Wang et al., 2020). For that reason, in addition to NTL data, social sensing data have been used for improving the detection of population

surface distribution and enhancement.

Open data such as NTL and point-of-interest (POI), extracted from an online map service provider application, have been used to map urban nighttime leisure space (UNLS) (e.g., Liu et al., 2020). UNLSs, referring to the spaces where urban residents engage in leisure activities between 6:00 P.M. and 6:00 A.M., have broad impacts on urban economic growth, employment, urban competitiveness, urban vitality, and social justice. The distribution, geometrical features, and functional qualities of UNLSs were inferred using the NTL pictures and POI data. Additionally, it is possible to pinpoint the geographic range and limits of UNLSs, which enables researchers to examine their morphological traits and associated markers. Although the approach can generate a UNLS distribution map for a given urban area, there are limitations related to the final model; the parameters used to combine the NTL and POI data, and linear model are based mainly on trial-and-error experiments and empirical knowledge. Hence, these parameters may be arbitrary and subjective to a certain degree. In addition, a linear model may not be the best choice for combining two kinds of data. Adopting machine learning (ML) techniques to obtain more reasonable parameters to enhance the model accuracy and to introduce nonlinear relation between the data sources will be an improvement carried out by other studies.

Apart from using POI, another research (Yu et al., 2019) attempts to combine NTL images, taxi trajectory data (a kind of popular social sensing data) and census data to spatialize the population. Some studies (e.g., Zheng et al., 2011) have used taxi trajectories among the various types of social sensing data sources because of their capacity to describe urban inhabitants' travel habits and represent the locations of their homes, businesses, and transit hubs. Taxi trajectory data describe human movements, which has great potential in modifying the errors made by excessively high light radiance. For instance, the taxi flow at night may indicate migrations from dining/entertainment locations with exceptionally high light brightness to residential zones or vice versa. While NTL images can reflect some 'static' features of human activities, taxi GPS trajectories have a great capability for delineating 'dynamic' characteristics of human movements. By utilizing NTL images and taxi GPS trajectory data, the study provided a new way to estimate population at fine scales. Using the initial population distribution grid estimated by nightlight radiance and the population calibration grid derived from the net inflow of people given by the taxi trajectory, it is possible to calculate the estimated population grid.

The rapid development of remote sensing technology allows us to get images with high-resolution satellite images (VHR). The research developing appropriate and efficient methods for scene presentation and classification becomes critical in the remote sensing image community. The high dimensionality and the statistical characteristics of these acquired

images are challenging for VHR scene understanding (Romero et al., 2016). It is possible to see a growing potential regarding the use of remote sensing techniques to supply socio-economic information for planning urban supply and disposal infrastructure. Specifically, object-based image analysis techniques have been used to detect single buildings (e.g., Grippa et al., 2017), and recent research has implemented those techniques to measure urban ecosystem functionality and to indicate the quality of life factors. As a case, the correlations between different socio-economic conditions and different recognized and classified building types are evaluated (Warth et al., 2020). Based on VHR remote sensing imagery, single buildings are detected and classified into eight different categories. The supervised classification is performed based on the reference information collected during the field survey and through a variety of attributes that are computed through the building detected polygon. A random forest classifier is trained on 25 features to predict the type of building, and this is employed to classify all buildings in the city. The key assumption underlying the approach of gathering relevant data for supply and disposal infrastructure planning is that different socio-economic groups of households living in different building types have different habits and lifestyles. That results in different construction materials and energy flows. To prove this correlation, the different socio-economic statuses of the households in the study area had to be determined and classified before being assigned to different building types. The establishment of the social, economic point as an index to describe residential socio-economic backgrounds delivers good results by combining information on expenditure, educational level, assets, and household income. Through statistical analysis of resident interviews, a relationship between building types and socio-economic groups can be established, and the socio-economic status can be inferred based on building type information.
Overall, social sensing and remote sensing data show a meaningful impact in describing land use and socio-economic studies but none of the previously discussed research related that kind of data to water consumption management or estimation.

2.3. Deep learning for socio-economic features extraction

Due to the high penetration of location-based services (LBS) into a variety of daily activities, including instant communication, navigation, online business, and entertainment, geographical coordinates of most activity locations can be automatically recorded every time subscribers send location requests and authorize LBS-based mobile applications. Even though this information stands well for the spatial distribution of individuals' daily

activities, access to social sensing data is often impeded by privacy issues and enterprise data sharing concerns (de Montjoye et al., 2013). Given this lack of access to social sensing data, advanced analytic techniques have been developed and applied to remote sensing data for the estimation of socio-economic features. Recent techniques exploit remote sensing imagery through an *end-to-end deep learning* framework for reliable estimates of socio-demographic features. Deep Convolutional Neural Networks (DCNNs) (LeCun et al., 2015) are appropriate algorithms to capture hidden hierarchies and non-linearities of geographical patterns. For instance, RS satellite images have been used as input to the CNN, achieving accurate estimations of the human activity volume (Xing et al., 2020). The architecture implemented is called *Neighbor-ResNet* (Figure 2.1) which is an end-to-end deep learning model with spatially neighbour augmentation.

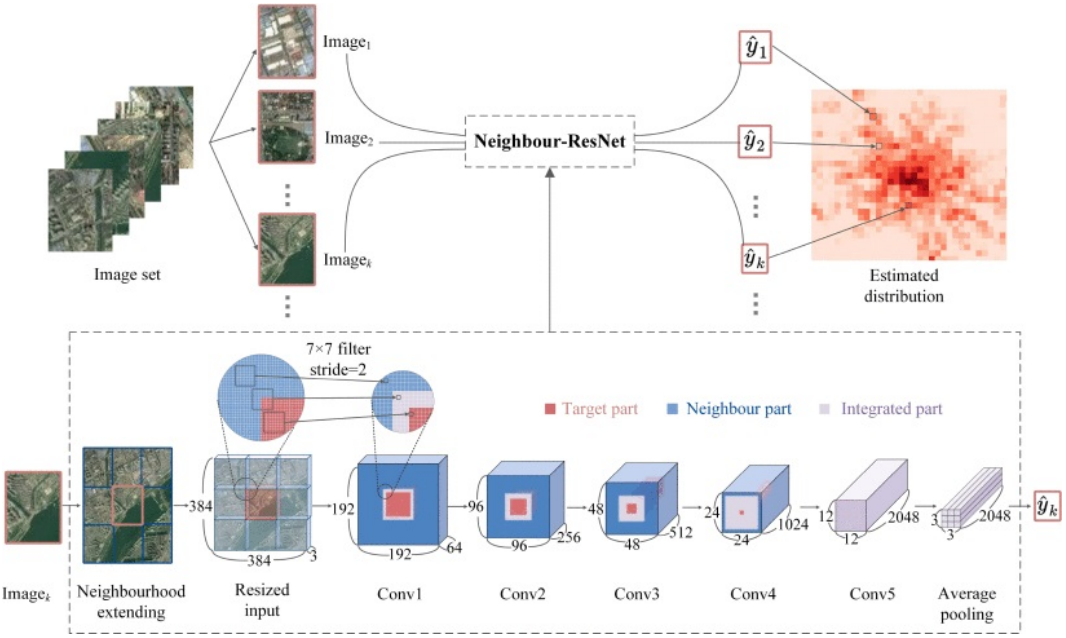


Figure 2.1: End-to-end framework of Neighbor-ResNet using RS images (Source: Xing et al., 2020).

The proximity of the environment has been proven to significantly impact the centric land cover, welfare, attractiveness, dynamics, and subsequent human-environment interactions. Due to their spatial linkages, neighbour features are essential for determining the centric targets for most geographical phenomena. It is determined that adding the neighbour impact to the end-to-end model significantly improves the model performance using RS photos as inputs and LBS data as labels in several cities. The input expansion explicitly uses the spatial relationships between the centre and neighbour tiles, as opposed to deepening the input channels or concatenating the feature vectors of parallel

picture patches. Since the volume data are in raster formats, they fix neighbour patches located at the eight nearest units of a target sample. Surrounding information is included in those neighbour RS tiles. The model summarizes individual knowledge of the target and neighbours and assembles their features via layer-wise convolutional operations. The *convolution filters*, sliding on feature maps of each layer, can extract interior characteristics when covering network cells with only target or neighbour information. Conversely, it can integrate them into adjacent parts. The resulting gradient-weighted class activation mapping (Grad-CAM) quantifies the relative contribution of the input pixels to the estimation and highlights salient ground features. Through image feature assembly and high-level feature generation, the model identifies subtle differences in regional layouts and reveals heterogeneous landscape imprints of human activity. Depending on the RS methodology, local socioeconomic features, and specific soil characteristics, some factors limit model performance. Building height and distinctive building appearance are one of the key variables that reflect human activity, but 2D scanned RS-Pictures provide limited information. For that reason, satellite images are important references for urban planning and urban studies. Still, given the heterogeneity of urban land-use types, it is difficult to differentiate different land-use types based on overhead remotely sensed data. For example, the overhead view of remotely sensed data only captures the spectral reflectance of building roofs, which can hardly reflect the different social functions or land-use types of buildings. Unlike the overhead remotely sensed data, Google Street View images capture the profile views of cityscapes, providing us with a new data source for urban studies from a very different perspective (e.g., Li and Zhang, 2016). An application of deep learning computer vision technique has been developed on GSV to determine socioeconomic statistics and political preferences in the US population (Gebru et al., 2017). This study presents a method for estimating the socioeconomic characteristics of regions across 200 US cities using 50 million images of street scenes captured by GSV vehicles. They use a CNN architecture to identify the brand, model, and year of every car encountered in a given region. Data from this census of motor vehicles, which enumerated 22 million automobiles in total (8% of all automobiles in the United States), were used to accurately estimate income, race, education, and voting patterns at the zip code and precinct level. The resulting associations are surprisingly simple and powerful. Using that approach, zip code or precinct level survey data collected for a few cities can be used to provide up-to-date demographic information for many American cities automatically. Thus, the fully automated computer vision methods applied to publicly available street scenes can extract social, economic, and political patterns in neighbourhoods across the US with low computational demands.

2.4. CNN: feature extraction and end-to-end learning

A CNN is composed of two parts: one being the feature extraction part, where the input image is reduced to a set of feature maps through a series of strategically arranged convolution and *pooling layers*, and another being the classifier where the features are passed into a series of fully connected or hidden layers and an output layer. The combination of these feature extractors and *fully connected layers* is called the end-to-end CNNs. In some cases, the high-level feature extracted from the pre-trained CNN model has substantial distinguishable power in classifying their objective and shows that the choice of the classifier is not significant in the related application. As an example, Jean et al. (2016) shows how a Convolutional Neural Network (CNN) can be trained with survey and satellite data to identify image features that can explain up to 75% of the variation in local-level economic outcomes in five African countries. The rationale for this approach relates to the inability of using NTL data to distinguish between differences in economic activity in regions living near or below the international poverty line. Nightlight images are potentially less useful for studying and tracking the livelihoods of poor countries. First, the CNN model has been pretrained on ImageNet, a large image classification dataset that consists of labelled images from 1000 different categories. Next, based on the knowledge gained from this image classification task, the CNN is fine-tuned on a new task, training it to predict the nighttime light intensities corresponding to input daytime satellite imagery. Nightlights are a noisy but globally consistent—and globally available—proxy for economic activity. By solving this related proxy task, the model learns how to extract features useful for the poverty estimation task. Finally, they use mean cluster-level values from the survey data and the corresponding image features extracted from daytime imagery by CNN to train ridge regression models that can estimate cluster-level expenditures or assets. The approach does not depend on nightlights being able to make this distinction and instead uses nightlights only as intermediate labels to learn image features correlated with economic well-being. In areas where such data are available, combining the derived features with other passively gathered data may also improve household and cluster-level predictive power. The model is, on average, significantly more predictive of variance in consumption and assets than nightlights alone, despite having been trained partially on nightlights. This further evidence suggests that the picture characteristics contain information beyond what nightlights provide.

Considering the previously discussed approach, a new research by Yeh et al. (2020) illustrates a different strategy: unlike the earlier method, which used nighttime light intensity

as intermediate labels for training a CNN feature extractor on daytime imagery, it incorporates both sets of imagery (nighttime and daytime), using models trained separately on daytime and nighttime images and then joined in a final fully connected layer. In other words, they concatenate the final layers of the separate daytime and nightlights models and trained a *ridge-regression model* on top. Without first specifying which elements it should look for, the model seeks to discover patterns in the daytime and nighttime photos indicative of asset wealth. These results exceed performance in earlier work on a similar task using high-resolution imagery (Jean et al., 2016) or mobile phone data (Blumenstock et al., 2015) as input and match or exceed benchmarks for in-country performance from geostatistical models used to predict health outcomes, standards of living, and housing quality in Africa. Consequently, the approach of directly using nightlight images as model inputs performs better than using them indirectly as a proxy, as in an earlier transfer learning approach. However, while the CNN-based approach outperforms approaches to poverty prediction that use simpler features common in the literature, the information the CNN uses to make a prediction is less interpretable than these simpler approaches, perhaps inhibiting adoption by the policy community. A key avenue for future research is improving the interpretability of deep learning models in this context and developing approaches to navigate this apparent performance interpretability tradeoff.

2.5. Water demand estimation with RS data

Apart from extracting interesting socio-economic features, RS data are used to estimate the resource demand directly. Multispectral satellite images and nonlinear topographical features, along with resource consumption survey data, have been used to estimate local water consumption from multispectral satellite images (Mohanty et al., 2022). To accomplish the task, the first step is the classification of building and non-building pixels in the study area through a raster ground-truth image from the training dataset: it consists in a binary classification that indicates building pixels as white pixels (positive class) and non-building pixels as black pixels (negative class). After creating the ground truth, they improve the feature space to add more context to each pixel and the information fed into the classifier. The feature space is also made more nonlinearly complex by the oriented gradients derived around each pixel from the histogram of the directed gradients descriptor. The Histogram of Oriented Gradients is a popular feature descriptor used for object detection and image processing (Dalal and Triggs, 2005). The second step is classifying residential and non-residential building pixels from the identified ones. Classifying building pixels according to building type can help to approximate emissions and resource consumption from an area captured by a satellite image since resource consumption by

buildings is a function of the type of buildings. Residential and commercial buildings require different quantities of water (and other) resources. They assume that residential and non-residential buildings have different emission/consumption rates per unit area and remain constant across all buildings of one type. The building type (residential or non-residential) was annotated manually using visual interpretation metrics, e.g., large building size, roofing materials, proximity to building clusters, roads, and industrial structures like storage tanks, etc. The final stage is to estimate how much water is used in the photographed area. The area covered by residential buildings ($\boldsymbol{A}_R$) and the area covered by non-residential buildings ($\boldsymbol{A}_{NR}$) is estimated using the following probabilistic formula (Mohanty et al., 2022):

$$\boldsymbol{A}_R = \sum_{pixel\, i} (a_p \times P\{i \in B\} \times P\{i \in R \mid i \in B\}) \tag{2.1}$$

$$\boldsymbol{A}_{NR} = \sum_{pixel\, i} (a_p \times P\{i \in B\} \times P\{i \in NR \mid i \in B\}) \tag{2.2}$$

where, a_p is the area of one pixel in m^2, B, R and NR, denote building, residential and non-residential, respectively. This figure is then weighted by the consumption per unit area (Star, 2012) to calculate the water consumption (W):

$$\boldsymbol{W} = \boldsymbol{A}_R \times \boldsymbol{W}_r + \boldsymbol{A}_{NR} \times \boldsymbol{W}_{NR} \tag{2.3}$$

where, $\boldsymbol{W}_{NR}$ and $\boldsymbol{W}_{NR}$ are the water consumption per unit area for residential and non-residential buildings, respectively. They use a simple closed-form expression that utilizes the result of the building type classification and the water usage data to approximate the daily water consumption from the buildings observed in the image spanning 1 km^2.

2.6. Research challenges

Although a lot of research has been carried out to extract socio-economic and other interesting economic features from public data to overcome privacy and lack of data issues, few studies really investigate the possibility of relating these data to the water consumption analysis or prediction in an urban context. The open-research challenges that have motivated this study are listed here:

- The first challenge is the resolution of water consumption prediction. Data on residential water consumption at such a high-resolution level is particularly scarce for two reasons. First, water consumption data is the property of water consumers

and is not shared for privacy reasons. Second, several water usage influential factors (such as socio-economic conditions), which may be used as a proxy to model water consumption, are challenging to collect in an automated fashion and, therefore, mostly unavailable on a residential building scale.

- The second challenge is relying only on public data. Many studies succeed in accurately estimating water usage at a very high-resolution level using ground-truth data or census/surveys in the study area. The aim of the thesis is to be dependent only on data such as Google Street View Images that could be easily extracted to make the final methodology reproducible in different contexts.
- The final challenge is to understand the relationship between water usage and its determinants: even though many projects investigated the elements affecting water consumption in residential buildings, there still not exists a clear formula explaining how these factors interact with each other to produce the final outcome.

3 | Methodology

This chapter presents the algorithms implemented and gives the details of each section of the methodology pipeline to help understand the approaches used to reach the final objective.

3.1. Methodology pipeline

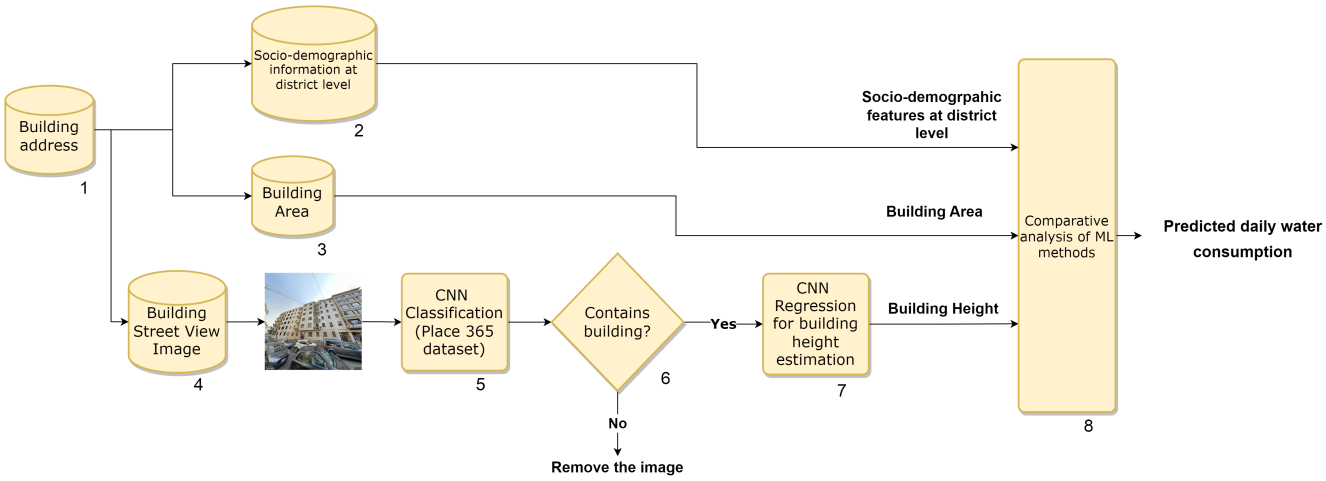


Figure 3.1: Methodology pipeline for daily water consumption estimation.

Figure 3.1 illustrates the flowchart of the methodology. The procedure consists in models that separately process the street view images, socio-demographic features, and open building data given a certain building address. The models result in different sets of features which are then combined and fed into a machine learning model that estimates the daily water consumption of the building. Starting from a specific building address (step 1 in Figure 3.1), the Google Street View image is extracted using the relative API (GoogleStreetViewAPI, 2023) (step 4). Since some street view images are unsuitable, i.e., buildings are blurred, or an image only shows streets or vegetation, a CNN pre-classification is applied to select relevant images, i.e., those that show clearly identifiable buildings (steps 5, 6). In a second step, images classified as relevant are fed into CNN

regression models for the building height estimation (step 7). To this end, both custom-built and pre-trained architectures have been explored using different techniques. The evaluation was based on two fundamental metrics: the mean squared error (MSE) and the mean absolute error (MAE) coefficient. Building characteristics have been found to influence urban water use (Guhathakurta and Gober, 2007): most prominently, these include exterior building features, such as building height, volume, compactness, and wall materials. For these features, ground-level images such as GSV ones that depict the extent and textures of building walls and façades are more informative. Also, the age of a dwelling can affect water demand, as older homes are more likely to have appliances and fixtures that are less water efficient than newer homes. However, data on the aforementioned building features is often scarce for two reasons. First, they contain private information that is typically not shared publicly. Second, data collection of these features is difficult to automate and, therefore, typically requires extensive manual labor. For these reasons, only the height of the building is inferred by the CNN while the relative area is retrieved from the public shapefile (step 3) of the study area as will be explained in Chapter 4. In addition to the area and height, public socio-demographic data such as population density related to the study area where the building addresses are provided are considered (step 2). Based on those features, different ML models were applied to comparatively predict the daily water consumption of the residential buildings (step 8). So, the problem was framed as a regression task evaluated through metrics such as MAE and the coefficient of determination (R2). However, as explained in Chapter 3, the target variable was discretized into different consumption levels, leading to a change in the problem type from regression to classification. Consequently, the evaluation metrics were also changed to reflect the new problem type, with metrics such as accuracy, precision, recall, and F1 score used to evaluate the performance of the classifiers. Each methodological phase of Figure 3.1 is explained in detail in the next sections, after a gentle introduction to neural networks and convolutional neural networks, which are used in several steps of the whole methodology.

3.2. Theoretical background on Neural Networks and Convolutional Neural Networks

Here some theoretical key concepts are introduced to better understand the methodology. Particularly, neural networks and convolutional neural network concepts will be introduced in the next sections.

Neural Networks

A neural network is a nonlinear model that can be used for *regression* and *classification* tasks. In a classification task, the objective is to decide which of the available class should be assigned to a given input. Let Δ be the set of the possible classes and $t \in \Delta$ the *target* class for input $x \in \mathbb{R}^N$. The objective is to find a function $Y : \mathbb{R}^N \to \Delta$ such that:

$$Y(x) = t \tag{3.1}$$

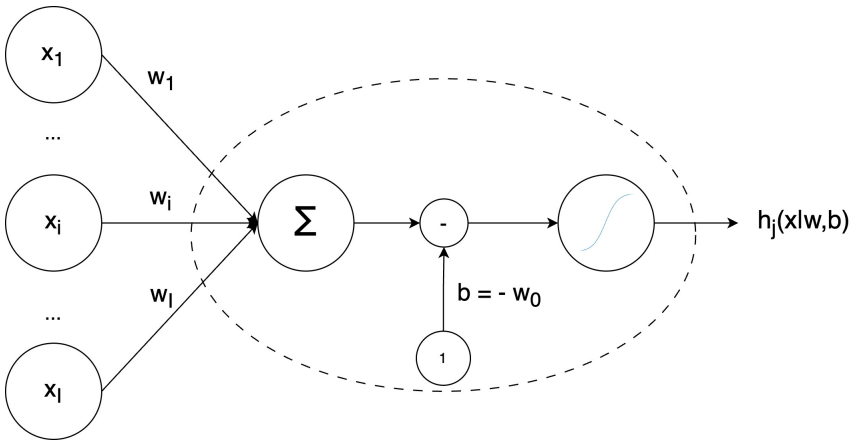


Figure 3.2: Perceptron architecture.

The basic computational unit of the brain and neural network is a neuron. A *perceptron* or neuron (Figure 3.2) computes a linear combination of its input by applying a nonlinear transformation h, called *activation function*. The activation function is a transfer function that is used to produce the desired output for the problem designed. It is used to map the input values in a certain domain depending on the function that is used. Among all the possible functions (such as sigmoid, hyperbolic tangent, etc.) the Rectified Linear Unit (ReLU) is the mostly used one: the goal of this layer is to cancel out all the negative values and transform the output from a linear computation, such as convolution, to a nonlinear one. The ReLU function is defined as:

$$f(x) = max(0, x) \tag{3.2}$$

leading to an output range $[0, inf)$.
Not all the inputs have the same importance and this is represented by weighting each input with a weight w_i. Additionally, every neuron has an additional input whose value is fixed and it is called *bias*. In literature, it is often referred to as w_0. Therefore, the

output of a neuron with inputs $x =< x_1, \ldots, x_I >$, weights $w =< w_1, \ldots, w_I >$ and bias $b = -w_0$ is defined as:

$$h_j(x|w, b) = h(\sum_{i=1}^{I} w_i x_i - b) \tag{3.3}$$

As shown in Figure 3.3, neural networks are modeled as collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons can become inputs to other neurons. If the graph is directed and acyclic, then the architecture is called a *feed-forward* neural network.

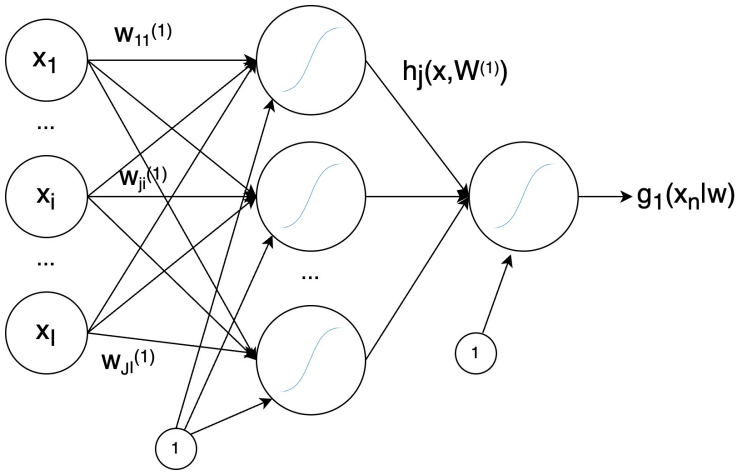


Figure 3.3: Fully-connected and feed-forward neural network architecture.

Instead of an amorphous blob of connected neurons, neural network models are often organized into distinct layers of neurons. The layers can be grouped in three typologies: an *input layer* containing the input neurons, one or more intermediate *hidden layer*, and an *output layer* containing all the output nodes. For regular neural networks, the most common layer type is the *fully-connected layer* (FC) in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections. Notice that when n-layer neural networks are intended, the input layer is not counted. Therefore, a single-layer neural network describes a network with no hidden layers (input directly mapped to output). Unlike all layers in a neural network, the output layer neurons most commonly have a unique activation function: the last output layer is usually used to produce the class scores (e.g., in classification), which are arbitrary real-valued numbers, or some kind of real-valued target (e.g., in regression).
One of the most important features of neural networks with at least one hidden layer is that they are universal approximators. It has been shown (see Cybenko, 1989) that given

any continuous function $f(x)$ and an error $\epsilon > 0$, there exists a Neural Network g(x) with one hidden layer (with a reasonable choice of nonlinearity, e.g., sigmoid) such that $\forall x, |f(x) - g(x)| < \epsilon$. In other words, the neural network can approximate any continuous function. Increasing the size and number of layers in a neural network can increase the capacity of the network; the space of representable functions grows since the neurons can collaborate to express many different functions. Let's focus on a feed-forward neural network composed of a single hidden layer, like the one shown in Figure 3.3. The activation functions of the J neurons of the intermediate layer apply a nonlinear transformation h with weights $W^{(1)} =< w_{11}^{(1)}, \ldots, w_{JI}^{(1)} >$ and bias $W_0^{(1)} =< w_{01}, \ldots, w_{0J} >$, while the neurons of the output layer apply a nonlinear transformation g_1 with weights $W^{(2)} =< w_{11}^{(2)}, \ldots, w_{1J}^{(2)} >$ and bias $w_0^{(2)}$. Let w be the set of all the weights of the network. Given an input whose components are $x_n =< x_1, \ldots, x_I >$, the output $g_1(x_n|w)$ is:

$$g_1(x_n|w) = g(\sum_{j=1}^{J} w_{1j}^{(2)} \cdot h(\sum_{i=1}^{I} w_{ji}^{(1)} x_i + w_{0j}^{(1)}) + w_0^{(2)}) \tag{3.4}$$

Training a neural network means finding the optimal values of the weights in order to minimize some predefined *loss functions*. There exist many types of loss functions depending on the specific task. Typically, one of the most used loss functions for regression tasks is the *Mean Squared Error* (MSE), here indicated with notation $E(w)$ and formulated for a set of weights w as:

$$E(w) = \sum_{n=1}^{N} \frac{(t_n - g_1(x_n|w))^2}{n} \tag{3.5}$$

where (x_n, t_n) is a pair input, target. The goal is to find a set of weights w that minimizes the loss function for all the pairs (x_n, t_n) of the *training dataset*. Notice that a neural network is a nonlinear model and the loss can be a non-convex function: finding its minimum is not trivial and the method used to find the right set of weights is called *backpropagation*. The weights of the network are updated using *stochastic gradient descent*. For each of them in the network, it is needed to subtract from the relative value the gradient of the loss function multiplied by a parameter α called *learning rate*:

$$w^{k+1} = w^k - \alpha \frac{\partial E}{\partial w}|_{w=w^k} \tag{3.6}$$

where w_k is the value of the weight for the current iteration while w_{k+1} is the value of the weight for the next iteration. The *forward pass* refers to the calculation and storage

of intermediate variables (including outputs) for a neural network from the input to the output layer: it's traversing through all neurons from the first to last layer and a loss function is calculated from the output values. And then *backward pass* refers to the method of calculating the gradient of neural network parameters: the method traverses the network in reverse order, from the output to the input layer, according to the *chain rule*. Backward and forward pass makes together one *iteration*. During one iteration, one usually passes a subset of the dataset, which is called *minibatch* or *batch*. An *epoch* defines the training process of the neural network using the entire dataset in batches.

Convolutional Neural Networks

Despite the ability of feed-forward neural networks' ability to approximate every continuous function with a single hidden layer, this type of architecture is not always the best choice for given mapping. It has been established that some tasks are better suited to Deep Neural Networks (DNN), i.e., feed-forward neural networks composed of multiple hidden layers (Lecun et al., 1998). *LeNet*, the network proposed in the paper, is an example of this (Figure 3.4).

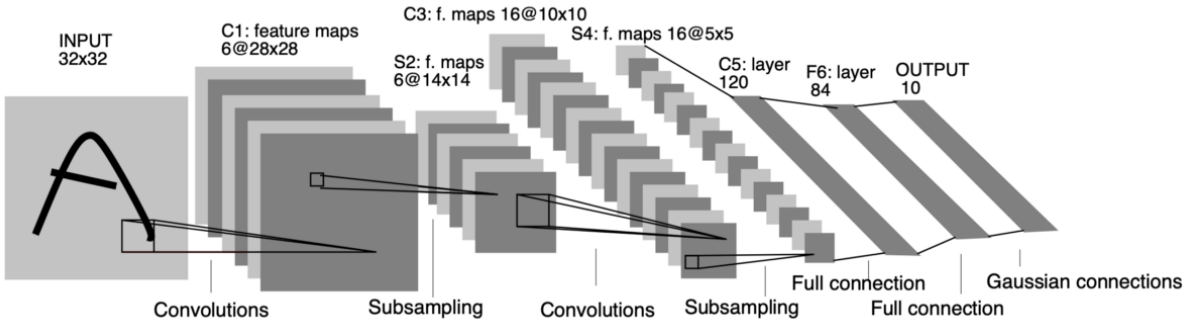


Figure 3.4: LeNet architecture (Source: Lecun et al., 1998).

Deep Neural Networks are an excellent tool for image recognition, which can assign a label that describes the input image. This task is typically achieved by a specific kind of DNN, called Convolutional Neural Network. This model is mainly composed of different kinds of layers (e.g., *max pooling*, *fully-connected*) but the main one is the *convolutional layers*. Let's first introduce the convolution operation to better understand how it works. Convolution is a filtering operation that consists in applying to an image a *filter* - or *kernel* - of size N applied to the pixel in position (r, c) of the image. Conceptually, by applying the convolution operation, multiplication is performed between the value of the pixel of a neighbourhood of the target position with the value of the filter. More specifically, let's consider an input image I of width W and height H, composed by only one channel (i.e.,

it has a depth of 1). This can be represented as a $1 \times W \times H$ tensor. After applying to it a filter w of size N, represented by a $1 \times N \times N$ tensor, the output of the convolution O applied to position (r, c) is a linear combination of the image and the filter:

$$O(r+c) = \sum_{x,y=-\lfloor \frac{N}{2} \rfloor}^{\lfloor \frac{N}{2} \rfloor} w(x,y) \cdot I(r+x, c+y) \tag{3.7}$$

In equation 3.7, x and y represent the indices of the filter kernel and are used to shift the filter window over the image I in both the vertical and horizontal directions. By applying the same kernel to every pixel of the input image, a filtered image is obtained. The application of a filtering operation is shown in Figure 3.5.

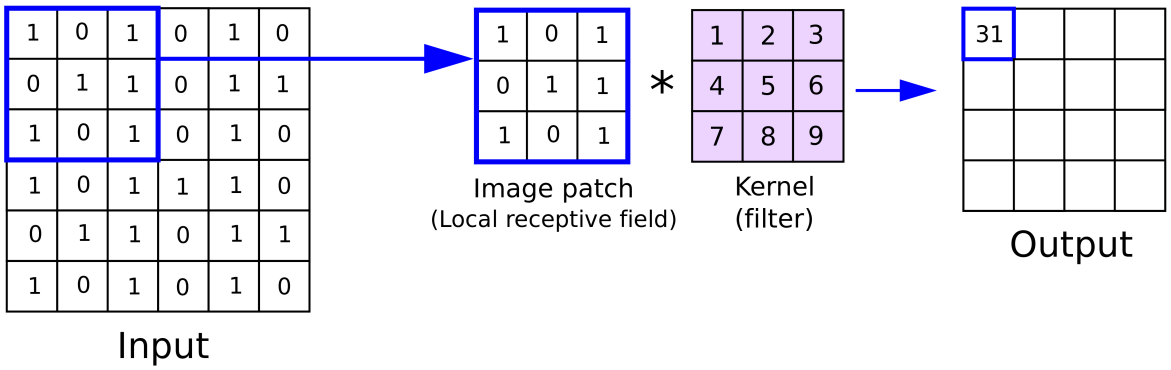


Figure 3.5: Filter functionality example (Reynolds, 2019).

Since the convolution is a linear combination of its inputs, it is easily implementable in a neuron that works as a convolutional filter. A network in which some of its neuron act as a convolutional filter is called Convolutional Neural Network. The resulting output volume after a convolutional layer is called *feature map*. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a Convolutional Neural Network might have size $5 \times 5 \times 3$ (i.e., 5 pixels width and height, and 3 because images have depth 3, the colour channels). During the forward pass, each filter is convolved across the width and height of the input volume and computes dot products between the entries of the filter and the input at any position. As the filter is slid over the width and height of the input volume, it will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that are activated when they see some type of visual features such as an edge of some orientation or a blotch

of some colour on the first layer. Now, let's consider an entire set of filters in each conv layer (e.g., 12 filters), and each of them will produce a separate 2-dimensional activation map. These activation maps will be stacked along the depth dimension and produce the output volume. While each filter must have the same depth as the input image, the depth of the output is dictated by the number of filters applied. While convolutional layers greatly increased the depth of the volume and slightly decreased its spatial extension, the pooling layers are used to reduce the spatial extension. The pooling layer operates independently on every depth slice of the input and resizes it spatially, often using the max operation: this class of layers is called *maxpooling*. Max pooling is a pooling operation that calculates the maximum value for patches of a feature map and uses it to create a downsampled (pooled) feature map. It is usually used after a convolutional layer. The most common form is a pooling layer with filters of size 2×2 applied with a stride of 2: it downsamples every depth slice in the input by 2 along both width and height (Figure 3.6).

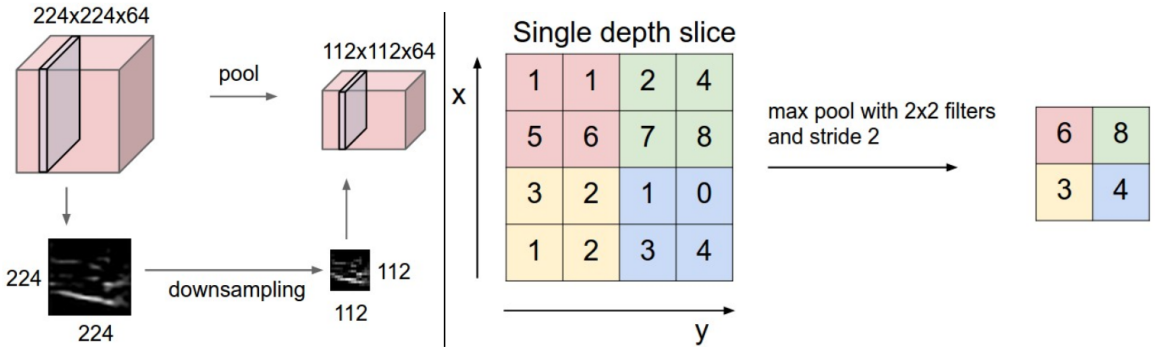


Figure 3.6: Max pooling functionality example (Source: Stanford, 2022).

Apart from convolutional and pooling layers, also fully-connected layers that have full connections to all activations in the previous layer are used in CNN, as seen in regular neural networks. Placed at the end of the CNN, the FC layers are then in charge of the generation of the output of a CNN. These layers receive as input the image manipulated by the previous layers and produce a vector of N dimension where N corresponds to the number of outputs. It is worth noting that the only difference between fully-connected and convolutional layers is that the neurons in the convolutional layer are connected only to a local region in the input and that many of the neurons in a convolutional volume share parameters. The local region in the input refers to a fixed-size rectangular area of the input tensor that is processed by a single filter. However, the neurons in both layers compute dot products, so their functional form is identical. Therefore, it turns out that it's possible to convert between fully-connected and convolutional layers.

After this brief introduction to neural networks and convolutional neural networks, we can proceed through the implemented methodology.

3.3. GSV image acquisition

Each available GSV image can be requested in an HTTP URL form using the GSV Image API along with the position of the GSV car and its moving direction. By defining URL parameters sent through a standard HTTP request using the GSV Image API, users can get a static image in any direction and at any angle for any point where GSV is available. An example of GSV static image request is shown at this link: `http://maps.googleapis.com/maps/api/streetview?size=640x640&location=VialeBrenta35,Milano&fov=120&pitch=25`. In this example url, the parameter "size" specifies the output size of the requested GSV image (640×640 pixel), "location" provides the geo-location of the GSV image (the GSV Image API will snap to the panorama photographed closest to this location), "pitch" specifies the up or down angle of the camera relative to the street view vehicle, and "field of view (fov)" determines the horizontal field view of the image. With this url, a GSV static image can be retrieved, as shown in Figure 3.7. Optionally, a parameter "heading" can be included in the url, which indicates the compass heading of the camera (the heading values range from 0 to 360). If no header is specified, a value will be calculated that directs the camera to the specified location value, from where the closest photo is taken. Using the GSV Image API, a list of addresses is directly submitted and then the retrieved house images are stored locally. This process avoids the potential accuracy problems introduced by geocoding procedure and successfully obtains all the house images except invalid street addresses. As the images are shot from streets, they usually contain the target house in the middle as well as parts of adjacent buildings on two ends. The API parameters were tuned to obtain the exact region of the target house. Considering the central field of vision for most people covers an angle of between 50°and 60°(Yang et al., 2009), for the research, the fov and the pitch were respectively set to 120 and 25; thus, images can cover almost all the horizontal and vertical extensions of the building. In particular, the fov should be assigned appropriately because a wide view will introduce neighbour buildings and a narrow view will only capture partials of the target building. Each retrieved building image is in 640×640 pixels, which is the largest size that GSV API provides.

Figure 3.7: Example of a GSV static image.

3.4. GSV image selection: Places365-VGG16

After the image acquisition, a pretrained CNN was used to filter the GSV images; the objective is to remove invalid images, such as the interior of buildings and those in which facades had been obscured by large vehicles (e.g., buses) or greenery. Also, images that were too dark or addresses not captured by GSV service were excluded. Figures 3.8a and 3.8b show examples of invalid images. The CNN used is called Places365-VGG16 and it is a CNN pretrained on a subset of *Place dataset*, a quasi-exhaustive repository of 10 million scene photographs, labeled with 434 scene semantic categories, comprising about 98% of the type of places a human can encounter in the world (Zhou et al., 2017). The rise of multi-million-item dataset initiatives has enabled data-hungry machine learning algorithms to reach near-human semantic classification performance at tasks such as visual

(a) Building hidden by greenery (b) Interior of a building

Figure 3.8: Examples of invalid GSV static images.

object and scene recognition. The strategy of Places is to provide an exhaustive list of the categories of environments encountered in the world, bounded by spaces where a human body would fit (e.g., closet, shower). The researchers define different subsets of Places database as benchmarks: from the 434 categories (classes), they selected 365 categories with more than 4000 images each to create Places365-Standard. It contains 1,803,460 training images with the number of images per class varying from 3,068 to 5,000. The validation set has 50 images per class and the test set has 900 images per class. Given the impressive performance of the deep Convolutional Neural Networks, particularly on the ImageNet benchmark (Krizhevsky et al., 2012), the Places365 researchers choose four popular CNN architectures, AlexNet (Krizhevsky et al., 2012), GoogLeNet (Szegedy et al., 2014), Residual Network (ResNet) (He et al., 2015a) and VGG16 convolutional-layer CNN (Simonyan and Zisserman, 2014), then train them on Places365-Standard and other benchmarks respectively to create baseline CNN models. The trained CNNs are named as PlacesSubset-CNN. As shown in Table 3.1, it is possible to see that Places365-VGG and Places365-ResNet have similar top performances in the prediction of the label that exactly matches the ground-truth label; for that reason, Places365-VGG16 is selected for implementation in our pipeline process.

	Test Set of Places365		Validation Set of Places365	
Model	**Top-1 acc.**	**Top-5 acc.**	**Top-1 acc.**	**Top-5 acc.**
Place365-AlexNet CNN	53.17%	82.89%	53.31%	82.75%
Place365-GoogLeNet	53.63%	83.88%	53.59%	84.01%
Place365-VGG	**55.24%**	84.91%	**55.19%**	85.01%
Place365-ResNet	54.174%	**85.08%**	54.65%	**85.07%**

Table 3.1: Places365 CNNs performance (Source: Zhou et al., 2017).

Places365-VGG16 architecture description

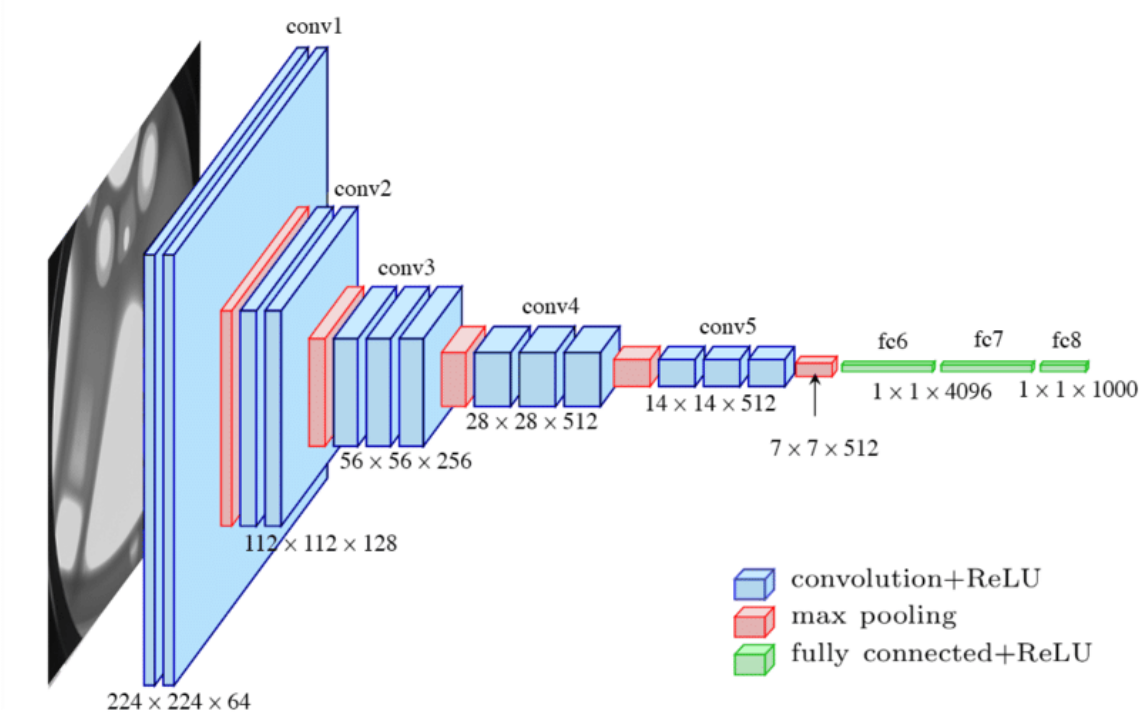


Figure 3.9: VGG16 architecture (Simonyan and Zisserman, 2014).

Considering the original architecture (Figure 3.9), VGG16 takes input as a tensor of size 224x244 with 3 RGB channels. The most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2. The convolution and max pool layers are consistently arranged throughout the whole architecture. At first, the network can be divided into two segments, a fully-connected part and a convolutional one. Considering the original architecture, Conv-1

Layer has 64 filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 have 512 filters. Three fully-connected layers follow the stack of convolutional layers: the first two have 4096 channels each, and the third contains 1000 neurons for classifying as many classes contained in the ImageNet dataset. The final layer is the soft-max layer and the predicted class is the one associated with the component of the highest value. Differently from the shown architecture, Places365-VGG has as last layer a fully-connected layer with 365 channels representing the 365 probabilities associated with each scene computed for the input image. The total number of trainable parameters is 135,755,949. Generally speaking, neural networks are initialized with random weights and after a series of training epochs weights will converge to some values that can properly classify the input images. In this specific case, those weights are already initialized to certain values that are already good to classify a certain dataset (Places365). Thus, a dataset is not needed to train the network since it is already efficient in detecting the target: the recognition of building scenes.

Image selection criteria

The CNN was applied to the GSV raw images and the list of all detected scenes were reported with the relative probability. In the preliminary data selection, two selection/exclusion criteria for detecting whether the image contains a building are introduced:

1. Building Class Detection: all the outdoor building labels ('apartment_building', 'hotel/outdoor', 'building_facade', etc.) of the Places dataset were inserted in a building list. After sorting the detected classes according to the probability given the GSV image, if none from the top 5 most probable predicted classes belong to the building list, the image is filtered out.

2. Probability: The probability of the image was set as the highest probability of the detected class that is in the building class list. If the probability was less than a threshold of 0.07, the image is filtered out.

As can be seen in Figure 3.10, the right side shows the top 5 predicted scene labels for the image shown on the left. In that case, the GSV image is considered invalid since none of the top 5 predicted labels belongs to the building label list. Conversely, Figure 3.11 shows a valid image: more than one of the top 5 predictions is in the building list label exceeding the threshold probability.

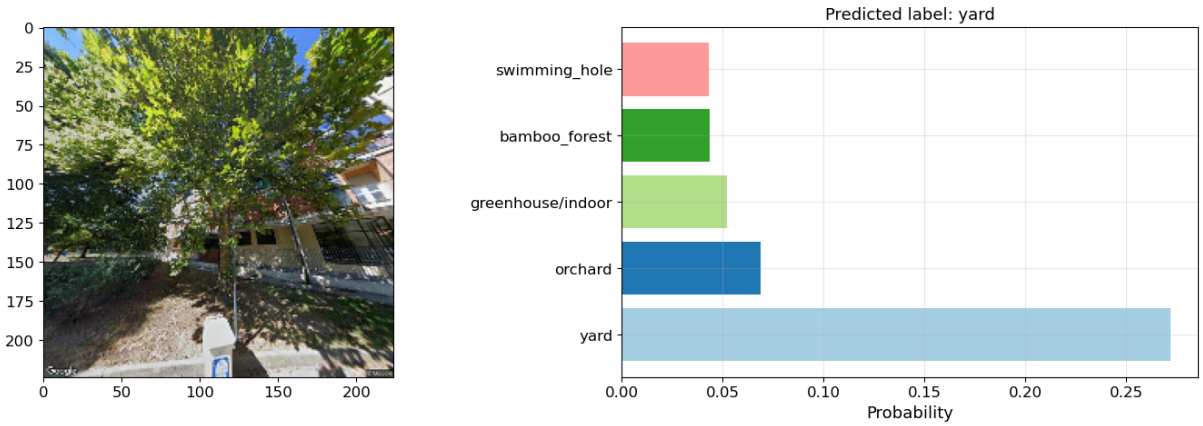


Figure 3.10: Example of Places365-VGG16 predictions on GSV invalid image.

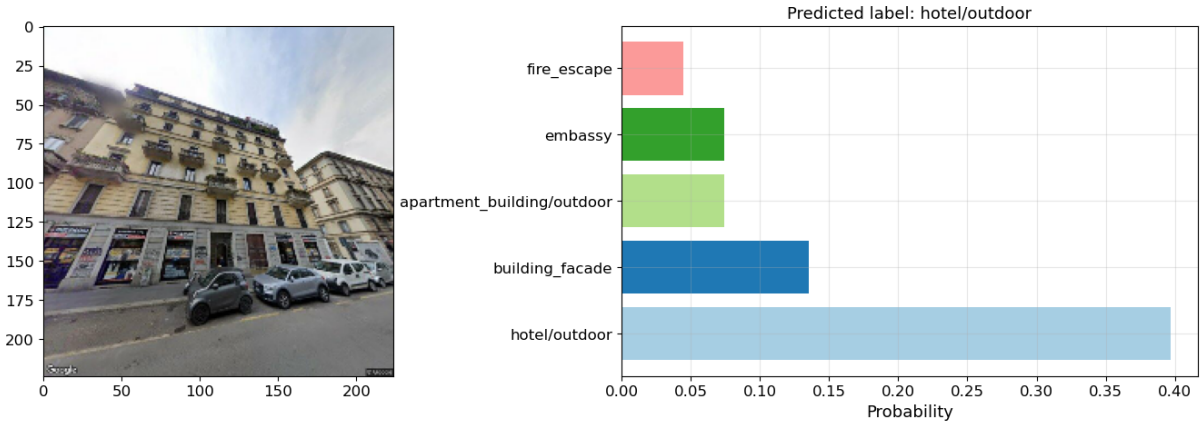


Figure 3.11: Example of Places365-VGG16 predictions on GSV valid image.

3.5. CNN for building height estimation

Once extracted the valid building's GSV images, the height associated with them was considered as a target to train a new Convolutional Neural Network able to estimate such feature. The relative heights were retrieved using a shapefile of the study area as explained later in Chapter 4. In order to reach the best estimation, different kinds of architectures, input preprocess functions, and training procedures were taken into consideration; here all the methods used will be listed and the corresponding results will be shown in Chapter 5 (Results).

3.5.1. GSV images preprocessing

The preprocess methods discussed in this section were applied both on the baseline model and even on the pretrained architectures. The dataset contains about 4245 images. The dataset was split into a training set of 2717 samples, a validation set of 679 samples and a test set of 849 samples. The *training dataset* is composed of pairs of images and heights used to fit the model. The *validation dataset* contains data used to provide an unbiased evaluation of a model fit on the training dataset while helping the development of the model. The evaluation becomes more biased as improvements on the validation dataset are incorporated into the model configuration. It is used to increase the model performance through the tuning of the model *hyperparameters* and *callback function*. Hence the model occasionally sees this data and so the validation set affects a model, but only indirectly. Consequently, evaluation of the actual performance of the model requires an external dataset. The *test dataset* is used to provide an unbiased evaluation of a final model. However, selecting the correct amount of training data for all of the features that need to be trained is a difficult question. If the training set has low variability, the network can *overfit* on the training data. More precisely, overfitting occurs when the model is very good at approximating the function for inputs it has been trained on but does not perform as well with inputs it has never seen before. Realistic images contain a variety of sizes, poses, zoom, lighting, noise, etc. To make the network robust to these commonly encountered factors, also the method of *data augmentation* is used in the project (Perez and Wang, 2017a). As more parameters are added to a CNN, it requires more examples to train the machine learning model. Deeper networks can have higher performance, but the tradeoff should be considered between increased training data needs and increased training time. For this reason, it is also convenient not to have to hunt for or create more images suitable for an experiment. Data augmentation can reduce the cost and effort of increasing the set of available training samples. Image data augmentation is now a famous and common method used with CNNs and involves techniques such as flips, rotation (at 90 degrees and finer angles), translation, scaling, noise addition, etc (Figure 3.12). These augmentations can be combined to make many variants of the original image and randomly applied. By rotating input images to different angles, flipping images along different axes, or translating/cropping the images, the network is exposed to different variations in the data and can learn to recognize the underlying patterns and features that are relevant to the problem at hand, even when they appear in different forms or orientations.

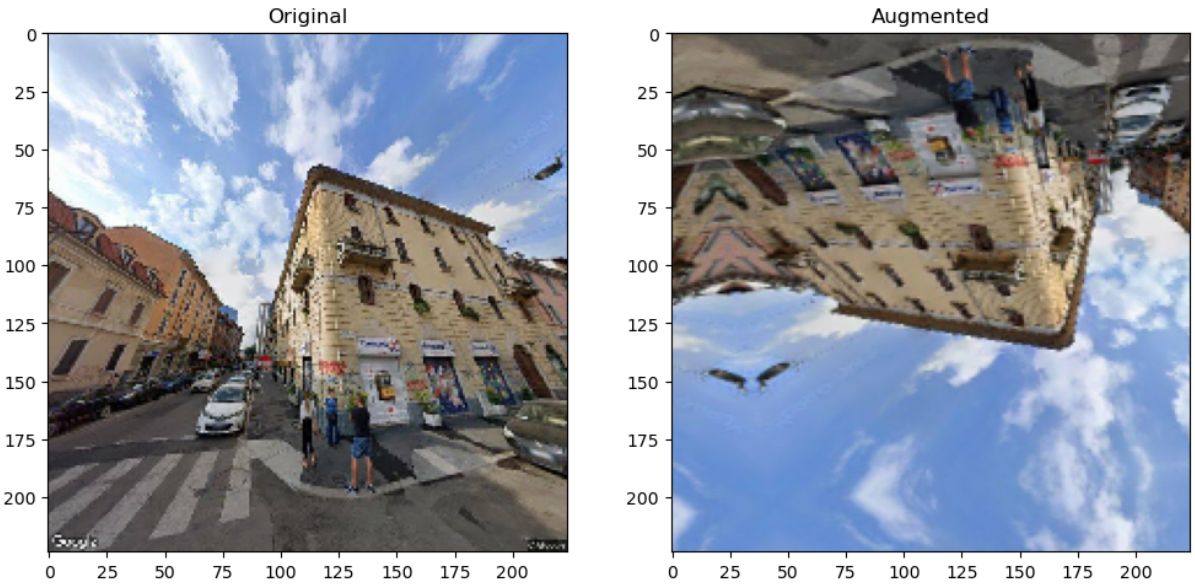


Figure 3.12: Left: training image. Right: Height shift, width shift, zoom, horizontal and vertical flip randomly applied randomly to the training image.

As it will be introduced in the next sections, apart from a baseline model, well-known architecture (VGG16, ResNet50, VGG16-Places365) will be exploited to estimate the building height. Apart from the preprocess methods discussed, those CNN require specific preprocess functions to work properly when applying transfer learning and fine-tuning.

3.5.2. Baseline model

The baseline model was intended to compare its performance against more complex models to understand which techniques could improve the results. As a baseline CNN, a handcrafted Convolutional Neural Network architecture was designed for image regression tasks. The architecture consists of several convolutional layers, each followed by a max pooling layer. The used convolutional layers contain in sequence 16, 32, 64, 128 and 256 filters, 3x3 kernel size, ReLU activation, and the same padding. The max pooling layers following them have 2x2 pool size. The last convolutional layer is followed by a flattening layer that converts the output into a 1-dimensional vector, which is then passed through two fully connected layers with dropout (Srivastava et al., 2014) regularization, and finally to an output layer with a single output unit and linear activation. The architecture is shown in Figure 3.13.

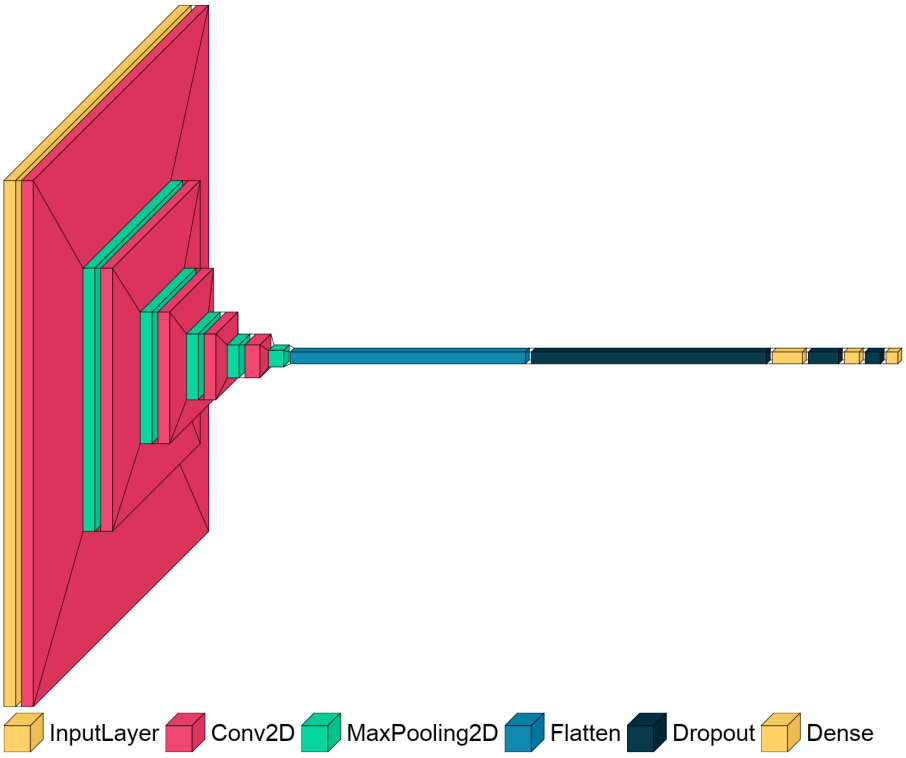


Figure 3.13: Baseline CNN architecture.

3.5.3. Pretrained models: VGG16, ResNet50 and Places365-VGG16

Apart from considering the baseline model, 3 pretrained models were evaluated to improve the performances using two different training techniques: transfer learning and fine tuning.

Transfer learning

The need of a large amount of data is one of the biggest problems to allow the model to understand long term and underlined patterns of data. Considering, as an approximation, that the complexity of the model should have an almost linear dependency on the amount of available data, it is clear how in deep neural networks the lack of samples is an inescapable problem. Moreover, the collection of organized and uniform data is, in most of the cases, very difficult or expensive and, nevertheless, they usually need to be further reviewed. To overcome the problem and enhance the performance *transfer learning* has been used: this technique is leveraged to store the knowledge about a specific task in a model that will be reused for another task, different but similar to the previous one (Bozinovski, 2020). The idea is based on the concept that the data are not independent and identically distributed. That means that the data in the target domain may have some

similarities with the data in the source domain and so the model in the target domain does not necessarily need to be trained from scratch. The main advantage is that in this way it is possible to perform tasks on small datasets by exploiting the inner dependence between two similar dataset. The dataset used for transfer learning is ImageNet (Deng et al., 2009) (large visual database designed for use in visual object recognition software research): the starting layers of the models trained on that dataset could extract relevant features useful for building height estimation. The idea was to use the architecture and weight of these CNNs to simplify the training process and to see if the results would be better than the baseline model. Therefore, some models already pretrained on ImageNet were chosen: VGG16 (Simonyan and Zisserman, 2014) and ResNet50 (He et al., 2015a). The original classification objective of the ImageNet CNNs was to classify images from the ImageNet dataset into one of 1,000 object categories. In addition to ImageNet, the Places dataset (Zhou et al., 2017) and its related model trained on it that has been used for the GSV image classification (Places365-VGG16) is also used for transfer learning: the CNN was used to recognize different scenes, including the characteristics of the buildings framed, and it can certainly be influential in the new task of estimating building heights. The workflow was to take these successfully pretrained CNNs, remove and design the new final fully-connected layers to match the new problem, freeze the weights of the previous layers and train the added layers in the new network using the training data. Since the target is the height, the CNN problem is a regression one; in that case, the old final layers used for the classification were removed for each CNN (ResNet50, VGG16, Places365-VGG16) and the introduced final architecture is designed to have as output layer only one neuron with a *linear activation function* to match the new regression task. The new final layers introduced for all the pretrained CNN were designed differently for each model: the details of the architectures will be explained in Chapter 5.

Fine tuning

While transfer learning refers to the practice of taking an existing pretrained CNN model and applying it to a new, similar task, *fine tuning* refers to the process of retraining certain layers of a pretrained CNN model on a new dataset. The main difference between transfer learning and fine-tuning lies in the extent to which the pretrained model is modified. In transfer learning, the pretrained model is used as a feature extractor, where the features learned by the pretrained model are fed into a new classifier to make predictions on a new dataset. Typically, the pretrained model is kept unchanged during this process, except for possibly the final layer which is replaced with a new layer tailored to the new task.

In contrast, fine tuning involves training not only the final layer but also some of the earlier layers of the pretrained model on the new dataset: the depth of the retraining procedure will be different with respect to each pretrained model. This is done to fine tune the features learned by the pretrained model to better suit the new task. In general, fine-tuning is more computationally expensive and requires a larger dataset than transfer learning. It has been taken into consideration since transfer learning is a good option when the new task is similar to the task the pretrained model was trained on (ImageNet or Places), while fine-tuning can provide better performance in the case that the new task (estimating building height) is more different from the original task. Therefore, the results of VGG16, ResNet50 and Places365-VGG16 will be analyzed both using transfer learning and fine tuning technique.

3.6. Machine learning model for water consumption estimation

Once the height of the building is estimated, the area and socio-demographic information associated with the neighbourhood where the building is located is normally publicly available. The socio-demographic data can be divided into two categories: data on the number of residents divided by age and gender and data regarding the size of families occupying the residential buildings in the urban neighbourhood. As explained, water consumption may be related to many potential factors (determinants) with nonlinear relationships, mostly unknown. The goal is to discover, express, and understand this nonlinearity using a machine learning model, while relying only on relevant input features. The target of the model is to estimate the average daily water consumption of a residential building (known via water meter reading data from utilities) using only public information sources.

3.6.1. Models features definition

Before using all the features, a base model was created using only the area (m^2) and height (m) of the building. The goal is to have a basic model from which to increase performance and understand how much building characteristics were influential in estimating water consumption and how estimation changes when inserting socio-demographic information. As can be seen in Figure 3.14, where DWC stands for daily water consumption, the distribution of observations for the study area (Chapter 4) does not follow a clear specific function with respect to height and area, estimating water consumption using only the building characteristic is limited. There is a slight increase when the area or height has

higher values, but the relationship seems not linear.
Unlike the base model, the final model will have height, area, and socio-demographic information of the neighbourhood where the building is located as features. The target variable will still be daily water consumption.

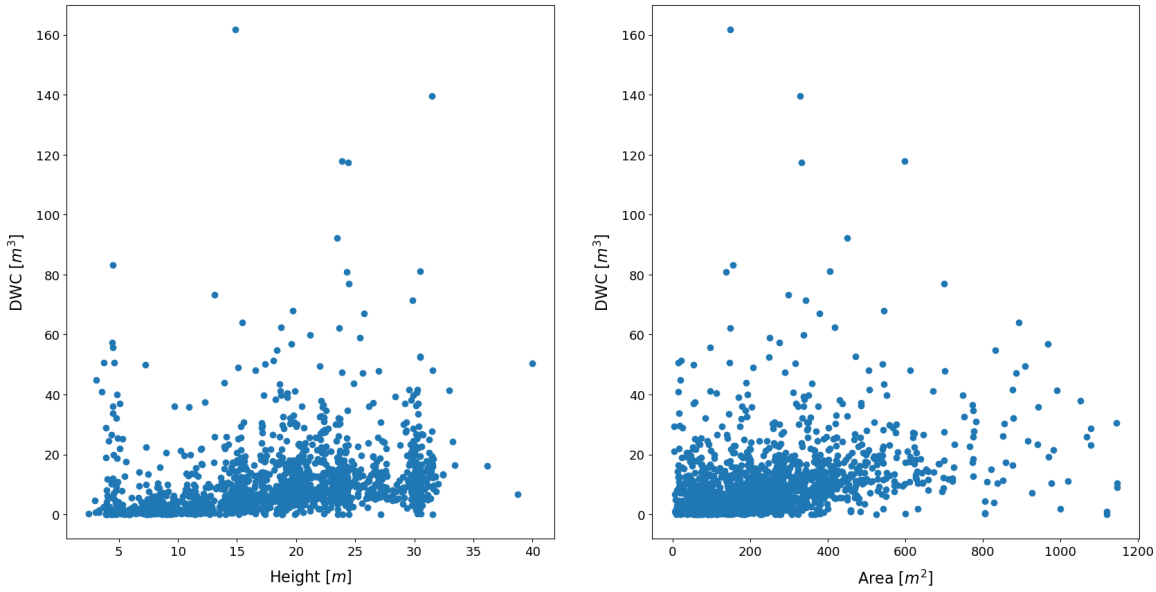


Figure 3.14: Left: daily water consumption distribution vs building height. Right: daily water consumption distribution vs building area.

3.6.2. Data preprocessing

Before defining any ML model, the data was cleaned by removing outliers using 3 different methods to evaluate their effects on the final algorithms. Only the one that leads to the best result will be kept. An outlier is an observation that lies an abnormal distance from other values in a set of random samples from a population.

Outliers detection

Isolation forest (Liu et al., 2008) is a tree-based unsupervised outlier detection algorithm. In the case of multiple features, the algorithm randomly selects a feature and a split value for each node in the tree, and then it splits the data based on this feature and value. The splitting continues recursively until the data points are isolated or the tree reaches the maximum allowed depth. When a new data point is evaluated, it is passed through the tree and the path from the root to the final leaf node is recorded. The path length is calculated as the number of edges traversed in the tree to reach the final leaf node. The path length is expected to be shorter for normal data points and longer for outliers. The

outliers are identified as those data points with a longer path length in the isolation trees compared to most of the data points. The threshold for defining an outlier is based on the average path length of the isolation trees. With multiple features, it builds multiple random trees to isolate each data point and measures the average path length from the root to the final leaf node. Outliers are identified as the data points with a longer average path length.

The second method used is *Local outlier factors* (Breunig et al., 2000). It measures the local deviation of the density of a given sample with respect to its neighbors. It is local because that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood. More precisely, locality is given by k-nearest neighbors, whose distance is used to estimate the local density. By comparing the local density of a sample to the local densities of its neighbors, one can identify samples that have a substantially lower density than their neighbors. These are considered outliers. Figure 3.15 shows the local outlier factor and isolation forest outlier detection methods on the baseline model data. Both techniques have as input parameter the percentage of outliers in the dataset called *contamination ratio*.

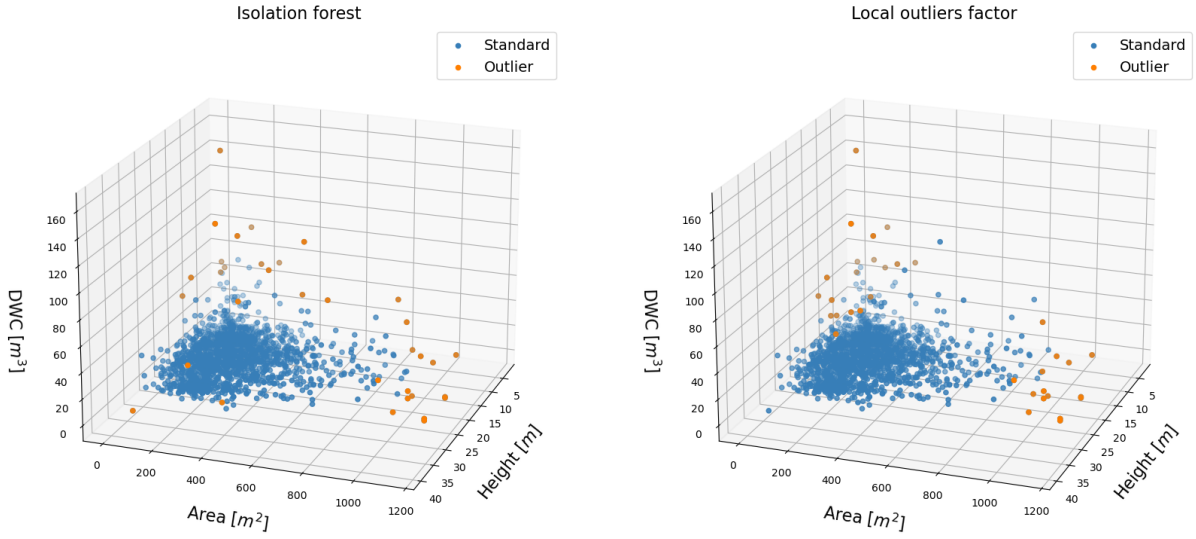


Figure 3.15: Left: outliers detected by isolation forest. Right: outliers detected by local outliers factor.

The last method is the *IQR* method based on quantile. A quantile defines a particular part of a dataset, i.e. a quantile determines how many values in a distribution are above or below a certain limit. The lower quartile, or first quartile (Q1), is the value under which 25% of data points are found when they are arranged in increasing order while the

upper quartile, or third quartile (Q3), is the value under which 75% of data points are distributed. The interquartile range (IQR) is the range between the 1st and 3rd quartile and it is used to identify extreme values that are distant from the majority of the observations. The lower and upper bounds are identified as $Q1 - 1.5 \times IQR$ and $Q3 + 1.5 \times IQR$. Any observation that falls outside of these bounds is considered an outlier (Figure 3.16).

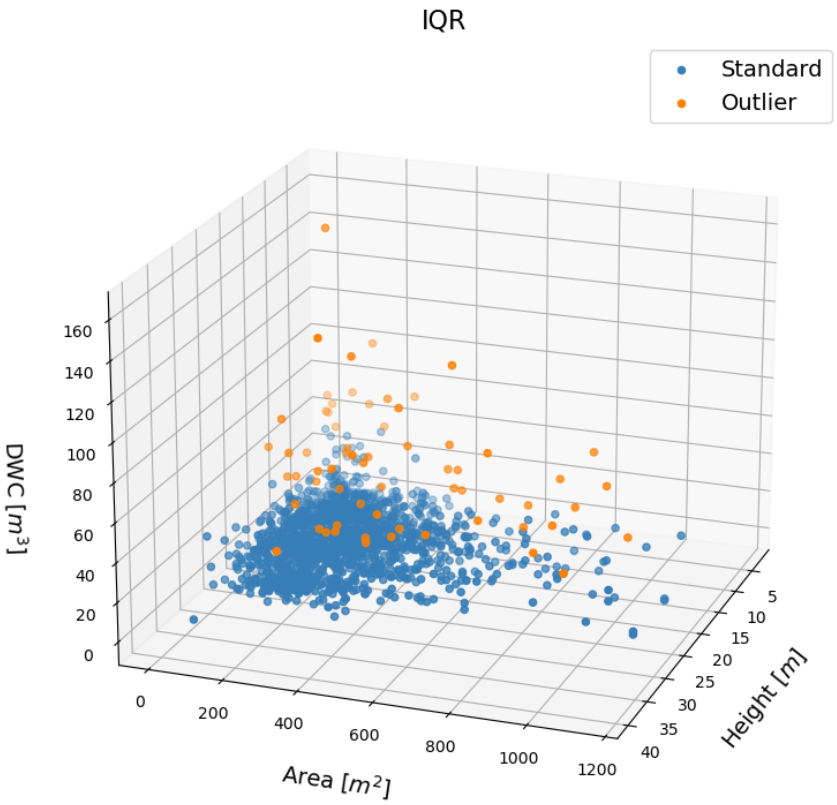


Figure 3.16: Outliers detected by IQR method.

All three methods will be applied separately, and the one that yields the best result will be chosen.

Scaling methods

After deleting the outliers, different scaling techniques were evaluated to reach the best performances and only the best one will be considered. Since input variables have different units (e.g., m, m^2, m^3, etc.), in turn, it may mean the variables have different scales. Differences in the scales across input variables may increase the difficulty of the problem being modelled. An example is that large input values can result in a model that learns

large weight values for a specific feature. A model with large weight values is often unstable, meaning that it may suffer from poor performance during learning and sensitivity to input values resulting in higher generalization error. Also, algorithms that use distance measures between examples or exemplars are affected, such as *k-nearest neighbours*. One of the scaling methods used is *MinMax* scaler that transforms features by scaling each feature to a certain range. So, it scales and translates each feature individually to be in the given range on the training set, e.g. between zero and one. The transformation is given by:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.8}$$

where $min(x)$ and $max(x)$ are, respectively the minimum and maximum value assumed by the variable x.
On the other hand, the *Robust Scaler* removes the median ($median(x)$) and scales the data according to the quantile range. That method is less affected by outliers:

$$x' = \frac{x - median(x)}{x_{75\%} - x_{25\%}} \tag{3.9}$$

$X_{75\%}$ and $X_{25\%}$ are respectively the first and third quartile.
Also, the *Standard scaler* was evaluated since it standardizes features by removing the mean and scaling to unit variance: standardization does not get affected by outliers because there is no predefined range of transformed features. In a mathematical form, this is shown as:

$$x' = \frac{x - \mu}{\sigma} \tag{3.10}$$

where μ is the mean and σ is the standard deviation.

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i) \tag{3.11}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{3.12}$$

The three scaling methods were applied individually to the data, and only the best re-

sulting method is considered.

3.6.3. ML models training

To have a more robust performance analysis of the models on the chosen metric and select the best hyperparameters, nested cross-validation was used. *Nested cross-validation* (Figure 3.17) is a technique commonly used in machine learning to evaluate the performance of a model and to select its hyperparameters (Cawley and Talbot, 2010). It involves performing an outer loop of *k-fold cross-validation* to estimate the generalization performance of the model, and an inner loop of k-fold cross-validation to tune the hyperparameters of the model.

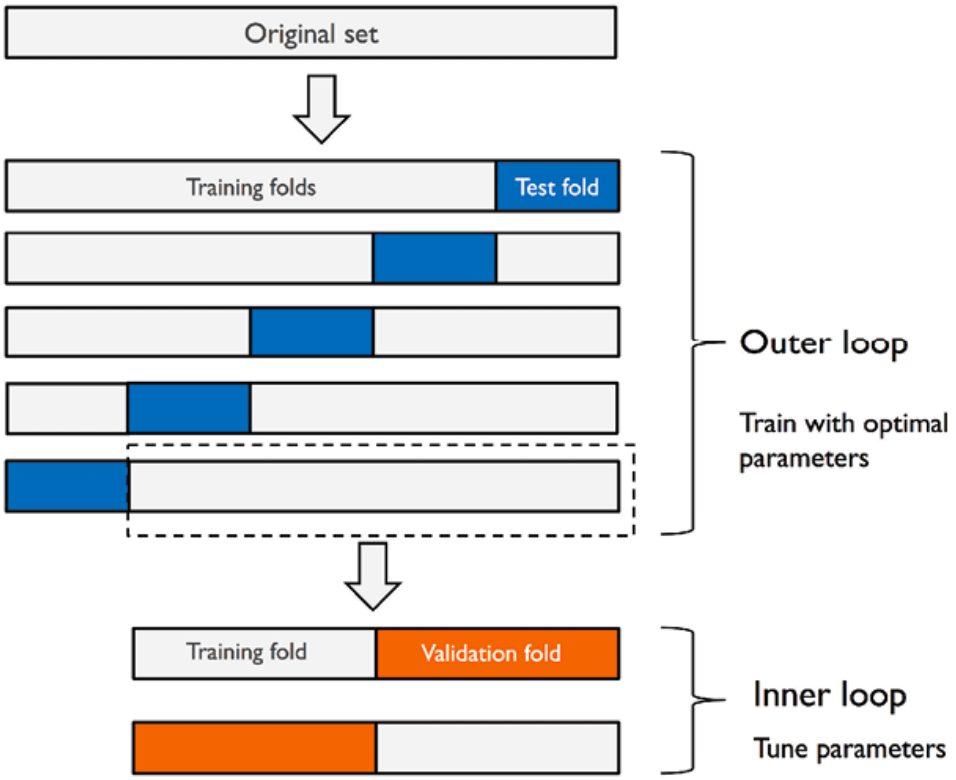


Figure 3.17: Nested cross-validation methodology (Source: DataAnalytics, 2020).

The outer loop of k-fold cross-validation is used to estimate the performance of the model on new, unseen data. In this loop, the data is split into k-folds, with one fold being held out as a test set and the remaining $k-1$ folds being used as the training set. The model is then trained on the training set and evaluated on the test set. This process is repeated k times, with each fold serving as the test set exactly once. The average performance over the k folds is then used as an estimate of the model's performance on new, unseen

data. The inner loop of k-fold cross-validation is used to tune the hyperparameters of the model. In this loop, the training set is further split into k-folds, with one fold being held out as a validation set and the remaining $k-1$ folds being used as the training set. The model is then trained on the training set and evaluated on the validation set. This process is repeated k times, with each fold serving as the validation set exactly once. The hyperparameters that result in the best performance on the validation set are then selected. The outer and inner loops of k-fold cross-validation are nested, meaning that the inner loop is performed for each fold of the outer loop. This allows for a more reliable estimate of the model's performance and hyperparameters, as the performance and hyperparameters are evaluated on different subsets of the data in each fold. Nested cross-validation is a powerful technique for evaluating and tuning machine learning models, as it helps to prevent overfitting and provides a more accurate estimate of the model's performance. The scaling methods seen before are applied to the training data inside the outer k-fold cross-validation loop. This ensures that the preprocessing steps are applied to each fold separately and prevents data leakage from one fold to another. Data leakage occurs when information from the test set is used to preprocess the training data, which can result in overly optimistic performance estimates. By performing the preprocessing steps inside the outer cross-validation loop, the model is trained and tested on independent sets of data, ensuring a more accurate estimate of its performance on new, unseen data. In addition, by applying the preprocessing steps separately to each fold, the model is more likely to generalize well to new data, as it has been trained on a variety of different preprocessed versions of the data. For searching the hyperparameter machine learning model, *GridSearchCV* is used (Liashchynskyi and Liashchynskyi, 2019). It is a function that performs an exhaustive search over a specified hyperparameter grid space for a given machine learning algorithm, used in the inner loop of the nested cross-validation to evaluate the performance of the algorithm on the training set. Hyperparameters are parameters of the model that are not learned from the data, but rather are set before the model is trained. GridSearchCV takes as inputs a machine learning algorithm, a dictionary of hyperparameters to be searched, and a cross-validation strategy. It then creates all possible combinations of hyperparameters from the input dictionary and trains and evaluates the algorithm on each combination using the inner loop of the nested cross-validation. It returns the best-performing model which will be evaluated by the outer loop. Therefore, once performed nested cross-validation on the training set, the final best model is trained on the entire training dataset and evaluated on the test set.

3.6.4. ML regression algorithms

In this research, different machine learning regression methods were comparatively evaluated for the purpose of solving the problem at hand. Specifically, the performances of five algorithms were investigated, including linear regression, polynomial regression, KNN, random forest, and XGBoost. Each of these methods was chosen based on its suitability for the task and its potential to provide accurate predictions. Through this evaluation, the objective is to determine which method would be the most effective for predicting the average daily water consumption using firstly only area and height of the building and, second, also with consideration of socio-demographic features.

Linear Regression

Linear Regression is a statistical method used to model the relationship between a dependent variable y and one or more independent variables $x = (1, x_1, \ldots, x_{M-1})$. The relationship is assumed to be linear, meaning that the change in y is proportional to the change in x. The goal of linear regression is to find the line of best fit that describes the relationship between y and x. The line of best fit is a straight line that minimizes the sum of the squared errors between the observed values of y and the predicted values of y based on x. The formula for a simple linear regression is:

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j x_j \tag{3.13}$$

where y is the dependent variable, x_j are the independent variables, w_0 is the y-intercept, w_k are the slopes of the line. The values of w_0 and w_k can be estimated using different methods such as *ordinary least squares*, which minimizes the sum of the squared errors.

Polynomial Linear Regression

To introduce nonlinearities in the target estimation, *Polynomial Linear Regression* is considered as a possible method. It is a type of linear regression that models the relationship between the dependent variable and one or more independent variables as an nth-degree polynomial $\phi(x) = (1, \phi_1(x), \ldots, \phi_{M-1}(x))$. The key difference between polynomial linear regression and simple linear regression is that the former allows for nonlinear relationships between the variables, while the latter assumes a linear relationship. This allows for more complex relationships between the variables and can lead to a better fit to the data. The formula for polynomial linear regression is:

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x) \tag{3.14}$$

The polynomial function could assume every kind of shape (e.g., $\phi_j(x) = x^j$).

K-Nearest Neighbors

As non-parametric mehod, *K-Nearest Neighbors* (KNN) has been chosen for regression. It makes predictions based on the k closest data points in the training set. The distance metric used to determine the closest neighbours can be any metric, the chosen for the thesis is the Euclidean one. The KNN regressor predicts the target value of a new data point by computing the average of the target values of its k nearest neighbours. The predicted value is then assigned to the new data point. The formula for KNN regression is as follows:

$$y = \frac{1}{K} \sum_{i=1}^{K} y_i \tag{3.15}$$

where y is the predicted target value for the new data point, k is the number of neighbours, and y_i is the target value of the i-th neighbour.

Random Forest

Random Forest (Breiman, 2001) is an *ensemble* learning algorithm that works by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Ensemble learning is a machine learning technique that involves combining multiple models to improve the accuracy and robustness of the predictions. Ensemble algorithms work by creating a set of base models, also known as weak learners, and then combining their predictions in a way that reduces the errors or uncertainties associated with individual predictions. A decision tree is a hierarchical structure that consists of nodes connected in a tree-like manner. Each node performs a test on an attribute, which results in a split into two or more branches leading to additional nodes. This process is repeated for each node until no further splits are possible or the desired depth is reached. The output is contained in the final nodes, known as leaves or leaf nodes. The split is determined using

various criteria, such as the *reduction of variance* method, which is commonly used in the regression. This technique identifies the optimal split value by dividing the training set samples in a way that minimizes the average variance of each set. In addition, a decision tree must select the most appropriate attribute for splitting a specific node, which is determined similarly to split criteria using reduction of variance between the optimal split of each attribute. Each tree of the Random Forest is grown using a different subset of the training data (*bagging*; Breiman, 1996)), and a different set of features is used for splitting at each node of the tree. In regression, the final prediction is the average of the individual tree predictions.

Extreme Gradient Boosting

While random forest's final prediction is made by averaging the predictions of all the trees in the forest, *XGBoost* (Chen and Guestrin, 2016) is a gradient boosting algorithm that builds a sequence of decision trees, where each subsequent tree tries to correct the errors of the previous tree. Gradient boosting (Friedman, 2001) is a powerful and popular machine learning technique used for both regression and classification problems. It belongs to the ensemble learning family of methods, which combines multiple individual models to improve overall prediction accuracy. In gradient boosting, a sequence of weak learners is built sequentially, where each subsequent model aims to correct the errors of the previous model. The key idea behind gradient boosting is to optimize a loss function by iteratively adding weak models to the ensemble. The loss function measures the discrepancy between the predicted and actual values of the target variable. In each iteration, the algorithm fits a weak model to the residual errors of the previous model. The residual errors are the differences between the predicted values and the actual values of the target variable. The aim is to compute a model to predict a target value y_i that minimizes a loss function, for instance, the MSE:

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{n=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.16}$$

It is possible to adjust $\hat{y}_i$ to try to reduce the error using the gradient:

$$\hat{y}_i = \hat{y}_i + \alpha \nabla_{MSE}(y, \hat{y}) \tag{3.17}$$

The gradient for the MSE is proportional to $(y_i - \hat{y}_i)$. Thus, each learner is estimating the gradient of the loss. Larger α means larger steps, smaller α smaller steps and smoothing

effect. XGBoost predicts the target value of a new data point based on an ensemble of decision trees. Each decision tree is trained on a subset of the training data, and the final prediction is the weighted sum of the predictions of each individual tree. The weights of the individual trees are determined through the boosting procedure.

3.6.5. Target discretization: ML classification algorithms

Apart from considering the water consumption estimation as a regression problem, the target was also discretized identifying three levels of building water usage as "low", "medium", or "high" water use. Discretization of a continuous variable refers to the process of dividing a continuous variable into a finite number of categories, in that the three labels are identified depending on how much water is consumed in the building involved. The daily water consumption is divided into three bins based on percentiles, which are points in the data below which a certain percentage of the observations fall. The boundaries of the three intervals are based on the 33rd and 67th percentiles, where the first bin contains values below the 33rd percentile (low), the second bin contains values between the 33rd and 67th percentiles (medium), and the third bin contains values above the 67th percentile (high).
Since the problem shifts from a regression to a classification one, the algorithms used are slightly different while the preprocessing methods (scaling and outlier detection) and the training procedure (nested cross-validation) are still applied in the same way. The ML techniques already explained previously and kept for testing in classification mode are only the *KNN*, *Decision Tree* and *Random Forest*. The bagging classifier was combined with the KNN and Decision Tree as base models to increase the performances: a bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregates their individual predictions (either by voting or by averaging) to form a final prediction. Other classifier methods are introduced for a more comprehensive comparative analysis and are described in the following paragraphs.

Logistic regression

Logistic regression is a binary classification algorithm that models the probability of a binary outcome variable as a function of one or more predictor variables. However, it can also be extended to handle multiclass classification problems using one of several techniques, such as the One-vs-Rest (OvR) or the Multinomial logistic regression. In OvR, a separate binary logistic regression model is trained for each class, with the samples of that class as the positive class and all other samples as the negative class. The predicted probabilities from each model are then combined to make the final prediction. In Multinomial

logistic regression, a single model is trained to simultaneously predict the probabilities of all classes using the softmax function, which converts the output of the model into a set of probabilities that sum to one. The formula for the Multinomial logistic regression model is as follows:

$$P(Y = j \mid \boldsymbol{x}) = \frac{\exp(\beta_j^T \boldsymbol{x})}{\sum_{k=1}^{K} \exp(\beta_k^T \boldsymbol{x})}$$

where Y is the multiclass outcome variable with K classes, $\boldsymbol{x}$ is the vector of predictor variables, β_j is the vector of coefficients for class j, and $\exp(\beta_j^T \boldsymbol{x})$ is the likelihood of observing class j given the predictor variables. The softmax function is used to ensure that the predicted probabilities sum to one, and the maximum likelihood estimation method is used to estimate the coefficients of the model. The final prediction for a new observation is the class with the highest predicted probability.

Gaussian Naive Bayes

Gaussian Naive Bayes is a type of Naive Bayes (Zhang, 2004) algorithm that is used for multiclass classification problems where the features are continuous and follow a Gaussian (normal) distribution. The formula for Gaussian Naive Bayes is as follows: For a given class C_k, the prior probability is calculated as:

$$P(C_k) = \frac{n_k}{n}$$

where n_k is the number of samples belonging to class C_k, and n is the total number of samples. The likelihood of observing a set of features $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ given the class C_k is modeled as a Gaussian distribution:

$$P(\boldsymbol{x} \mid C_k) = \prod_{i=1}^{n} P(x_i \mid C_k) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} \exp\left(-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2}\right)$$

where $\mu_{k,i}$ is the mean of feature i for class C_k, $\sigma_{k,i}$ is the standard deviation of feature i for class C_k. The posterior probability of class C_k given the features $\boldsymbol{x}$ can be calculated using Bayes' theorem:

$$P(C_k \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid C_k)P(C_k)}{\sum_{j=1}^{K} P(\boldsymbol{x} \mid C_j)P(C_j)}$$

where K is the total number of classes. The predicted class for a given set of features $\boldsymbol{x}$ is the class with the highest posterior probability.

Extremely randomized tree

Extremely Randomized Trees (Extra Trees) (Geurts et al., 2006) is a type of ensemble learning algorithm that can be used for multiclass classification problems. It is similar to Random Forests, but with some key differences in how it constructs decision trees. To construct an Extra Trees model, a set of decision trees using a random subset of the features and a random subset of the samples from the training set is defined. For each decision tree, the data are split at each node using a random threshold for each feature, rather than finding the best split based on information gain or *Gini* impurity. The predicted class for a given observation is then determined by aggregating the predictions of all decision trees. For multiclass classification, the most common approach is to use a one-vs-all strategy. The predicted class for a given observation is then the class that has the highest probability score among all binary classifiers. The probability score for each class is calculated as follows:

$$P(y = c \mid \boldsymbol{x}) = \frac{\sum_{i=1}^{B} I(y_i = c) w_i}{\sum_{i=1}^{B} w_i}$$

where y is the class label, $\boldsymbol{x}$ is the feature vector of the observation, B is the number of decision trees in the model, y_i is the predicted class label of the i-th decision tree, w_i is the weight of the i-th decision tree, and $I(y_i = c)$ is an indicator function that returns 1 if $y_i = c$ and 0 otherwise. The weight of each decision tree is determined by its accuracy on a validation set, with better-performing trees being assigned higher weights. This allows the model to give more weight to decision trees that are more accurate on the validation set and to discard decision trees that perform poorly.

AdaBoost

AdaBoost (Schapire, 1999) is an ensemble learning algorithm that can be used for multiclass classification by combining the predictions of multiple weak classifiers. The type of weak classifier used in AdaBoost is decision trees. In the case of multiclass classification with decision trees, the AdaBoost algorithm trains a sequence of decision trees, where each tree focuses on distinguishing one class from the others. During each iteration, the algorithm assigns weights to each sample, with the weights of the misclassified samples increasing and the weights of the correctly classified samples decreasing. The algorithm

then trains the next decision tree on the reweighted samples and combines the predictions of all the decision trees using a weighted voting scheme.

4 | Data and experimental settings

This chapter provides a description of the study area and related data, followed by preliminary data analysis to gain insights into the variables analysed, and experimental settings. In this research, data from mainly 3 sources are used: (i) time series water consumption dataset for more than 1500 buildings in the city of Milan (Italy), (ii) a shapefile containing information on building location and features, and (iii) open datasets of socio-demographic features at district level. The water consumption time series are used as targets for the ML models, while the shapefile is used to extract the buildings height used as target for the CNN and the area that will be considered as a direct input with the socio-demographic features to the final model. The time series water consumption is data shared confidentially by the water utility while the shapefile and socio-demographic features are open and constitute all needed for the model input.
At the end of the chapter, algorithm training, hyperparameter tuning, and performance metrics of the different models will be described.

4.1. Study Area

Milan is the chosen study area for this thesis, with a focus on water consumption. As the second most populous city in Italy, Milan presents a unique set of challenges and opportunities in understanding water usage patterns and developing effective strategies for sustainable water management. The city's population of over 1.3 million people, combined with its role as a hub for industry, commerce, and tourism, makes it a significant consumer of water resources. According to Milan Government Plan of the Territory (*PGT*: Piano di Governo del Territorio), the urban area in Milan is divided into 88 NILs (Figure 4.1). *NIL* stands for "Nuclei di Identificazione Locale" which translates to "Local Identification Nuclei". NILs in Milan are administrative subdivisions that are used for a variety of purposes such as census data collection, urban planning, and public services. Each NIL is identified by a unique code that consists of a letter and four digits, and covers a specific geographic area within the city. The use of NILs allows for more targeted analysis and management of urban resources and services based on local needs and characteristics.

Figure 4.1: NILs distribution of the Milan city area (Source: Riva and Lucchini, 2014).

4.2. Water consumption time series

The water consumption data are provided by MM Spa. MM Spa is a company created by the City of Milan in 1955 to design and build the first underground lines. Since that time, it has participated in the realisation and management of major infrastructure in the city. The water consumption time series used in this study starts on 01-01-2019 and end on 08-03-2020 and are collected from a specific *PDR* (Punto di Riconsegna). The PDR is a numerical code, consisting of a minimum of 5 up to a maximum of 21 digits, which uniquely identifies the location of individual water use. The displayed value represents the water consumption of a building associated to the PDR in a single day. For each building and corresponding PDR, there are an associated address and civic number, a flag indicating whether the structure is residential or not, the NIL code and NIL name

where the structure is located and finally its latitude and longitude. The reference system used for latitude and longitude definition is *WGS84*. Figure 4.2 illustrates two example time series associated with two different PDRs. *WC* on the y-axis stands for water consumption and it is measured in m^3. Data preprocessing is needed to fill in missing values during certain periods of the year as can be noticed in Figure 4.2.

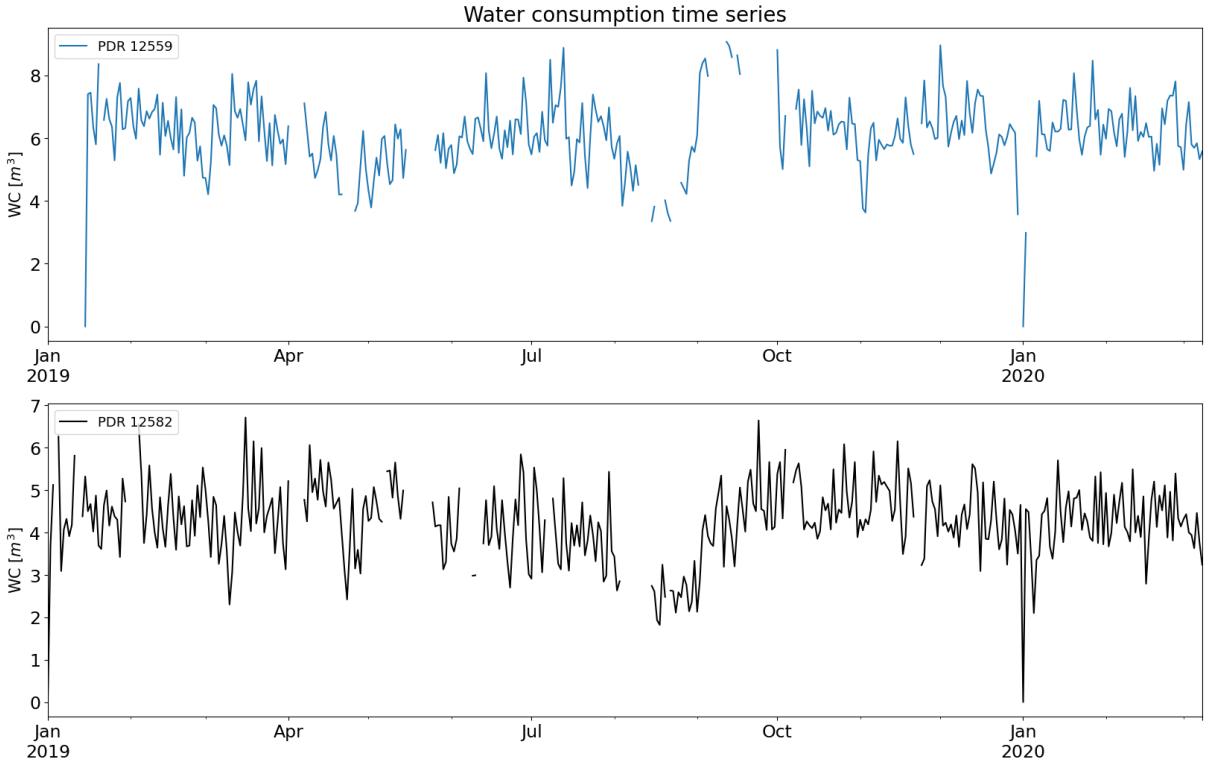


Figure 4.2: Examples of two water consumption time series associated to two different PDRs. Values are sampled every day from 01-01-2019 through 08-03-2020.

4.2.1. Data preprocessing

Before calculating the average daily water usage, a preprocessing phase was necessary to manage missing and noisy values. Since many time series had missing values during the month of January, all series will start on 30-01-2019. To fill in the best way possible the missing value, it is important to underline that water consumption time series usually reveal a type of annual and weekly periodicity, which makes the time series not stationary (Stańczyk et al., 2022). Specifically, residential water usage exhibits a clear weekly seasonality, with lower consumption on weekdays and higher consumption on weekends (Ghiassi et al., 2005). This pattern is likely due to differences in daily routines and activities between weekdays and weekends. For example, people tend to be at home more on weekends, and may engage in activities that require more water, such as doing laundry or

watering plants. Studies have shown that the magnitude of the weekly seasonality varies depending on factors such as climate, socio-demographic characteristics, and the type of water use (Gato et al., 2007, Ghiassi et al., 2005). Understanding the weekly variation of residential water usage can be useful for water utilities and policy makers in developing strategies to manage water demand and ensure sustainable water use. For this reason, the calculation of missing values is performed considering the day of the week for which the water consumption value is missing: for example, if the water usage value for Thursday is missing for a certain PDR, it is imputed by considering the average water consumption over the previous 5 weeks on the same day of the week. In case there are not enough values available in the previous weeks, the missing values calculation will involve the following weeks.

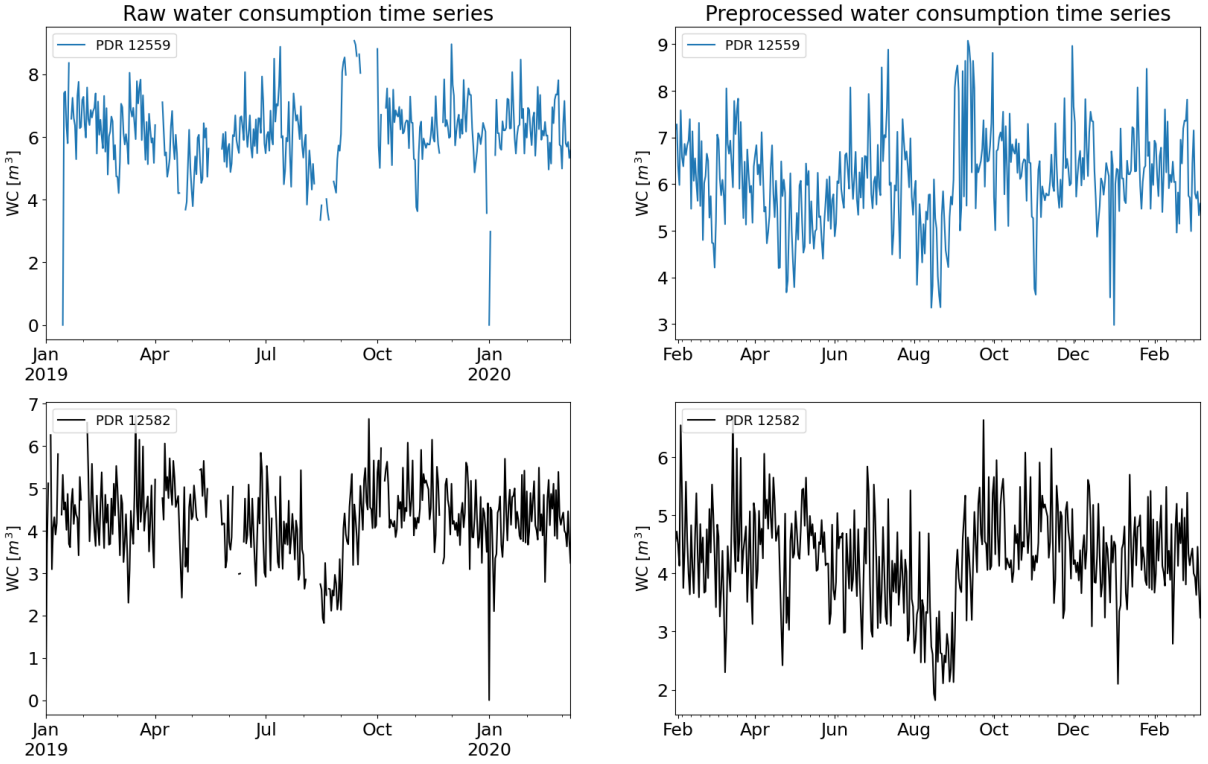


Figure 4.3: Example of two water consumption (WC) time series before and after missing value filling preprocess.

Furthermore, all time series are reported with a value of 0 on the first day of the year 2020; it is an outlier compared to the normal data distribution and for this reason, it has been considered as a missing value and recalculated as previously explained. Figure 4.3 represents an example of the final time series that will be used in the methodology: the average daily water consumption was extracted from them.

4.2.2. Preliminary data exploration

To visualize and gain preliminary insights from data, *Time series decomposition* was the first step applied in an exploratory data analysis. It consists in considering a series as a combination of trend, seasonality, and noise components. Time series data can exhibit a variety of patterns, and it is often helpful to split a time series into several components, each representing an underlying pattern category. There are three types of time series patterns: *trend*, *seasonality* and *residuals*. Trend, as its name suggests, is the overall direction of the data. Seasonality is a periodic component. And the residual is what's left over when the trend and seasonality have been removed. Residuals are random fluctuations and can be considered as a noise component.

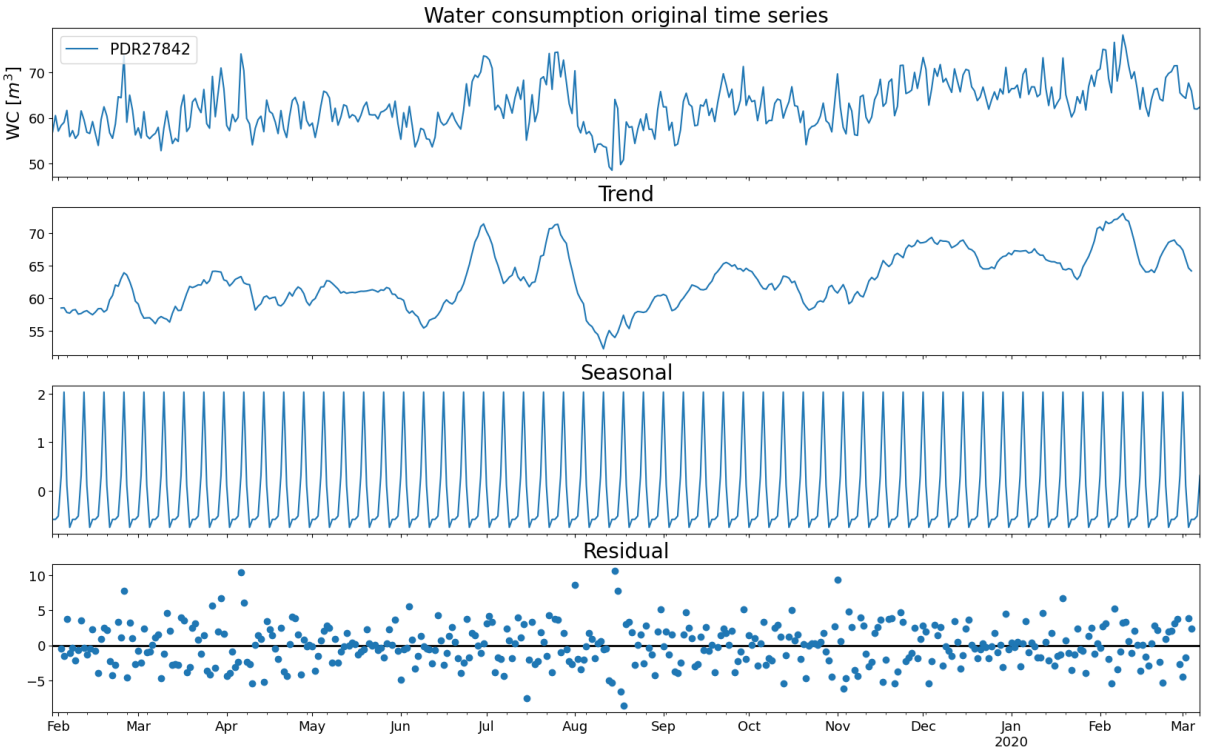


Figure 4.4: Season, trend and residual of water consumption (WC) time series.

There are two major ways to divide the components: the first way simply decomposes time series as the sum of the three components while the second way decomposes time series data as a multiplication of all three components. During the analysis the first technique was used, an example is shown in Figure 4.4. Considering the trend of different time series, the most common feature is the minimum water consumption in mid-August. Families tend to go on vacation during this period and leave their homes. In addition, there was an increase in consumption in July, justified by the increase in temperatures.

The periodic pattern found lasts for 7 days, confirming what was said previously. Decomposition provides useful and interesting information about water consumption changes and periodicity over the year and weekdays.

4.3. Building characteristics

A *shapefile* is a popular geospatial vector data format for storing and sharing geographical data. It is a digital vector format for storing geometric location and associated attribute information. A shapefile usually contains information about geographical features such as points, lines, and polygons, which are represented by vector data, along with their associated attributes. It is a commonly used format for sharing geographical data with others, as it can be easily read and edited by many geographic information systems (GIS) software. A city shapefile contains geographic information on the boundaries and features of a particular city, typically represented as a collection of interconnected *points*, *lines* and *polygons*. The data in a city shapefile can be used to create detailed maps, perform spatial analyses, and support a wide range of urban planning and environmental applications. In the official Milan geoportal website (GeoportaleMilano, 2012) it is possible to find, as public data, shapefiles related to streets, buildings, waterways, greenery, and other features that make up the urban landscape of a city, providing valuable information for researchers studying various aspects of city life and infrastructure. Geographic data that describes the natural and human-made aspects of a location is structured into three main categories: into *layers*, *themes*, and *classes*. The reference structure is made up of the class, which defines the representation of a specific type of territorial objects: the properties, the structure of the data, the rules of acquisition, structuring and relationship with the other items. Layers and themes do not represent a classification but are rather intended to collect classes into subsets that are morphologically or functionally homogeneous, whose homogeneity in the data structure is exploited to simplify the description or specification of the classes that belong to it. More information could be retrieved in the official document specifications (RegioneLombardia, 2022). Since the objective is to extract information about the Milan residential building, the layer involved is the "Immobili e antropizzazioni" ("Real estate and human-made alterations"), the theme is named "Edificato" ("Built") and the class is "Unità Volumetrica" ("Volumetric Unit"). The volumetric unit is the elementary volume referred to a building. The term indicates a constructed body whose top is made up of either a real flat surface, such as a flat roof, or an ideal flat surface that defines what can be considered the volume of the built body to calculate its volume, albeit approximated. The height of this flat surface, whether real or ideal, is called the "eaves level" of the volumetric unit. Once defined the layer, class

and theme, the shapefile is extracted. The spatial reference system of the shapefile is *ROMA40*. A visual representation is shown in Figure 4.5.

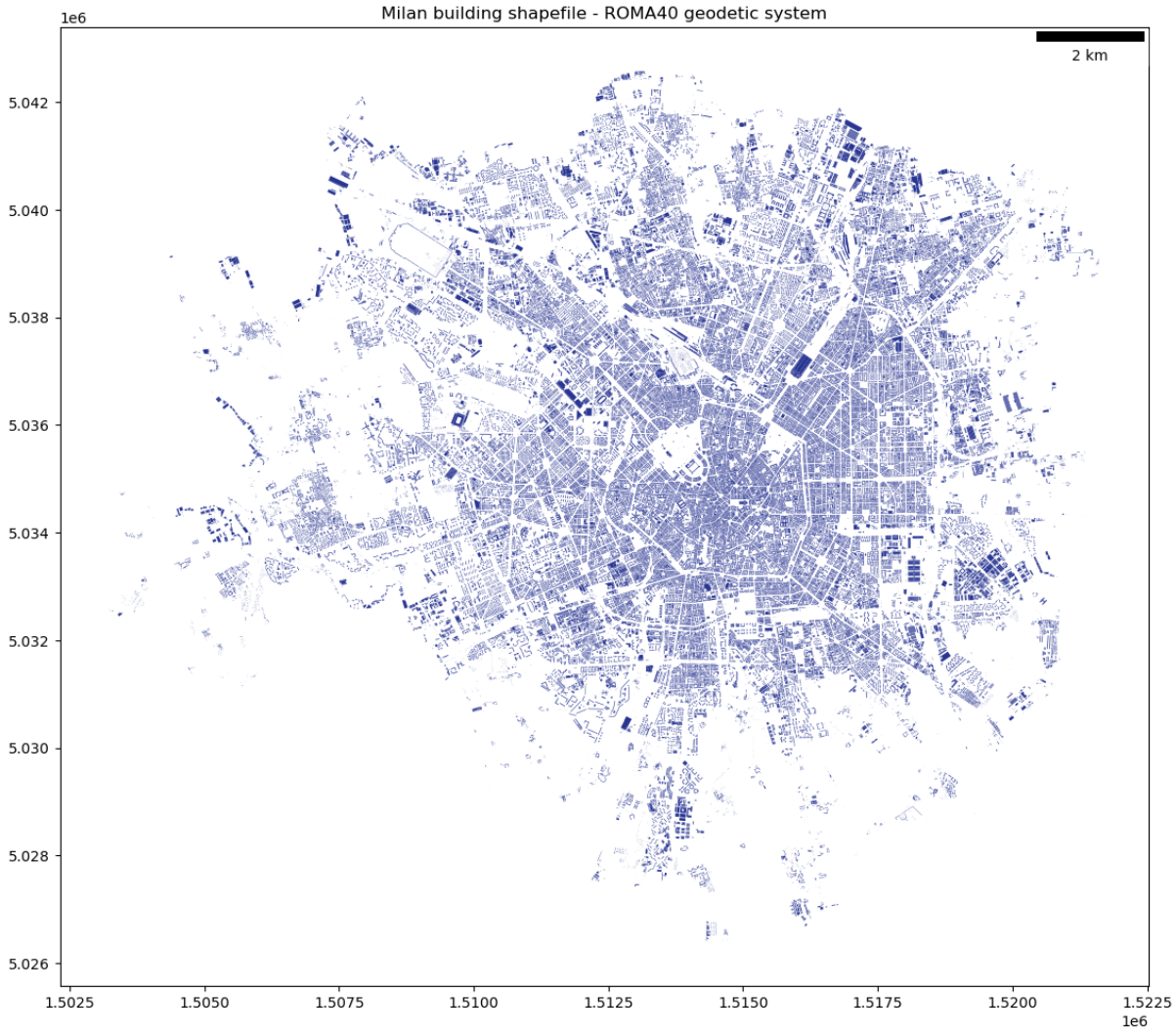


Figure 4.5: Milan building shapefile.

The two features of interest retrieved from the Milan building shapefile are the building polygon and its related height. The polygon has a two-dimensional shape defined by a set of vertices (points with specific longitude and latitude) that are connected by straight line segments to form a closed shape. Polygons are used to represent the volumetric units, thus the building footprint in the study area. The height will be used as a target for the CNN whose objective is to estimate the building height.

4.3.1. Mapping addresses coordinates into building shapefile

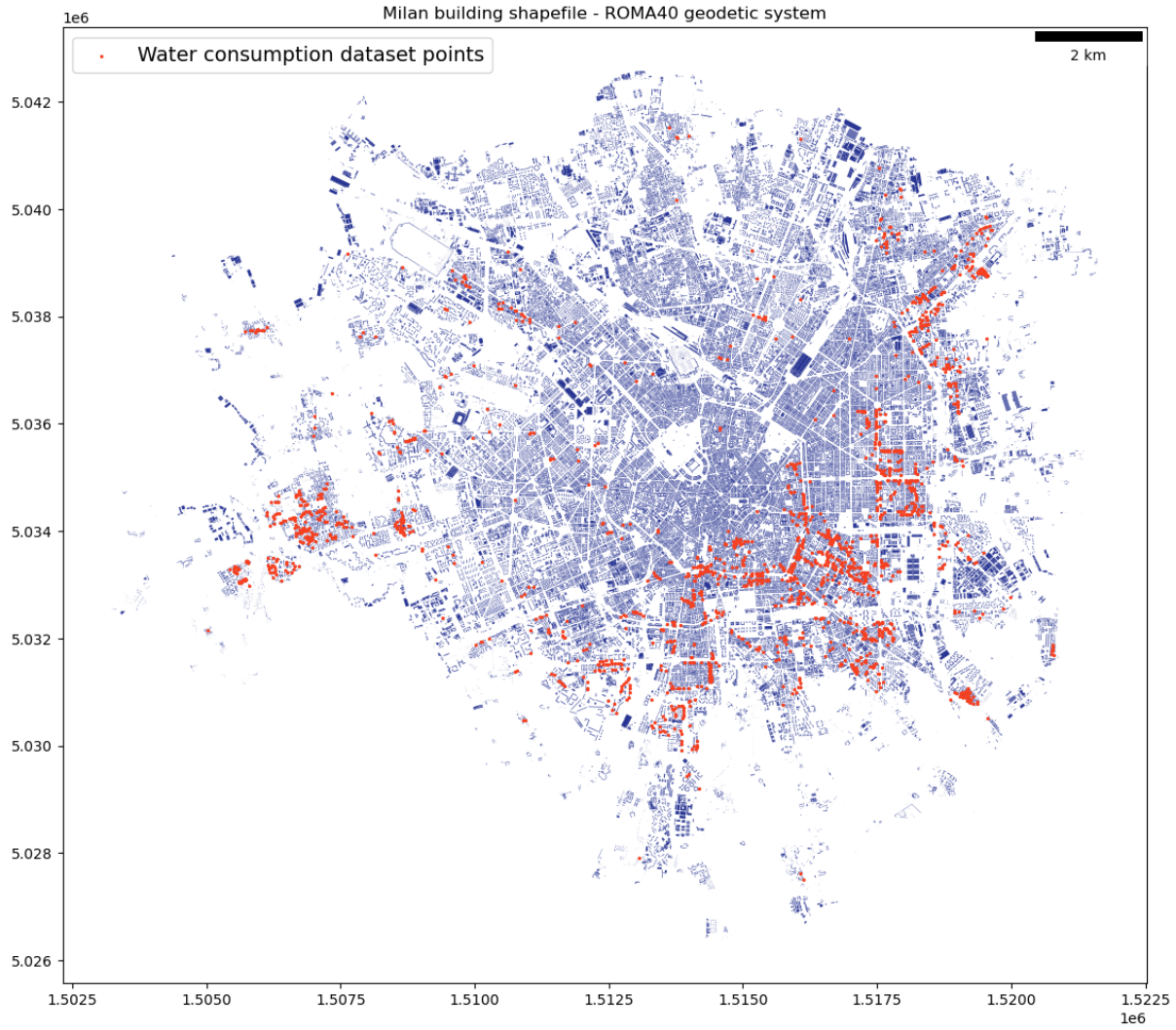


Figure 4.6: Red points represent the addresses associated with the water time series available in the dataset. The blue polygons describe the Milan building footprints.

Shapely is a Python package for working with vector geometries: it includes functions for creating geometries, as well as functions for applying geometric operations on geometries, such as calculating the centroid of a polygon. Once each polygon is expressed as a Shapely object the area (m^2) is calculated. The aim is to map the building addresses in the water consumption dataset into the shapefile in order to associate the area of the building that will be used as a feature in the ML model with its corresponding daily water usage. Therefore, the addresses associated with each PDR account were converted from the *WGS48* system to *ROMA40*. The point distribution of building addresses is shown in Figure 4.6. In addition to the addresses in the water consumption dataset, other random

addresses were considered for the CNN training that estimates the building heights. In the official Milan geoportal, there is a dataset indicating all the addresses of residential buildings with their respective latitude and longitude position. By extracting their GSV image and projecting the point into the shapefile to extract the height, a dataset containing the image and target was created, which will be used by the Convolutional Neural Network.
Despite the high quantity of information used, some issues and limitations were encountered during the data acquisition process. Unfortunately, the geo-location (latitude and longitude position) of addresses in the dataset sometimes is not precise and therefore there are some points that do not belong to any volume in the shapefile (see, for example, Figure 4.7). They were simply associated with the nearest building in the shapefile.

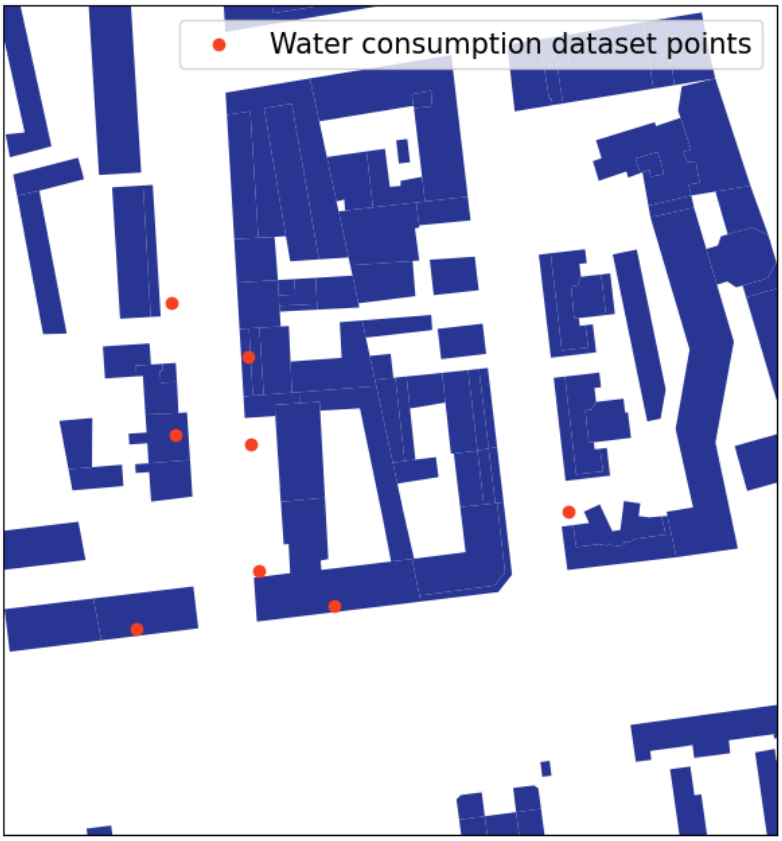
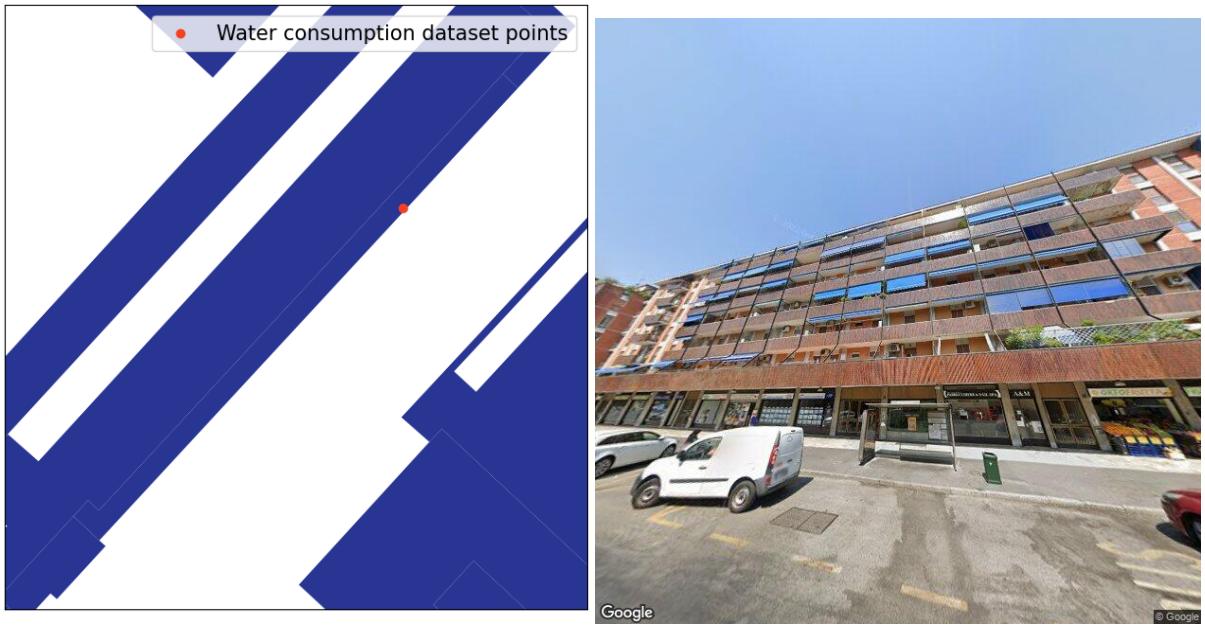


Figure 4.7: Example of collected addresses that are not associated to any building in the shapefile.

There are also special cases where the point is equally distant from two buildings. In this case, that specific address is excluded from the analysis. In addition, as already explained in the shapefile features, a single building is composed of multiple volumetric units. These are used to define the different architectural sections of the building: this

means that for a single building, there are multiple heights associated with the different volumes that compose it. The methodology consisted in analyzing the volume heights of individual buildings in addition to the volume containing the point: the average, variance and highest value of the volumes height for a single building are considered. The goal is to use these characteristics to exclude buildings that may have incorrect heights. Buildings with a maximum height of less than 3 meters and having a too large variance due to the large number of volumes with different heights are excluded.
The same type of problem in a more severe way occurs for the area. If an address point falls in a building volume that is only a small portion of its larger area, it will be associated to an incorrect surface value and so turns out to be an outlier.



(a) Mapped water consumption address point in Milan building shapefile

(b) GSV image

Figure 4.8: Occurring errors in shapefile.

Additionally, a single building in the shapefile can be composed of multiple civic address numbers. The time series on water use are defined at the level of a single civic number, so there is the risk to associate an area that involves multiple numbers to the water consumption of a single one. A summary of the limitations encountered using the shapefile as a source of information is presented in Figure 4.8. Figure 4.8a shows the shapefile of the building captured by the GSV image 4.8b. Let's consider the longitude and latitude position of the address that is plotted in the shapefile as a point. As can be seen, the point belongs to a volume having a terrace on the first floor: extracting the height and area using that information will be wrong. In addition that single volume would involve

multiple civic addresses while the analysis is interested in calculating the area of a single one. The issues described are certainly influential in the final machine learning model for calculating daily water consumption. A possible future development and improvement could be to also consider other types of public sources for more accurate data that could lead to a better final result.

4.4. Socio-demographic information

The geoportal of the Milan municipality contains various information on the socio-demographic characteristics of its inhabitants. The goal is to include useful features for the ML model. Due to privacy concerns, the surveys are at the NIL level. The data include information on the resident population and the type of family. For each NIL, both the number of residents and the size of the neighborhood are indicated. Using this information, it is possible to calculate the population density in that area of the city. Population density has been found to be an important factor in estimating water consumption in urban areas. High population density is associated with increased water demand due to higher household occupancy rates, smaller dwelling sizes, and greater use of communal water facilities (e.g., apartment buildings). In contrast, low population density is associated with increased per capita water use due to larger lot sizes, greater outdoor water use, and increased use of private swimming pools (Shuster et al., 2005). Overall, population density is an important factor to consider when estimating water consumption in urban areas. However, the exact relationship between population density and water use may vary depending on local factors such as climate, culture, and infrastructure. Therefore, it is important to carefully evaluate the specific context in which population density is being used as a predictor of water consumption. Apart from the population density, the population composition, i.e., the numbers of males, females, minors and people over 65 years old are public for each NIL and used in the final models developed in this thesis. In addition, data on the number of families and their members number is involved in the ML model. The general family composition can be an important factor for estimating residential water consumption. Based on the number of people composing them, the families are divided into three categories: (i) single families composed of only 1 person, (ii) multi families composed of 2-4 people, and (iii) large families with more than 4 people. For each NIL, the number of these three kinds of family was used as socio-demographic features for the model.

4.5. Algorithms training

In this section, it will be presented the training setup of the different models used in the methodology (Chapter 3): the CNN for building height estimation and the final ML model for the water consumption prediction (regression and classification).

4.5.1. CNN for height estimation: training setup

To obtain a more accurate estimate of the height in meters of buildings giving the relative GSV image, several CNN models were considered. Despite the different architectures and related layers, the training procedure and some parameters used were the same to make the comparison as reliable as possible while others are unique depending on the specific architecture. The two types of CNN involved are: the baseline CNN and the pretrained CNN that include VGG16 and ResNet50 pretrained on ImageNet and Places365-VGG16 pretrained on Places365. The main common element in the training procedure of these models are listed here:

- Initially, the dataset composed of 4245 GSV images was split into training set (80%) and (20%) test set. In turn, the training set was split using its 20% of samples for the validation set, resulting in a final number of 2717 data instances used for training and 679 for validation. 849 instances make up the test set. The batch size used during the training process was 32.
- The size chosen for the input image to all the CNNs is 224×224. For some of the networks, the images were rescaled, which is a common preprocessing step in deep learning. This is because image data is usually represented as pixel values ranging from 0 to 255, where 0 represents black and 255 represents white. Rescaling images to a specific range by dividing each pixel value by a certain quantity helps to normalize the data and ensure that all features have a similar scale. This is important because it can improve the training process and prevent the model from being overly sensitive to a specific range of pixel values in the input images. Additionally, scaling the input data can also help to prevent issues with *vanishing* or *exploding gradients*, which can occur during backpropagation and make the training process more difficult (Bengio et al., 1994).
- The optimization algorithm used for all the CNNs is called *Adam* (Kingma and Ba, 2014): a popular optimization algorithm used in deep learning, particularly in neural networks. It stands for Adaptive Moment Estimation and is a combination of two other optimization methods, *Adagrad* (Duchi et al., 2011) and *RMSprop*.

Adam optimizer updates the model's weights iteratively based on the gradients of the loss function with respect to the weights. It keeps track of the first and second moments of the gradient and adjusts the learning rate accordingly for each weight. This allows the learning rate to adapt to the gradient and to update the model parameters in a way that is proportional to their significance.

- During the training process, the callback functions were involved: those functions are called at specific points during the training process to perform a specific action. In deep learning, training a neural network can take a lot of time and resources, especially when working with large datasets or complex models. Callback functions provide a way to monitor the progress of the training and take actions accordingly. To prevent overfitting, all the models were trained using the *Early stopping* callback function: it stops the training process before the model starts to overfit. It takes several arguments, including the monitored metric, the number of epochs to wait before stopping, and whether to restore the weights of the best epoch. During training, the callback function monitors the MAE on the validation dataset. If the metric does not improve for a specific number of epochs, the training process is stopped early. Apart from early stopping also *ReduceLROnPlateau* was introduced to dynamically adjust the learning rate of the optimizer. The purpose of this function is to improve the accuracy of the model and speed up the training process by reducing the learning rate of the optimizer when the loss on the validation set has stopped improving.
- During the training of a CNN, the learning rate is a hyperparameter that determines the size of the steps taken by the optimizer when updating the model's weights. If the learning rate is too high, the optimizer may overshoot the optimal values, leading to slow convergence or even divergence. Conversely, if the learning rate is too low, the optimizer may take too long to converge or get stuck in a suboptimal solution. The learning rate chosen for all the CNNs is equal to $1e^{-3}$. It will be modified only for one specific architecture.
- Regarding the weight initialization, two common initializers are used: *He* initialization and *Glorot* (or *Xavier*) initialization. He initialization (He et al., 2015b) is specifically designed for the rectified linear unit ($ReLU$) activation functions; it sets the initial weights of each layer using a Gaussian distribution with mean 0 and variance $2/n$, where n is the number of input units in the weight tensor. Glorot (or Xavier) initialization (Glorot and Bengio, 2010) is designed to work well with a variety of activation functions, including *sigmoid*, *tanh*, and $ReLU$. Glorot initialization

sets the initial weights of each layer using a Gaussian distribution with mean 0 and variance 2/(n_in + n_out), where n_in and n_out are the numbers of input and output units in the weight tensor, respectively.

Baseline model: augmented training and hyperparameter tuning

As an initial model, an arbitrary architecture with arbitrary parameters was considered and trained using GSV images resized to 224×224 and normalized in the range $[0, 1]$. As previously explained in the methodology (Chapter 3), in addition to considering simple rescaled images as input, also augmentation was applied to the training dataset. Image augmentation is useful for training CNNs for several reasons (Perez and Wang, 2017b) but the most important are the increase of the number of data and robustness improvement. Image augmentation allows for the creation of new training examples from existing ones. This is especially useful when the size of the available training dataset is small. By generating new images with different variations, the CNN is exposed to more diverse examples, which can help to improve its ability to generalize to new, unseen images. That lead also to an improve the robustness of a CNN to variations in the input data. By exposing the network to augmented images that simulate common variations, such as rotation, scaling, and translation, the network learns to recognize the same object or pattern under different conditions. This helps to improve the generalization ability of the CNN, making it more robust to variations in the real-world images. However, inappropriate augmentation techniques could not help to improve the performance of the model. For example, if the images in the dataset have a specific orientation or aspect ratio, then applying random rotations or scaling may not be helpful. For this reason, all techniques that could have obscured or modified the facade of the building too much were not considered. The variations implemented are the height and width shift, rotations, vertical and horizontal flip, shear variations with a *nearest* fill mode that determines how the pixels in an image are filled when the image is transformed and the size of the image changes.

When building a deep learning model, the choice of its architecture, i.e., the number of layers, the size of the layers, the type of activation functions, etc., is crucial for its performance. Selecting the best architecture can be challenging, and it is often necessary to try several architectures and hyperparameters before finding the optimal one. Since the baseline architecture was designed manually and arbitrarily, it was better to find a way to tune at least the most important parameters that compose the structure. *KerasTuner* has been leveraged to increase the performance. It is an open-source hyperparameter tuning library for Keras that automates the process of hyperparameter tuning and architecture search by searching over a defined search space of hyperparameters using several search

strategies, such as random search, Bayesian optimization, or hyperband. This approach makes it easier and more efficient to find the optimal architecture for the model, without the need for arbitrary guessing. It works by creating a set of candidate models with different hyperparameter values and training them on a small subset of the data. The performance of each model is then evaluated using a chosen metric. Based on the results, KerasTuner decides which hyperparameters to explore further in the search space. The process is repeated until the optimal set of hyperparameters is found. One of the main advantages of using that method is that it allows you to explore a wider range of architectures and, at the same time, saves a lot of time and computational resources. Tuning the hyperparameters manually can be a time-consuming process, and it may require training hundreds or even thousands of models. The hyperparameters chosen to be tuned include the number of filters for the 5 convolutional layers and the number of units in the two final dense layers. The searching strategy for hyperparameter tuning chosen was hyperband (Li et al., 2016) which is designed to be more efficient than traditional methods such as random search or grid search. Hyperband is a sequential optimization algorithm that uses a technique called successive halving to quickly identify promising hyperparameter configurations. The algorithm works by first training a set of candidate models on a small subset of the data for a short period of time. After each round of training, half of the worst-performing candidate models are eliminated, while the best-performing models are trained again on a larger subset of the data for a longer period of time. This process is repeated until only one candidate model remains. Even though KerasTuner is a powerful tool for hyperparameter tuning, it can be difficult to use if the research space is too wide. This is because the number of possible hyperparameter combinations grows exponentially as the number of variables to tune is increased. As a result, the search space becomes very large, which can make it difficult and time-consuming to find the best set: it requires a large number of computational resources and time to explore the search space thoroughly. This can become impractical, especially for deep learning models that are already computationally intensive. To avoid these issues, it is important to carefully select the hyperparameters to tune and limit their number. It is also important to properly set the range of values for each hyperparameter to search over, as well as the number of trials to perform. These settings can significantly affect the performance of the model and the time required for tuning.

VGG16: transfer learning and fine tuning

VGG16 is trained using both transfer learning and fine tuning techinques. The images are preprocessed using specific function that performs the following changing steps on the

input images: it will convert the input images from RGB to BGR and then, then will zero-center each color channel with respect to the ImageNet dataset (which was used to pretrain the VGG16 model) by subtracting the RGB mean values (i.e., $[103.939, 116.779, 123.68]$) from the pixel values of each image channel. The purpose of this preprocessing function is to make the input images compatible with the format of the ImageNet dataset used to train the VGG16 model, which helps to improve the performance of the model on new data. It is important to apply this preprocessing function to any new images that are being fed into the VGG16 model for inference.
The transfer learning process involves removing the last few layers of the pretrained model and replacing them with new layers that are specific to the new task. The idea is that the earlier layers of the pretrained model have learned general features that are useful for many image classification tasks, while the later layers have learned task-specific features that may not be as useful for the new task. So the weights of the earlier layers of the pretrained model are kept fixed and only the weights of the newly added layers are updated during training. The last fully connected original VGG16 layer, which is typically used for classification, is removed and new layers are added. The architecture is structured in this way: the VGG16 model is first loaded with the specified input shape (224×224) and its layers are set to non-trainable. Then, the model is extended by adding a *global average pooling* layer (Lin et al., 2013) to reduce the spatial dimensions, followed by two fully connected (*Dense*) layers with 384 and 256 neurons respectively. Each Dense layer is followed by a *BatchNormalization* layer (Ioffe and Szegedy, 2015), an Activation layer with *ReLU* activation function, and a *Dropout* layer (Srivastava et al., 2014). Finally, a Dense layer with a single neuron and linear activation function is added as the output layer.
On the other hand, fine tuning, involves taking a pretrained CNN and training all of its layers, in addition to the new ones introduced, on a new dataset. In the training procedure, only the first 4 layers of the VGG16 architecture that involves the first two convolutional layers were frozen: they were able to extract interesting features for the ImageNet dataset, so it would be better to do not modify their weights. All the other layers were considered trainable during the procedure.

Resnet50: transfer learning and fine tuning

What has been said for VGG16 also applies to ResNet50. The architecture is based on a series of convolutional and pooling layers, with the addition of residual connections between some of the layers. In addition, the architecture also includes *skip connections* that bypass some of the convolutional layers, allowing the network to learn features at

different scales. Even in this case, there exists a preprocess function to apply to the input images that performs the following operations on the input image: subtracts the mean pixel value of the ImageNet dataset from each pixel and finally reverses the order of the color channels from RGB to BGR. These preprocessing steps are important because they ensure that the ResNet50 model receives input data that is consistent with how it was originally trained on the ImageNet dataset. This helps to maximize the accuracy of the predictions made by the model on new images.
The initial layers of the pretrained ResNet50 model are used as a feature extractor to extract high-level features from the input images. The last layer of the ResNet50 model is removed, and a *global average pooling* layer is added to reduce the spatial dimensions of the output features. After the global average pooling layer, there is a *Dense* layer with 512 neurons, followed by *BatchNormalization*, *ReLU* activation, and *Dropout* regularization. Finally, there is an output dense layer with a single neuron and a linear activation function to predict the building height.
Differently from the VGG16, all the ResNet50 pretrained layers were unfrozen. The weight of the whole ResNet50 trained on ImageNet is exploited as a starting point for training on a new objective. Differently from the previous models, the learning rate was modified from $1e^{-3}$ to $1e^{-4}$. This is because the pretrained weights have already learned useful features, and the objective is to adjust the new weights slowly to avoid overwriting the pretrained features.

Place365-VGG16: transfer learning and fine tuning

Places365-VGG16 is a CNN model that has been pretrained on the Places365 large-scale dataset containing multiple scene categories.
Using different pretrained models on new data apart from ImageNet could be an improvement since the Places365-VGG16 could have learned different features from the scene images. For instance, multiple scenes recognized concern the identification of the captured building (church, office, museum, etc.) and the features extracted to perform the task could be similar to the ones useful for the building height estimation. The pretrained model is first loaded with the specified input shape (224×224) and then the output is extended using a *global average pooling* layer. Two fully connected (*Dense*) layers with 128 and 654 neurons respectively followed by a *BatchNormalization* layer, an Activation layer with *tanh* and *ReLU* activation function, and a *Dropout* layer precedes the final layer: a single neuron and linear activation function.
As well as the transfer learning training, no preprocess function was applied to the GSV images in input during the fine tuning training. The first 10 layers of the Places365-

VGG16 model were not set as trainable while the others layer were considered modifiable.

4.5.2. Water consumption estimation: regression training setup

Initially, a baseline machine learning model will be established, incorporating only the building's height and area as features. Subsequently, the model will be further improved by incorporating socio-demographic features combined with area and height to achieve a more accurate estimation of the daily water consumption value. Both the final and baseline model are trained using the same procedure explained in the next lines. The dataset consists of 1699 samples, 80% of which will be used as the training set and the remaining portion as the test set. The training set will be further divided into different inner and outer folds to apply the *nested cross-validation* method discussed in chapter 3. Before starting the model training, outliers are identified using three different methods: *local outlier factor* (Breunig et al., 2000), *isolation forest* (Liu et al., 2008), and *interquartile range*. All three methods will be used separately, and the one that yields the best result will be chosen. The same type of analysis was performed on the scaling methods. *Standard scaler*, *robust scaler*, and *minmax* were applied individually to the data, and only the best one will be taken into consideration. The nested cross-validation with *GridSearchCV* method will return the best parameters for the algorithm used on the dataset. For the *linear regression*, the parameter chosen to be tuned is the *fit intercept*: it determines whether or not to include an intercept term in the model. The intercept is the value at which the regression line crosses the axis when all the independent variables are zero. The same information is inferred for the *polynomial regression* algorithm. In addition, for that algorithm the *degree* is a fundamental parameter: specifically, it determines the number of features that are created from the original input features and how flexible the model will be in capturing the relationship between the input features and the target variable. A higher degree polynomial will generally be able to fit the training data more closely, but it may also overfit the data and perform poorly on new data. On the other hand, a lower degree polynomial may not fit the training data as closely, but it may generalize better to new data. Regarding the *KNN*, the best *number of neighbors* was extracted: it represents the number of neighboring data points to consider when making a prediction for a new data point. In KNN, the predicted class or value for a new data point is determined by the majority class or the average value of its k-nearest neighbors in the training dataset. For both *Random Forest* and *XGBoost*, there are two parameters in common: one is the *number of estimators* that specifies the number of decision trees to be included in the forest. Each decision tree in the forest is constructed independently of the others, using a random subset of the training data and a random subset of the features. The final

prediction of the random forest is obtained by averaging the predictions of all the trees in the forest determined by the tuned value. The other parameter is the max depth of the constructed tree: it controls the *maximum depth* of each decision tree in the forest. The depth of a decision tree refers to the number of splits required to classify a sample. Increasing the value of the depth allows the tree to capture more complex relationships in the data, but also increases the risk of overfitting. In the XGBoost algorithm, also the best *learning rate* was estimated. Obviously, apart from the ones previously listed, there exists also other parameters to tune for the algorithm involved. Nonetheless, when there are many hyperparameters, the number of combinations to be evaluated grows exponentially, which quickly leads to a combinatorial explosion in the number of models fits that need to be computed. This can make the process of searching for the optimal hyperparameters infeasible or take an unreasonable amount of time. For that reason, only the most relevant ones were chosen.
The tuned parameters will be given in input for the ultimate models retrained from scratch on the entire training set and then evaluated on the test set.

4.5.3. Water consumption estimation: classification training setup

The target of the regression, i.e. the DWC, was discretized into three intervals representing low, intermediate or high water consumption for that particular building: the division of the classes took place via percentiles as explained in the methodology chapter 3. The 33rd percentile which defines the upper limit for the "low" water consumption set assumes a value of 4.19 m^3 while the 66th percentile that is the lower bound for the "high" water usage set is equal to 10.46 m^3. The water usage within those two limits identifies residential buildings belonging to the "medium" set.
Nested cross validation was used to find the best parameters for each algorithm: for DecisionTree-based algorithms such as *Random Forest*, *Extremely Randomized Tree* and *AdaBoost* the main parameters are the number of estimators and the maximum depth of the tree. For the *KNN* it is the number of neighbors to consider for the classification while for the *Gaussian Naive Bayes* there is a small constant value to tune and add to the variance of a Gaussian distribution in order to prevent division by zero errors and improve the accuracy of a Gaussian Naive Bayes classifier. For the *Logistic regressor* the type of penalization is inferred: it is a technique used to avoid overfitting the model by adding a penalty term to the likelihood function. The two most common forms of penalization used in logistic regression are Lasso (Tibshirani, 1996) and Ridge (Hoerl and Kennard, 2000).

4.6. Performance metrics

During the methodology evaluation, different metrics were considered, specific for each defined model.

Loss and *metric* were the same for all the CNNs. Loss refers to the measure of how well the CNN model performs in predicting the output for a given input. It represents the difference between the predicted output and the true output, also known as the ground truth. The goal of CNN training is to minimize the loss, i.e., to make the predicted output as close as possible to the true output. There are several types of loss functions that can be used in CNN training, since the outcome is an integer variable, the loss used for all the CNNs is the *Mean Squared Error* (MSE). Metrics, on the other hand, are used to evaluate the performance of the CNN model during and after training. Unlike loss, which is used to optimize the model, metrics are used to measure how well the model is performing. The metric used is the *Mean Absolute Error* (MAE):

$$MAE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{4.1}$$

where y represents the real target and $\hat{y}$ the estimation. During CNN training, both loss and metrics are calculated at each iteration, and the optimizer adjusts the weights and biases of the model to minimize the loss. The goal is to achieve a model with low loss and high performance on the chosen metric.

For evaluating the regression machine learning algorithms, *R-squared* (R^2) and *MAE* were calculated. R^2, or the coefficient of determination, is a statistical metric that represents the proportion of variation in the dependent variable that is explained by the independent variables in a regression model. The formula for R^2 is:

$$R^2 = 1 - \frac{RSS}{TSS} \tag{4.2}$$

$$RSS = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{4.3}$$

$$TSS = \sum_{i=1}^{N} (y_i - \overline{y})^2 \tag{4.4}$$

where the *residual sum of squares* (RSS) represents the squared difference the sum of

the differences between the predicted $\hat{y}$ and actual values y of the dependent variable. The *total sum of squares* (TSS) calculates the sum of the squared differences between the actual values and the mean value of the dependent variable. An R^2 value of 1 indicates a perfect fit, while a value of 0 indicates that the model does not explain any of the variations in the dependent variable. An R^2 value between 0 and 1 indicates the proportion of the variation in the dependent variable that can be explained by the independent variables in the model.

The metrics used to evaluate the performance of the classification models in this thesis are accuracy, precision, recall, and F1 score. Accuracy measures the overall correctness of the model's predictions, while precision measures the proportion of true positives among all instances that are classified as positive. Recall measures the proportion of true positives that are correctly identified by the model, while the F1 score is the harmonic mean of precision and recall, providing a balanced measure of both metrics. These metrics are calculated using the following formulas:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.5}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.7}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.8}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. These metrics will be used to compare and select the best performing classification model in the experiments conducted in this thesis.

5 | Results

In this chapter, the results obtained by running the CNNs architecture designed and the final ML models for water consumption estimation are reported and discussed. The analysis of the results will be useful to evaluate the performance of the methods and define possible improvements.
In summary, first, the corresponding GSV image is requested and fed as input to the Place365-VGG16 CNN to classify it as valid or invalid depending on whether it correctly represents the structure. The performance of this CNN will not be analyzed as it is already a fully pre-trained model with assigned weights. The relevant performance can be found in the corresponding paper (Zhou et al., 2017).
Second, once the valid GSV images are filtered, the building height is estimated from them using different CNNs architectures whose performances will be analyzed in this chapter.
Third, the results of the final ML models will be discussed.
In addition, the performance of the classifiers will also be discussed to evaluate how accurately the models are able to estimate the level of consumption of a building between low, medium and high.

5.1. CNNs performance for building height estimation

5.1.1. Baseline CNN performances

Standard training

Figure 5.1 represents the training and validation loss and metric. As can be seen, the training loss decreases as the model learns to fit the training data better. However, the validation loss, after an initial descending phase, starts to increase at a certain point even if the training loss continues to decrease. At that point, the early stopping interrupts the training process when the validation loss starts to increase, instead of continuing until the training loss reaches zero. The MAE and MSE (Table 5.1) achieved will be useful as a

baseline performance to improve in the successive steps.

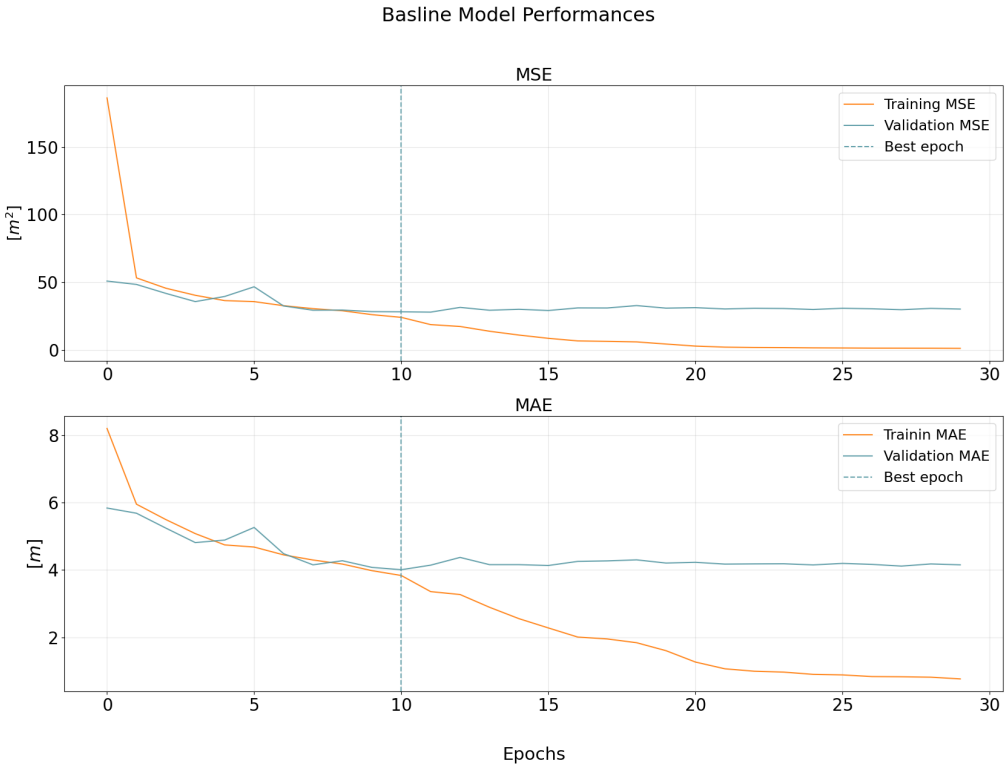


Figure 5.1: Baseline CNN training and validation loss and metric.

Standard training performances

	MAE [m]	MSE [m²]
Validation Set	4.01	27.81
Test Set	4.14	29.63

Table 5.1: MSE and MAE reached by the baseline CNN on validation and test set.

Augmented training

Interestingly, the MSE achieved through augmentation is lower in both test (28.66) and validation set (27.52 m) with respect to the standard training (29.63 m and 27.81 m) while the MAE is higher (Table 5.2). Due to augmentation, the model took longer to converge during training since it needs to learn to recognize and generalize across the augmented images. Figure 5.2 shows that the best result was achieved around the 90th epoch differently from the standard training in which the training stops already at the 10th

one. The method is computationally demanding and it slightly increases the performance: the training process could be harmed simply because an invalid input is provided to the network. It's not so rare that flips or other image augmentations might prevent the model from understanding some common patterns: finding the best set of augmentation parameters could be a future improvement to apply.

Augmented training performances

	MAE [m]	MSE [m^2]
Validation set	4.03	27.52
Test set	4.19	28.66

Table 5.2: MSE and MAE reached by the standard CNN on validation and test set using augmentation.

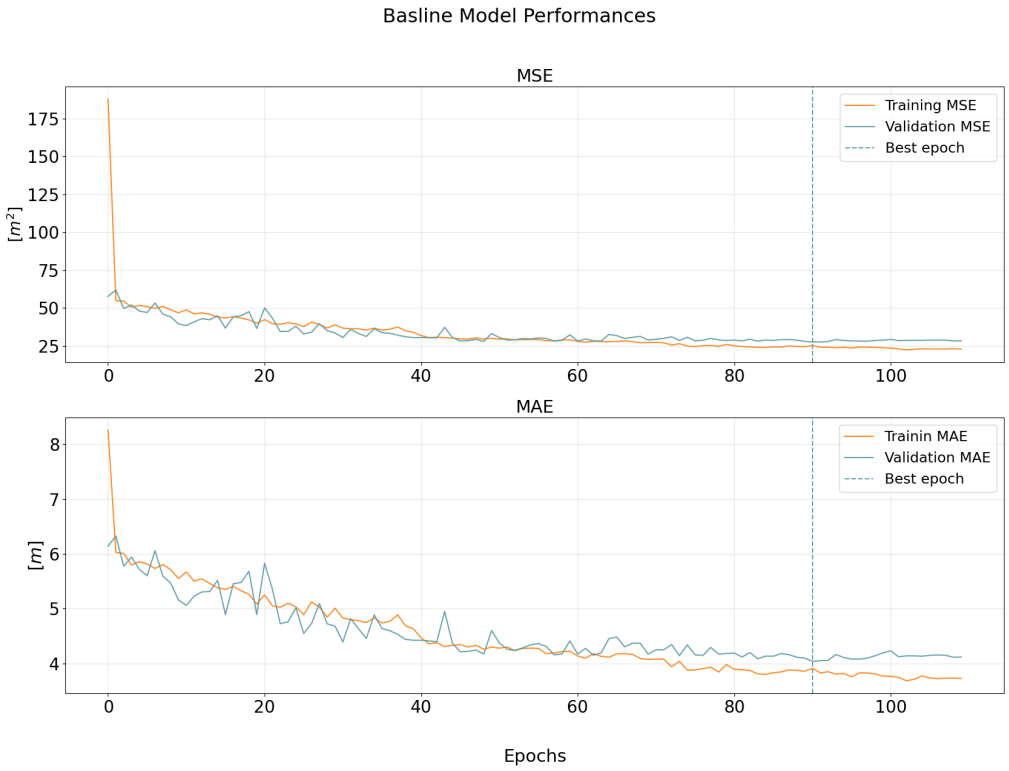


Figure 5.2: Baseline CNN augmented training and validation loss and metric.

Hyperparameter Tuning

Looking at the Table 5.3 and Figure 5.3, the MAE has slightly decreased on the test set compared to the augmented and standard training, demonstrating the effectiveness of the method. Given a complex model with many hyperparameters, effective hyperparameter tuning may drastically improve performance: in this case, the baseline architecture is quite simple and the number of parameters tuned is limited due to the computational resource requested. If a wider hyperparameter search space could be defined, the performance could be better.

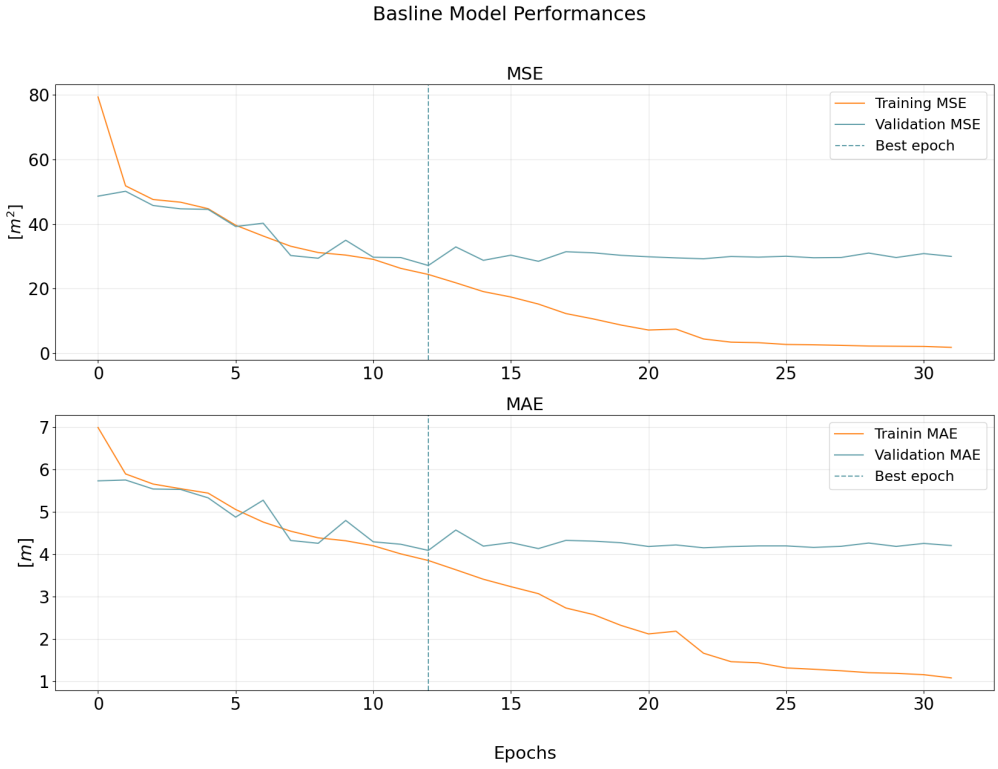


Figure 5.3: Tuned baseline CNN training and validation loss and metric.

Training with tuned hyperparameters performances

	MAE [m]	MSE [m²]
Validation set	4.08	27.19
Test set	4.09	28.84

Table 5.3: MSE and MAE reached by the tuned baseline CNN on validation and test set.

5.1.2. VGG16 performances

Transfer learning

Looking at the result in Table 5.4 and Figure 5.4, transfer learning on VGG16 is not performing as well as expected. The MAE (4.25 m) and MSE (31.51 m) were even higher than the standard trained baseline model. This approach works well when the pretrained model has learned relevant features that can be applied to the new dataset. This is one of the reasons why fine-tuning was introduced: unfreezing also the original layers could improve the performance and adapt the architecture to the new task, estimating the height of the building giving the GSV image.

Transfer learning VGG16 performances

	MAE [m]	MSE [m^2]
Validation set	4.17	28.76
Test set	4.25	31.51

Table 5.4: MSE and MAE reached by VGG16 on validation and test set.

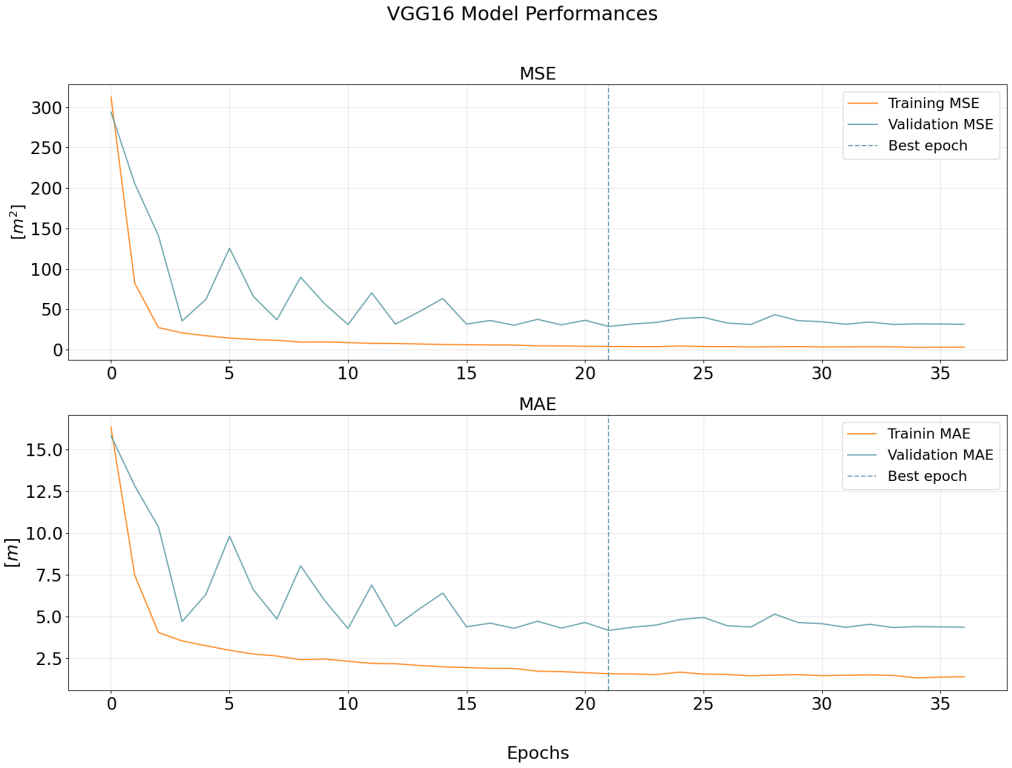


Figure 5.4: VGG16 transfer learning training and validation loss and metric.

Fine tuning

The performance of fine-tuned VGG16 (Table 5.5) are better than the transfer learning, but still not superior compared to the base model. Looking at the Figures 5.5 and 5.4, since more layers are being trained, fine-tuning took longer to converge than transfer learning.

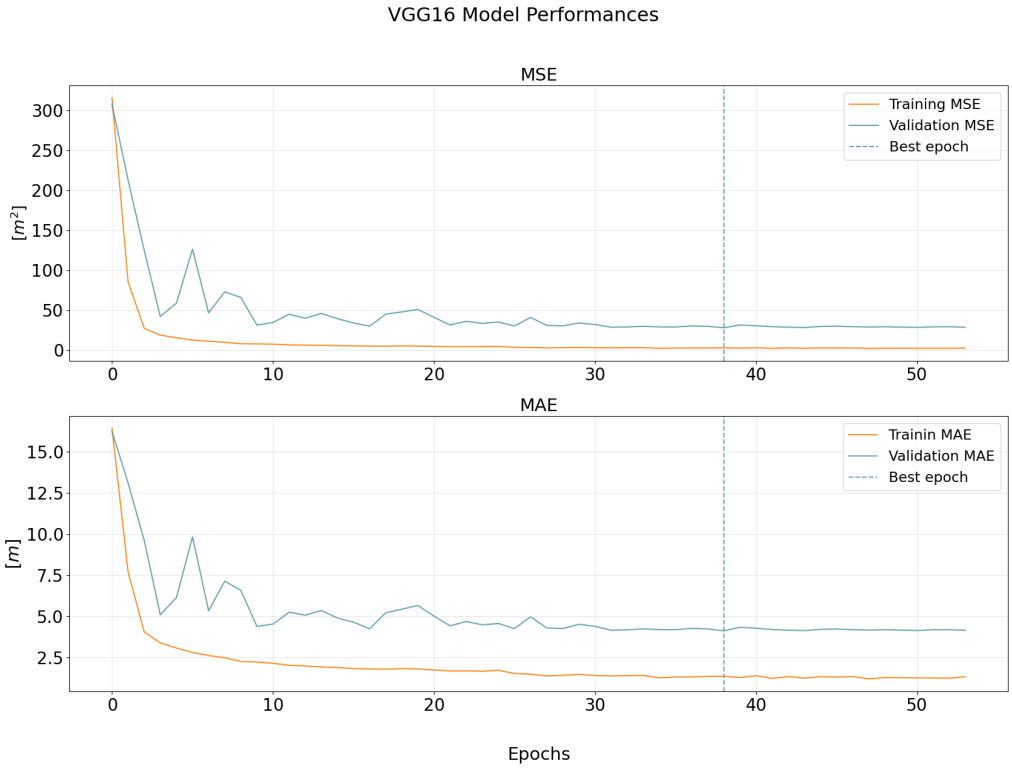


Figure 5.5: VGG16 fine tuning training and validation loss and metric.

Fine tuning VGG16 performances

	MAE [m]	MSE [m²]
Validation set	4.11	28.02
Test set	4.18	30.49

Table 5.5: MSE and MAE reached by fine-tuned VGG16 on validation and test set.

5.1.3. ResNet50 performances

Comparing the VGG16 and Resnet50 architectures and relative performances (Table 5.4), the transfer learning technique seems more effective to VGG16 model. Interestingly,

looking at the Figures 5.4 and 5.6 the training convergence is slower in the VGG16 due the higher number of trainable parameters considered.

Transfer learning

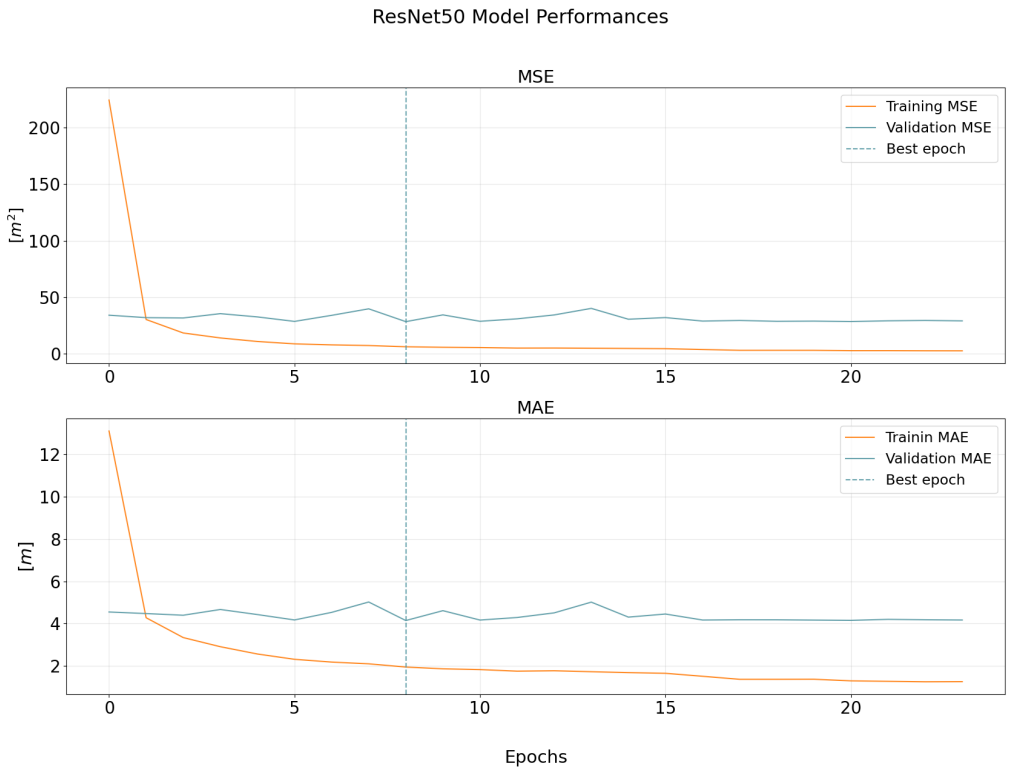


Figure 5.6: ResNet50 transfer learning training and validation loss and metric.

Transfer learning ResNet50 performances

	MAE [m]	MSE [m^2]
Validation set	4.14	28.34
Test set	4.31	29.87

Table 5.6: MSE and MAE reached by ResNet50 on validation and test set.

Fine tuning

Table 5.7 shows that the fine tuned ResNet50 reached best performances so far both in validation and test set exceeding the baseline model results with tuned hyperparameters.

The ImageNet and ResNet50 architecture result being effective for the height estimation task.

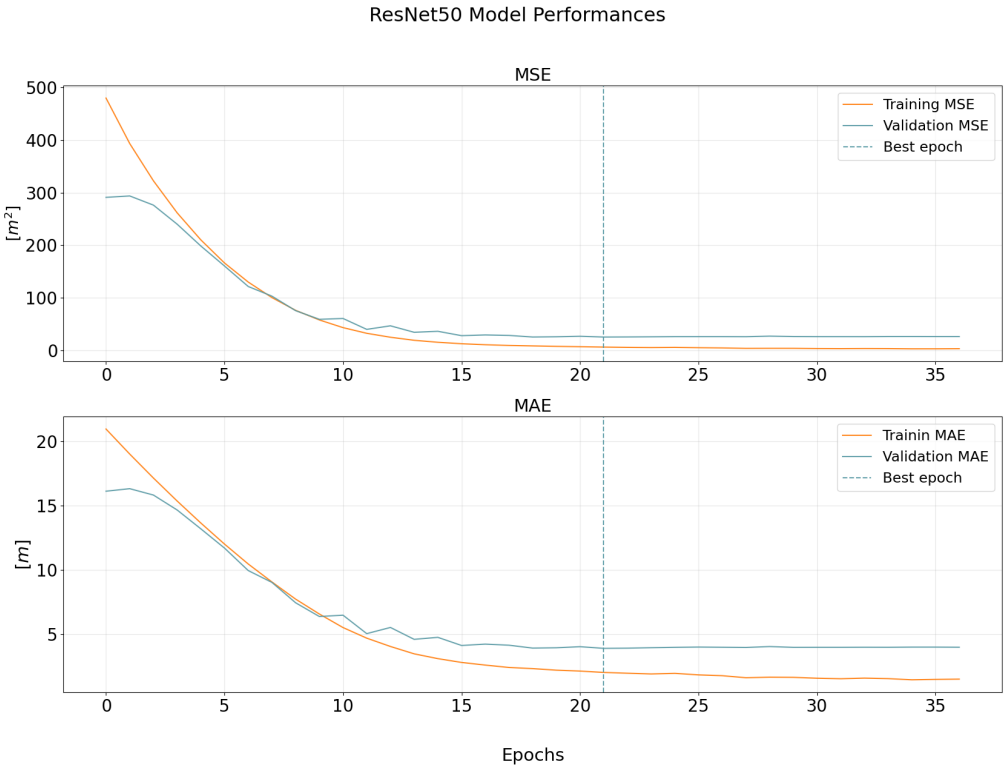


Figure 5.7: ResNet50 fine tuning training and validation loss and metric.

Fine tuning ResNet50 performances

	MAE [m]	MSE [m²]
Validation set	3.87	25.70
Test set	4.05	27.70

Table 5.7: MSE and MAE reached by fine-tuned ResNet50 on validation and test set.

5.1.4. Place365-VGG16 performances

Transfer learning

The results described in Table 5.8 and the linear loss decrease in Figure 5.9 show a significant relationship between the pretrained model Places365-VGG16 and the newly assigned task regarding the building height estimation particularly in the validation set

but the test set performances are lower than the ResNet50 and VGG16 architecture trained on ImageNet.

Transfer learning Places365-VGG16 performances

	MAE [m]	MSE [m^2]
Validation set	4.00	27.58
Test set	4.21	30.90

Table 5.8: MSE and MAE reached by Places365-VGG16 on validation and test set.

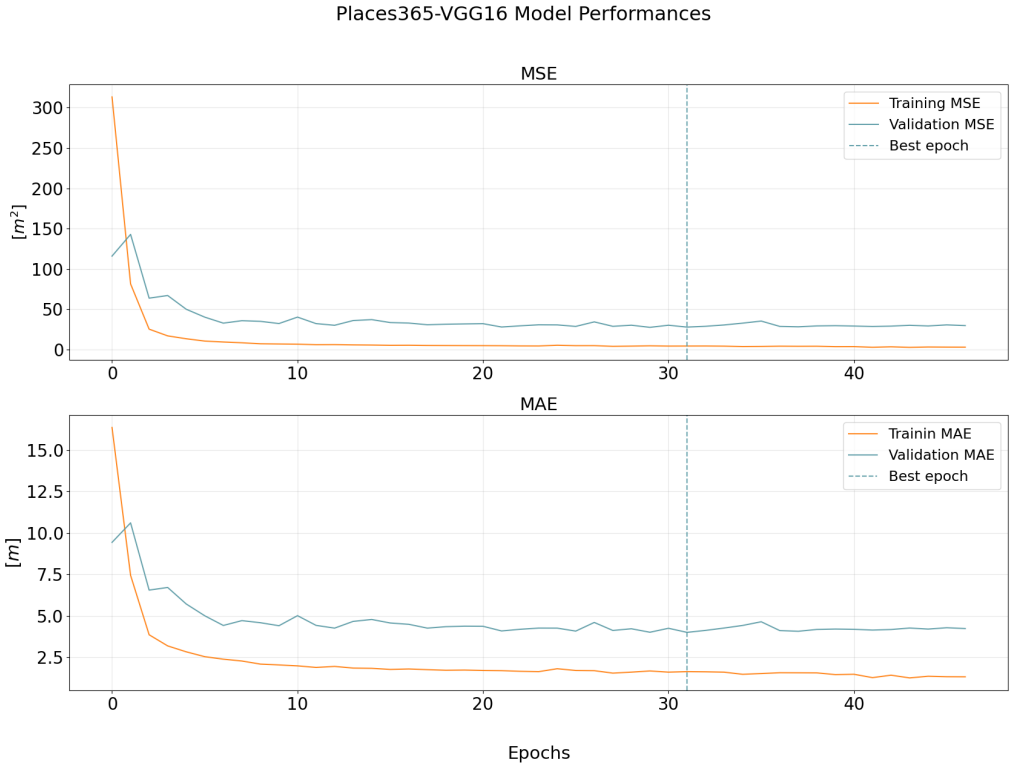


Figure 5.8: Places365-VGG16 transfer learning training and validation loss and metric.

Fine tuning

The fine-tuning results (Table 5.9) did not increase as much as it was expected, also considering the strict relationship that exists between the Places365 dataset about the exterior building and height estimation task and also the good transfer learning performances showed (Table 5.8). Different final layers were introduced in the architecture to

shift the original task, a future improvement could be to find the best hyperparameter through KerasTuner.

Fine tuning Places365-VGG16 performances

	MAE [m]	MSE [m²]
Validation set	4.15	28.44
Test set	4.20	30.74

Table 5.9: MSE and MAE reached by fine tuned Places365-VGG16 on validation and test set.

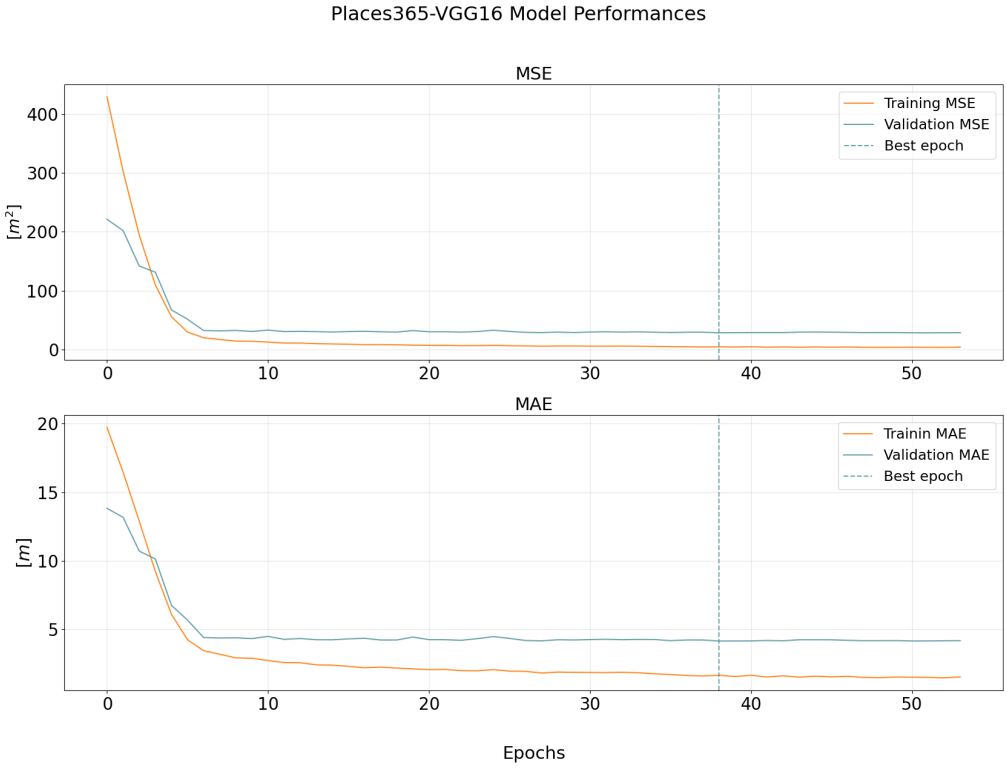


Figure 5.9: Places365-VGG16 fine tuning training and validation loss and metric.

5.1.5. Overall results

Table 5.10 shows the best results for each type of architecture. Regarding the baseline model, the best performance on the test set was achieved by training with the parameters selected by KerasTuner. The VGG16 architecture performed better using transfer learning technique and thus utilizing the weights learned from ImageNet as a basis for a new

training for the new task of predicting building heights. However, the result was not sufficient to outperform the simpler baseline model with tuned hyperparameters on the test set: it achieved a MAE of 4.09 m while the VGG16 achieved 4.18 m. ResNet50, on the other hand, achieved the best results on both the test (4.05 m) and validation set (27.70 m) through a retraining of all its layers, which brought the greatest benefits. The final pretrained Places365-VGG16 did not reach the same performances as the other CNN architectures, demonstrating that the ImageNet dataset was more related than Places365 to the final objective.
The CNNs estimations are slightly imprecise for two main reasons: despite the fact that invalid images were filtered out by the CNN Places365-VGG16, there are many GSV images that capture multiple buildings of different heights, or they are quite zoomed in and cannot capture the entire height of the building. Surely this can be a limitation in the final calculation of the target. In addition, as already mentioned in the chapter on data 4, the heights of the buildings extracted from the shapefile are not always accurate since a single building is composed of different volumes with different heights, and extracting the correct one is not trivial.

Overall CNN performances on validation and test set

	Test Set		Validation Set	
Model	**MAE [m]**	**MSE [m²]**	**MAE [m]**	**MSE [m²]**
Baseline CNN	4.09	28.84	4.08	27.19
VGG16	4.18	30.49	4.11	28.02
ResNet50	**4.05**	**27.70**	**3.87**	**25.70**
Places365-VGG16	4.20	30.74	4.15	28.44

Table 5.10: CNNs performances.

Overall, regardless of the architecture, results are always very similar and consistent, suggesting that all models are able to provide fairly good estimations. The mean MAE value on the test set is about $4m$ and so an uncertainty a little bit higher than 1 floor.
Figure 5.10 shows two GSV images fed in input to the ResNet50 architecture. All the GSV images are passed through a specific preprocess function before being fed into the ResNet50 CNN. That function will convert the input images from RGB to BGR, then will zero-center each colour channel with respect to the ImageNet dataset, without scaling. For that reason the image color is different from the original one.

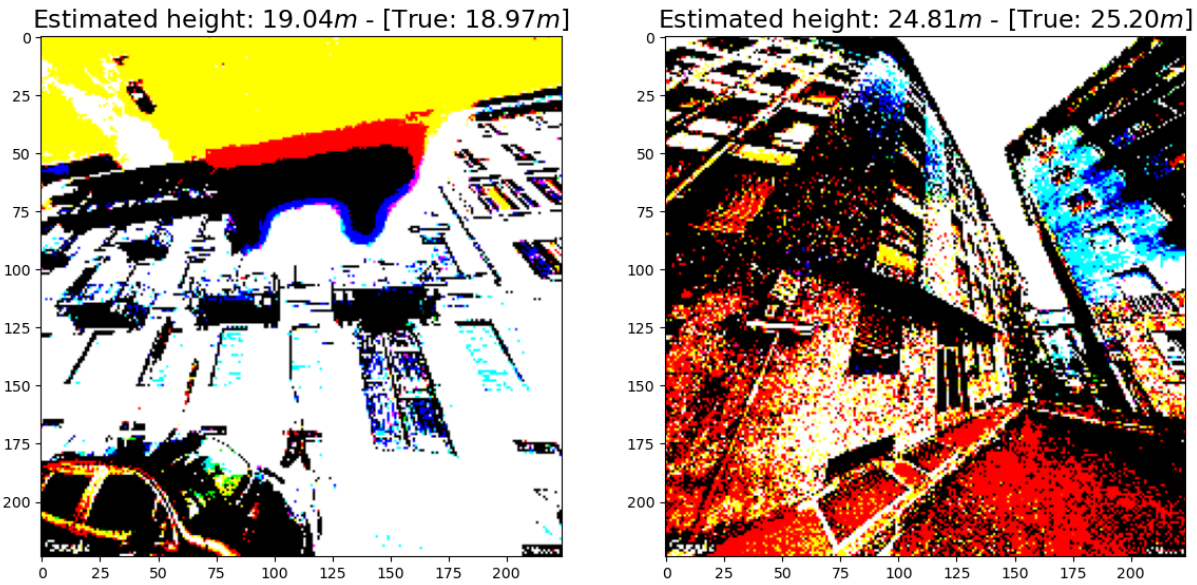


Figure 5.10: ResNet50 buildings height prediction on GSV images. On the left of the image title, it is reported the height estimated by the RseNet50 model while on the right it is showed the true height.

5.2. ML performances for daily water consumption estimation

5.2.1. Baseline model performances

	Name	Type/Unit
feature	Height	m
feature	Area	m^2
target	DWC	m^3

Table 5.11: Baseline model features and target.

The baseline ML model has only two features, i.e., building height (m) and area (m^2), while the target variable is the daily water consumption (DWC) (m^3) (Table 5.11). The performances shown in Table 5.12 suggest that polynomial regression achieves the best performance. Yet, all baseline models do not provide very accurate estimations of DWC in regression mode. We can explain this as due to the following reasons: first of all, trying to predict the water consumption of a building considering only its physical characteristics

might be limited. Water usage is closely related to human behavior and for this reason, it is complex to calculate it even knowing the smallest details.

Baseline models performances on test set

	Test Set	
Model	**MAE [m^3]**	**R2**
Linear Regression	5.08	0.32
Polynomial Regression	4.90	**0.38**
KNN	**4.78**	0.34
Random Forest	5.09	0.29
XGBoost	4.70	0.32

Table 5.12: Baseline models performances.

Additionally, it is also important to consider the margins of errors shown in Chapter 4 related to the building dimensions due to the shapefile. The height and area dimensions of the buildings are not always correct, and in addition, the number of samples is limited to 1699. There is thus an error propagation effect when such inaccurate inputs are passed on to the DWC estimation phase. The KNN reached the minimum MAE on the test set.

5.2.2. Final model regression performances

Apart from considering the building shape information about area and height, the final ML models are built considering also socio-demographic features at NIL level in the input set. Specifically, it is possible to retrieve the NIL each building belongs to and then associate the relative information. The number of families composed of 1, 2-4 or more than 5 members were considered as features as well as the different numbers of male, female, minors and elderly people for each single NIL (Table 5.13).

	Name	Type/Unit
feature	Height	m
feature	Area	m^2
feature	Single families	units
feature	Multi families	units
feature	Large families	units
feature	Females	units
feature	Males	units
feature	Minors	units
feature	65+	units
feature	Population density	$(people/km^2)$
target	DWC	m^3

Table 5.13: Final model features and target.

Looking at Table 5.14, the introduction of new features improves the overall performance of the models, with decision tree-based algorithms being the ones that benefited the most. On the other hand, KNN did not improve due to the curse of dimensionality: it refers to the phenomenon where the performance of the K-nearest neighbours (KNN) algorithm deteriorates rapidly as the number of features or dimensions in the data increases. As the number of features increases, the data becomes more sparse and the distance between data points becomes less meaningful. This means that the KNN algorithm will have difficulty finding meaningful clusters of points and identifying nearest neighbours. XGBoost achieved the best performance on the test set. The improvement due to the new features is not so crucial, but it demonstrates that by integrating demographic and social information, a better result can be achieved and so it assesses that kind of information is influential in water consumption estimation. The best scaling method reveals to be the Minmax one for all the algorithms apart from the Random Forest that shows superior performances applying the Standard Scaler. Regarding the outlier methods used, Local Outlier Factor and IQR lead to the best results for both the baseline models and final models.

Final models performances on test and validation set

	Test Set	
Model	**MAE [m³]**	**R2**
Linear Regression	4.66	0.28
Polynomial Regression	4.70	0.25
KNN	4.68	0.27
Random Forest	4.49	**0.39**
XGBoost	**4.19**	0.38

Table 5.14: Final models performances.

Figure 5.11 shows the feature importance analysis performed on the ML algorithms based on tree structure: Random Forest and XBGoost. Feature importance refers to a technique that calculates the importance of each feature for a given model. A higher score means that the specific feature will have a larger effect on the model that is being used to predict a certain variable, in this case, water consumption.

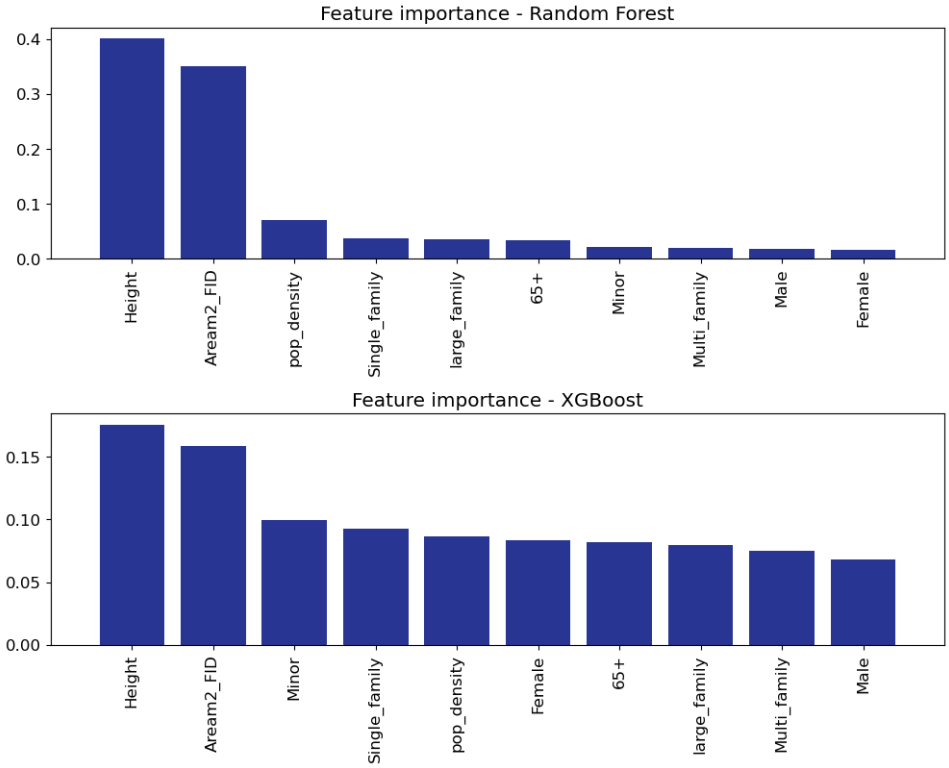


Figure 5.11: Features importance for the Random Forest and XGBoost algorithms.

Interestingly, the two algorithms exploit the feature's predictive power differently: Ran-

dom Forest evaluates as most significant the feature related to the building dimension while the socio-demographic ones are considered almost ineffective for the final estimation apart from the population density. On the other hand, XGBoost algorithm seems to balance the relevance of each feature even if there is still a certain difference between the height and area from the other ones. It should be noticed that the importance coefficient of building characteristics is small and it is not so far from the values of the socio-demographic factors differently from the Random Forest case. Thus, considering that the socio-demographic information is collected at NIL level and so there is a loss of information due to the lower resolution, these can be considered relevant in the final estimation, especially for the XGBoost algorithm.

5.2.3. Classification: training parameters and metrics

The features considered in input for the classification task are the same as the final regression model, so both the socio-demographic and the building dimension ones.

5.2.4. Final model classification performances

Figure 5.12 shows the accuracy calculated on the test set. Based on the results, the Bagging(Tree) algorithm achieved the highest accuracy on the test set, followed closely by KNN and Bagging(KNN). Random Forest also achieved decent performance, but the Extremely Randomized Trees and Ada Boost algorithms did not perform equally well. The models with the highest precision are Bagging(Tree), Bagging(KNN), KNN and Logistic Regressor (Figure 5.13), while the models with the highest recall are Bagging(Tree), KNN, and Bagging(KNN) (Figure 5.14). However, we should note that the values of these metrics vary depending on the specific class distribution and the threshold used to make predictions. Therefore, it is important to consider both precision and recall values together when evaluating the performance of a classification model, which is typically done using the F1 score. In Figure 5.15, the Bagging(Tree) model has the highest F1 score of 0.651, followed closely by Bagging(KNN) and KNN with F1 scores of 0.647 and 0.639, respectively.

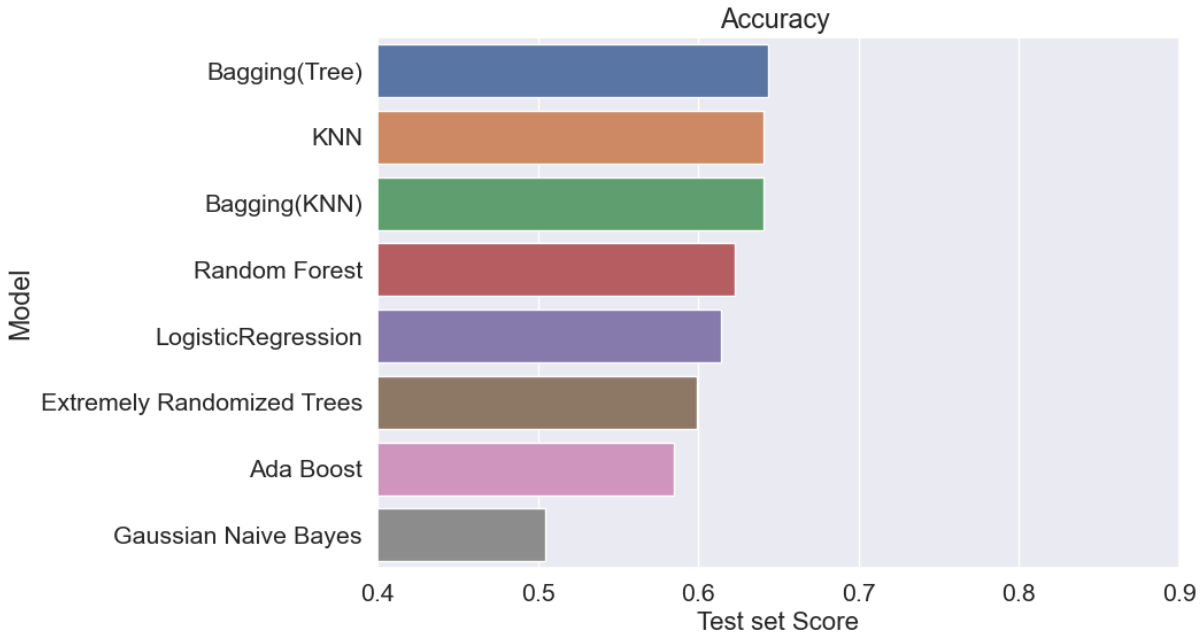


Figure 5.12: Comparison of models accuracy.

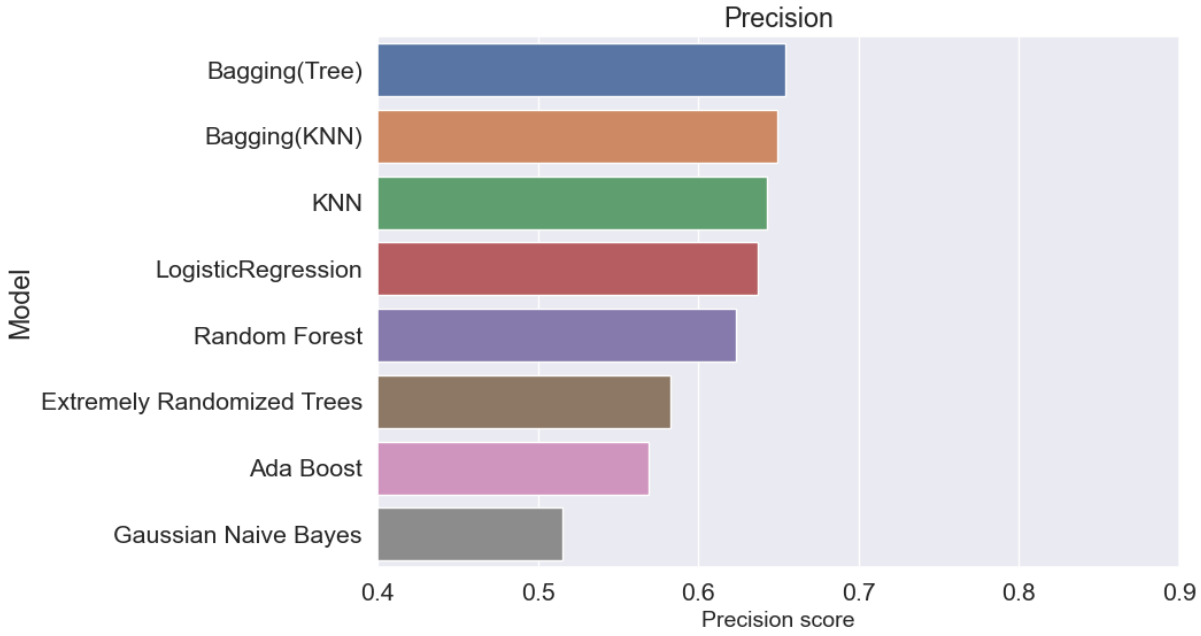


Figure 5.13: Comparison of models precision.

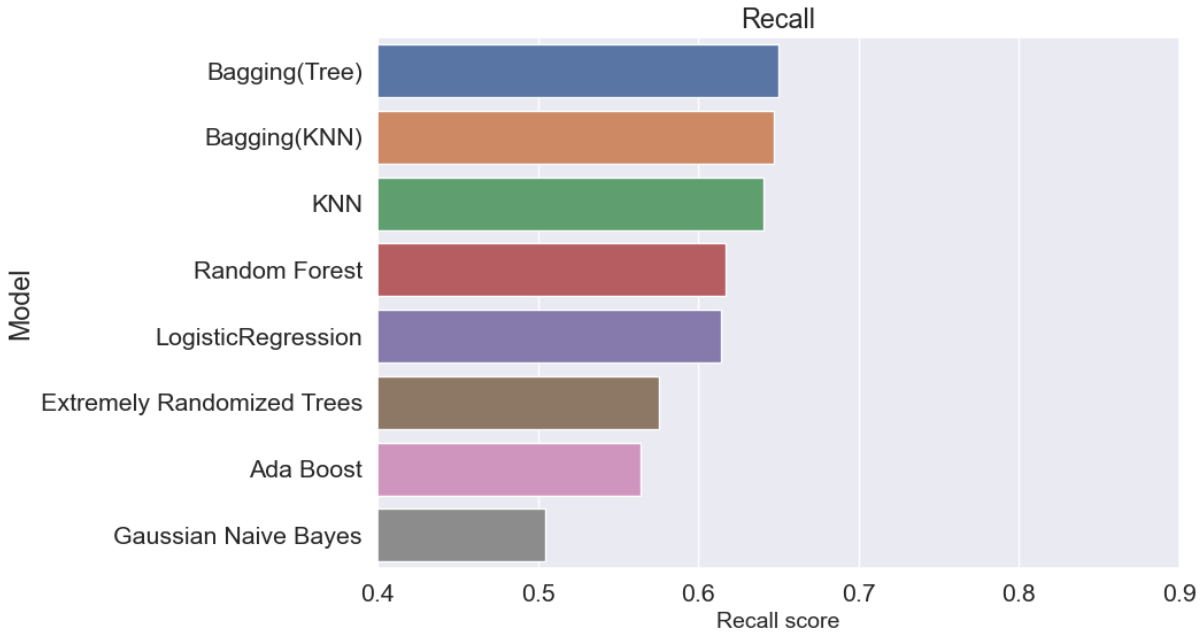


Figure 5.14: Comparison of models recall.

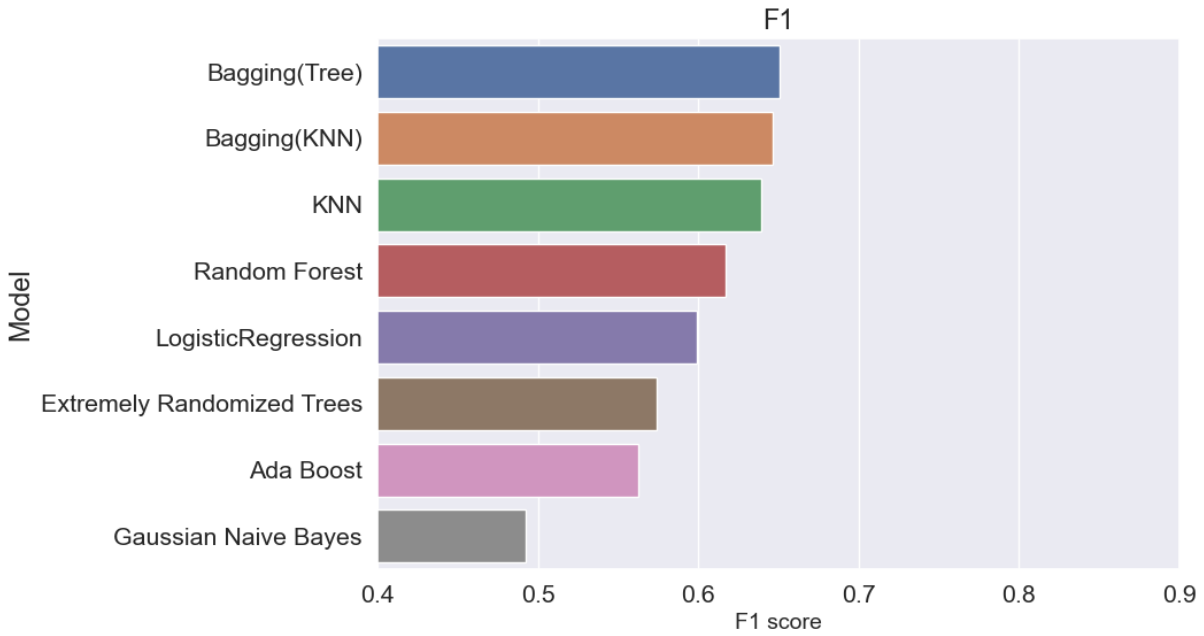


Figure 5.15: Comparison of models F1.

However, it is important to note that the differences between the top models in terms of precision, recall, and F1 are relatively small, so the choice of the model may depend on other factors such as interpretability, computational efficiency, or ease of implementation. Although the model is not accurate in predicting the exact continuous value of daily water

consumption, there is a consequent improvement in identifying residential buildings in a certain category depending on their water usage. During the discretization process, nearby values are merged into a common bin, losing a little bit of information and resolution. Thus, the classification algorithms can easily model the data distribution: this is the main reason that justifies a decent classification accuracy and a barely-sufficient regression performance.

5.3. Water consumption approximate formula: a comparison

In order to see if the effort spent in developing and implementing deep learning and ML techniques was justified in terms of improved accuracy, the performances of the previously described models are compared to the most basic water consumption estimation formula based on approximation regarding the area, number of stories, population per square meter and water consumption per capita per day. ISTAT (Italian National Institute of Statistics) is the main statistical institute in Italy. It is responsible for collecting, processing, and disseminating official statistics on the Italian economy, society, and environment. The most recent data on water consumption in Italy published by Istat is for the year 2020, which shows that the average daily water consumption per capita in Italy was 236 litres (ISTAT, 2022). However, as already explained, this is just an average and the actual water consumption can vary depending on a range of factors, including the region, the time of year, and the type of building but an approximative comparative formula is enough. The basic formula is expressed as a multiplication of several factors:

$$b\_dwc = per\_capita\_dwc * b\_area * b\_stories * NIL\_pop\_density \tag{5.1}$$

The final outcome of the formula (b_dwc) represents the building daily water consumption expressed in litre per day. b_area and $b\_stories$ are the dimensional features of the building expressed respectively in m^2 and unit. per_capita_dwc is the daily water usage per single person estimate by ISTAT while $NIL\_pop\_density$ determines the number of people per square meters for each single NIL. Since the building daily water consumption estimated by the ML models is expressed in m^3, the formula outcome is converted from litres to the target unit measure by dividing it by 1000.
With all the information at our disposal, the calculation of the average water consumption was applied to the entire dataset of 1699 samples. Obviously, since it is a simple formula, it does not require a training procedure but is simply applied to each row of the

dataframe. The estimated final quantity variable was eventually discretized into three fundamental classes according to the limits previously discussed. The purpose of this is to use the approximate formula as a classification depending on the level of water usage and to compare it to the actual labels already calculated when defining the classification problem. To make the comparison between the approximate formula and the best classification algorithm discovered (bagging with Decision Tree) more reliable, the portion of the dataset used as the test set in the algorithm training will be used for evaluation. It contains 337 samples.

The results show (Table 5.15) that the bagging tree model has higher accuracy, precision, recall, and F1 score compared to the approximate model. This indicates that the machine learning approach is providing better results than the approximate approach. The difference in accuracy between the two models is quite significant as can be seen looking at Image 5.16 and 5.17. Similarly, the difference in precision is 0.135, recall is 0.157, and F1 score is 0.213. These differences are meaningful and suggest that the bagging tree model is able to provide more accurate and precise predictions than the approximate model. These results justify the effort spent in researching a machine learning solution combined with deep learning techniques. Machine learning algorithms are able to capture complex relationships between the predictors and the outcome variable, and thus provide more accurate and precise predictions compared to approximate solutions. Furthermore, machine learning algorithms are able to learn from new data and adapt to changing circumstances, making them more versatile and useful in real-world applications. This is especially important in rapidly changing environments or fields where traditional approaches may not be able to keep up with the pace of change. While the results of the ML model are superior to the approximate model, there is still room for improvement as will be explained in the next chapter.

	Bagging (Tree)	**Approximate formula**
Accuracy	0.641	0.484
Precision	0.644	0.509
Recall	0.641	0.484
F1	0.642	0.429

Table 5.15: Comparison between best ML classifier developed and approximate formula.

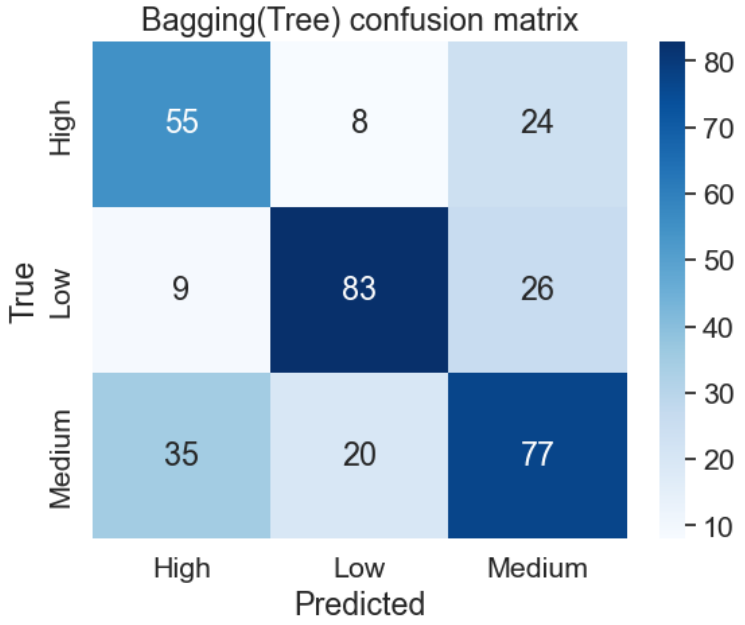


Figure 5.16: Bagging Tree confusion matrix.

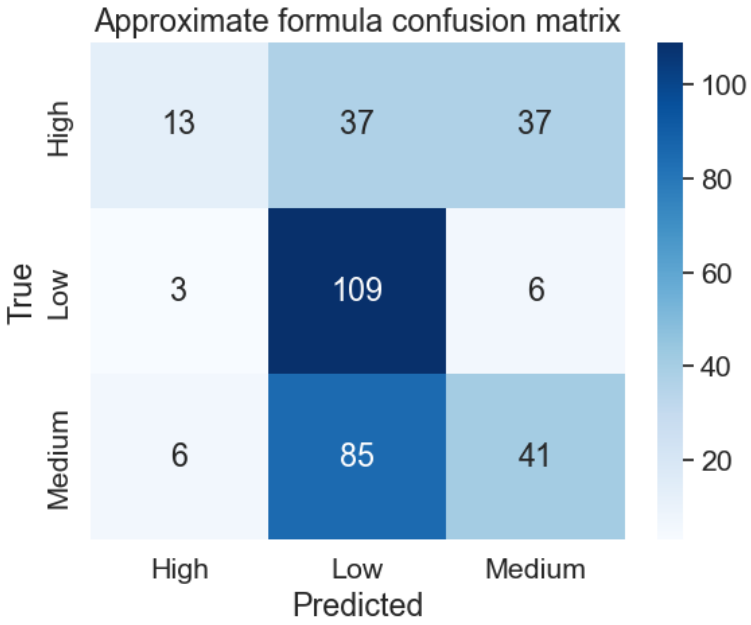


Figure 5.17: Approximate formula confusion matrix.

6 | Conclusion and future research

This thesis contributed a data-driven machine learning framework to predicting water consumption using publicly available data. Results show that it is possible to build a model that predicts water consumption based on a range of features, including building characteristics and socio-demographic information, which are retrieved from public databases. Given an address, three types of information were extracted, for each building: its Google Street View (GSV) image, its area, and socio-demographic features of the Nucleo d'identità Locale (NIL) where it is placed. Based on the above information, our machine learning framework than performs three steps. First, the GSV image is filtered and deemed as valid using the Place365-VGG16 architecture, a Convolutional Neural Network (CNN) pretrained on a subset of Place dataset, a repository of 10 million scene photographs, labeled with 434 scene semantic categories. Second, the image is fed in as input to different deep learning CNNs to estimate the relative height. State of art techniques were exploited to achieve the best possible performance. Third, once the target is retrieved, it was combined with the area and socio-demographic features for the definition of the final ML model to predict the daily water consumption of the building. Different machine learning methods were comparatively assessed to evaluate their suitability to predict building-level water consumption. While the regression and even more classification algorithms achieved decent performances explaining the weight of each feature in the final prediction, our results also suggest that there is room for improvement of those algorithms that did not perform as well. Future research could focus on identifying ways to optimize these algorithms or developing new methods that can outperform the current state of the art regarding both the ML and deep learning fields.

The analysis presented in this thesis demonstrates the value of using machine learning and deep learning techniques for water consumption prediction and highlights the potential of publicly available data sources for this type of study. Estimating the daily water consumption of residential buildings using machine learning and deep learning techniques could be beneficial for several reasons:

- Better understanding of water usage patterns: By using machine learning techniques

to analyze water usage data, we can gain a better understanding of how water is being used in residential buildings. This information can be used to inform public policy and educational campaigns aimed at reducing water waste and promoting conservation.

- Conservation of water: Estimating the daily water consumption of residential buildings can help identify areas where water is being wasted, allowing for targeted conservation efforts and leakage detection. By conserving water, it is possible to reduce the strain on the water supply and help ensure water availability.
- Cost-effectiveness: Residential buildings that use less water can save on their water bills. By identifying areas and buildings where water is being wasted and implementing changes to reduce consumption, building owners and tenants can save money.
- Improved infrastructure planning: Knowing how much water a residential building consumes can help water utilities plan for future infrastructure needs. By understanding water usage patterns, water utilities can ensure that they are able to provide sufficient water to all customers, even during times of high demand.
- Increased sustainability: Estimating the daily water consumption of residential buildings can help identify opportunities to make buildings more sustainable. For example, by identifying areas where water is being wasted, building owners can implement changes such as installing low-flow faucets, smart water sensors or water saving appliances to reduce water consumption.

One of the objectives of the thesis is to consider public data in order to make the model reproducible in any area of interest, specifically in those where the acquisition of water data is difficult due to the lack of monitoring tools or privacy issues. The CNN that takes as input GSV images allows the extraction of the height dimension of the building and that technique is quite reproducible considering different locations, provided that certain conditions exist; the presence of the GSV images must be ensured, and the buildings exterior structure of the new targets should be similar to Milan's residential buildings as them represents the training set for the CNN. One possible improvement could be to expand the dataset of GSV images to include different types of buildings and make the model more generalized. Regarding the footprint of the structures, during the development of the project, it was considered as input data thanks to the Milan shapefile. But for future improvement, satellite images could be used to extract the area of interest since the shapefiles may not be publicly available or may not exist for certain areas. Even if shapefiles do exist, they may not have accurate or up-to-date information on the boundaries of the study area or the locations of individual buildings within it. In this way, the

data related to the characteristics of the building would be dependent only to GSV and satellite images, two kinds of images that could be easily extracted on internet.
As described in the results chapter, CNNs for building height estimation showed more satisfactory results than the final model. To increase the performance of both models, obtaining higher-quality data is certainly useful. Public data may not provide detailed socio-demographic and building dimension information and most of them have low and aggregated spatial resolutions, which can limit the effectiveness of water consumption estimation. The public socio-demographic features were collected at NIL level and increasing the resolution of data can provide more accuracy. The same type of problem is encountered in the shapefile used for the physical characteristics of the buildings: some structures are merged with others in the same polygon reducing the resolution of the information.
Another area of improvement is to introduce temporal data such as weather data for estimating the water consumption level in different periods during the year: weather variables such as temperature, precipitation, and humidity can have a significant impact on water usage patterns, as they can affect factors such as evaporation rates, plant growth, and household behaviour. By incorporating weather data into water consumption models, it may be possible to more accurately predict water usage and identify patterns and trends.
The comparison evaluated between the developed final model and the most basic approximate water consumption estimation formula supports more complex approaches based on machine learning and data-driven techniques but at the same time, they can be expensive in terms of computational resources and development effort.

Bibliography

S. Amaral, A. M. V. Monteiro, G. Camara, and J. A. Quintanilha. Dmsp/ols night-time light imagery for urban population estimates in the brazilian amazon. *International Journal of Remote Sensing*, 27(5):855–870, 2006. doi: 10.1080/01431160500181861. URL https://doi.org/10.1080/01431160500181861.

F. Arbués, M. Ángeles Garcıa-Valiñas, and R. Martınez-Espiñeira. Estimation of residential water demand: a state-of-the-art review. *The Journal of Socio-Economics*, 32(1):81–102, 2003. ISSN 1053-5357. doi: https://doi.org/10.1016/S1053-5357(03)00005-2. URL https://www.sciencedirect.com/science/article/pii/S1053535703000052.

B. Bates, Z. Kundzewicz, and S. Wu. *Climate change and water*. Intergovernmental Panel on Climate Change Secretariat, 2008.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66, 02 1994. doi: 10.1109/72.279181.

J. Blumenstock, G. Cadamuro, and R. On. Predicting poverty and wealth from mobile phone metadata. *Science*, 350(6264):1073–1076, 2015. doi: 10.1126/science.aac4420. URL https://www.science.org/doi/abs/10.1126/science.aac4420.

S. Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44, 09 2020. doi: 10.31449/inf.v44i3.2828.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996. ISSN 1573-0565. doi: 10.1007/BF00058655. URL https://doi.org/10.1007/BF00058655.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 10 2001. doi: 10.1023/A:1010950718922.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, may 2000. ISSN 0163-5808. doi: 10.1145/335191.335388. URL https://doi.org/10.1145/335191.335388.

R. Brown, N. Keath, and T. Wong. Urban water management in cities: Historical, current and future regimes. *Water science and technology : a journal of the International Association on Water Pollution Research*, 59:847–55, 02 2009. doi: 10.2166/wst.2009.029.

G. C. Cawley and N. L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11 (70):2079–2107, 2010. URL http://jmlr.org/papers/v11/cawley10a.html.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. URL http://arxiv.org/abs/1603.02754.

A. Cominola, M. Giuliani, A. Castelletti, P. Fraternali, S. L. H. Gonzalez, J. C. G. Herrero, J. Novak, and A. E. Rizzoli. Long-term water conservation is fostered by smart meter-based feedback and digital user engagement. *npj Clean Water*, 4(1):29, May 2021. ISSN 2059-7037. doi: 10.1038/s41545-021-00119-0. URL https://doi.org/10.1038/s41545-021-00119-0.

A. Cominola, L. Preiss, M. Thyer, H. R. Maier, P. Prevos, R. A. Stewart, and A. Castelletti. The determinants of household water consumption: A review and assessment framework for research and practice. *npj Clean Water*, 6(1):11, Feb 2023. ISSN 2059-7037. doi: 10.1038/s41545-022-00208-8. URL https://doi.org/10.1038/s41545-022-00208-8.

R. Connor. *The United Nations world water development report 2015: water for a sustainable world*, volume 1. UNESCO publishing, 2015.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005. doi: 10.1109/CVPR.2005.177.

DataAnalytics. *Python – Nested Cross Validation for Algorithm Selection*, 2020. https://vitalflux.com/python-nested-cross-validation-algorithm-selection/ [Accessed: 16/01/2023].

Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports*, 3(1):1376, Mar 2013. ISSN 2045-2322. doi: 10.1038/srep01376. URL https://doi.org/10.1038/srep01376.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

A. Di Mauro, A. Cominola, A. Castelletti, and A. Di Nardo. Urban water consumption at multiple spatial and temporal scales. a review of existing datasets. *Water*, 13(1), 2021. ISSN 2073-4441. doi: 10.3390/w13010036. URL https://www.mdpi.com/2073-4441/13/1/36.

P. Diaz, P. Stanek, N. Frantzeskaki, and D. Yeh. Shifting paradigms, changing waters: Transitioning to integrated urban water management in the coastal city of dunedin, usa. *Sustainable Cities and Society*, 26, 04 2016. doi: 10.1016/j.scs.2016.03.016.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 07 2011.

C. D. Elvidge, K. E. Baugh, E. A. Kihn, H. W. Kroehl, E. R. Davis, and C. W. Davis. Relation between satellite observed visible-near infrared emissions, population, economic activity and electric power consumption. *International Journal of Remote Sensing*, 18(6):1373–1379, 1997. doi: 10.1080/014311697218485. URL https://doi.org/10.1080/014311697218485.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi: 10.1214/aos/1013203451. URL https://doi.org/10.1214/aos/1013203451.

S. Gato, N. Jayasuriya, and P. Roberts. Forecasting residential water demand: Case study. *Journal of Water Resources Planning and Management*, 133(4):309–319, 2007. doi: 10.1061/(ASCE)0733-9496(2007)133:4(309). URL https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9496%282007%29133%3A4%28309%29.

T. Gebru, J. Krause, Y. Wang, D. Chen, J. Deng, E. L. Aiden, and L. Fei-Fei. Using deep learning and google street view to estimate the demographic makeup of neighborhoods across the united states. *Proceedings of the National Academy of Sciences*, 114(50): 13108–13113, 2017. doi: 10.1073/pnas.1700035114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1700035114.

GeoportaleMilano. *Territory shapefiles*, 2012. https://geoportale.comune.milano.it/sit/tematiche/territorio/ [Accessed: 03/03/2023].

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*,

63(1):3–42, Apr 2006. ISSN 1573-0565. doi: 10.1007/s10994-006-6226-1. URL https://doi.org/10.1007/s10994-006-6226-1.

M. Ghiassi, H. Saidane, and D. Zimbra. A dynamic artificial neural network model for forecasting time series events. *International Journal of Forecasting*, 21(2):341–362, 2005. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2004.10.008. URL https://www.sciencedirect.com/science/article/pii/S0169207004001116.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

GoogleStreetViewAPI. *Overview of the Street View Static API*, 2023. https://developers.google.com/maps/documentation/streetview/overview?hl=it [Accessed: 03/04/2023].

T. Grippa, M. Lennert, B. Beaumont, S. Vanhuysse, N. Stephenne, and E. Wolff. An open-source semi-automated processing chain for urban object-based classification. *Remote Sensing*, 9(4), 2017. ISSN 2072-4292. doi: 10.3390/rs9040358. URL https://www.mdpi.com/2072-4292/9/4/358.

S. Guhathakurta and P. Gober. The impact of the phoenix urban heat island on residential water use. *Journal of the American Planning Association*, 73(3):317–329, 2007. doi: 10.1080/01944360708977980. URL https://doi.org/10.1080/01944360708977980.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015a. URL http://arxiv.org/abs/1512.03385.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015b. URL http://arxiv.org/abs/1502.01852.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000. ISSN 00401706. URL http://www.jstor.org/stable/1271436.

K. Hussey and J. Pittock. The energy–water nexus: Managing the links between energy and water for a sustainable future. *Ecology and Society*, 17(1), 2012. doi: 10.5751/ES-04641-170131. URL https://www.ecologyandsociety.org/vol17/iss1/art31/. 31.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL https://arxiv.org/abs/1502.03167.

ISTAT. *ISTAT WATER STATISTICS | YEARS 2019-2021*, 2022. https://www.istat.it/it/files//2022/04/Report_ISTAT-WATER-STATISTICS.pdf [Accessed: 24/03/2023].

N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794, 2016. doi: 10.1126/science.aaf7894. URL https://www.science.org/doi/abs/10.1126/science.aaf7894.

S. Keola, M. Andersson, and O. Hall. Monitoring economic development from space: Using nighttime light and land cover data to measure economic growth. *World Development*, 66:322–334, 2015. ISSN 0305-750X. doi: https://doi.org/10.1016/j.worlddev.2014.08.017. URL https://www.sciencedirect.com/science/article/pii/S0305750X14002551.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. doi: 10.1109/5.58325.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

E. F. Lambin and P. Meyfroidt. Global land use change, economic globalization, and the looming land scarcity. *Proceedings of the National Academy of Sciences*, 108(9):3465–3472, 2011. doi: 10.1073/pnas.1100480108. URL https://www.pnas.org/doi/abs/10.1073/pnas.1100480108.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

S. Leyk, A. E. Gaughan, S. B. Adamo, A. de Sherbinin, D. Balk, S. Freire, A. Rose, F. R. Stevens, B. Blankespoor, C. Frye, J. Comenetz, A. Sorichetta, K. MacManus,

L. Pistolesi, M. Levy, A. J. Tatem, and M. Pesaresi. The spatial allocation of population: a review of large-scale gridded population data products and their fitness for use. *Earth System Science Data*, 11(3):1385–1409, 2019. doi: 10.5194/essd-11-1385-2019. URL https://essd.copernicus.org/articles/11/1385/2019/.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. 2016. doi: 10.48550/ARXIV. 1603.06560. URL https://arxiv.org/abs/1603.06560.

X. Li and C. Zhang. Urban land use information retrieval based on scene classification of google street view images. In *SDW@GIScience*, 2016.

P. Liashchynskyi and P. Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for nas, 2019.

M. Lin, Q. Chen, and S. Yan. Network in network, 2013. URL https://arxiv.org/abs/1312.4400.

F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008. doi: 10.1109/ICDM.2008.17.

J. Liu, Y. Deng, Y. Wang, H. Huang, Q. Du, and F. Ren. Urban nighttime leisure space mapping with nighttime light images and poi data. *Remote Sensing*, 12(3), 2020. ISSN 2072-4292. doi: 10.3390/rs12030541. URL https://www.mdpi.com/2072-4292/12/3/541.

Y. Liu, X. Liu, S. Gao, L. Gong, C. Kang, Y. Zhi, G. Chi, and L. Shi. Social sensing: A new approach to understanding our socioeconomic environments. *Annals of the Association of American Geographers*, 105(3):512–530, 2015. doi: 10.1080/00045608.2015.1018773. URL https://doi.org/10.1080/00045608.2015.1018773.

C. Matos, C. A. Teixeira, R. Bento, J. Varajão, and I. Bentes. An exploratory study on the influence of socio-demographic characteristics on water end uses inside buildings. *Science of The Total Environment*, 466-467:467–474, 2014. ISSN 0048-9697. doi: https://doi.org/10.1016/j.scitotenv.2013.07.036. URL https://www.sciencedirect.com/science/article/pii/S0048969713008048.

S. Mohanty, A. Vijay, and S. Deshpande. Understanding urban water consumption using remotely sensed data. In *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, jul 2022. doi: 10.1109/igarss46834.2022.9883890. URL https://doi.org/10.1109%2Figarss46834.2022.9883890.

L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning, 2017a. URL https://arxiv.org/abs/1712.04621.

L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning, 2017b. URL https://arxiv.org/abs/1712.04621.

RegioneLombardia. *Regional content specifications for geotopographic databases*, 2022. https://www.geoportale.regione.lombardia.it/documents/10180/0/Allegato+2_III_Specifiche/19458997-44c0-4d54-b07d-d26ae5d4c6d8 [Accessed: 12/02/2023].

A. H. Reynolds. *Convolutional Neural Networks (CNNs)*, 2019. https://anhreynolds.com/blogs/cnn.html [Accessed: 14/01/2023].

E. Riva and M. Lucchini. La natalità delle imprese straniere a milano: un'analisi spaziale. *Imprese Città*, pages 85–94, 12 2014.

A. Romero, C. Gatta, and G. Camps-Valls. Unsupervised deep feature extraction for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54(3):1349–1362, March 2016. ISSN 1558-0644. doi: 10.1109/TGRS.2015.2478379.

R. E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, page 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

W. Shuster, J. Bonta, H. Thurston, E. Warnemuende, and D. Smith. Impacts of impervious surface on watershed hydrology: A review. *Urban Water Journal - URBAN WATER J*, 2:263–275, 12 2005. doi: 10.1080/15730620500386529.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL https://arxiv.org/abs/1409.1556.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

J. Stańczyk, J. Kajewska-Szkudlarek, P. Lipiński, and P. Rychlikowski. Improving short-term water demand forecasting using evolutionary algorithms. *Scientific Reports*, 12 (1):13522, Aug 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-17177-0. URL https://doi.org/10.1038/s41598-022-17177-0.

Stanford. *Convolutional Neural Networks (CNNs / ConvNets)*, 2022. https://cs231n.github.io/convolutional-networks/#pool [Accessed: 16/01/2023].

E. Star. *Water Use Tracking*, 2012. https://www.energystar.gov/sites/default/files/buildings/tools/DataTrends_Water_20121002.pdf [Accessed: 23/03/2023].

D. Suh, Y.-S. Yoo, I. Lee, and S. Chang. An electricity energy and water consumption model for korean style apartment buildings. pages 1113–1117, 01 2012. ISBN 978-1-4673-2247-8.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the royal statistical society series b-methodological*, 58:267–288, 1996.

E. Toth, C. Bragalli, and M. Neri. Assessing the significance of tourism and climate on residential water demand: Panel-data analysis and non-linear modelling of monthly water consumptions. *Environmental Modelling and Software*, 103, 02 2018. doi: 10.1016/j.envsoft.2018.01.011.

A. C. Townsend and D. A. Bruce. The use of night-time lights satellite imagery as a measure of australia's regional electricity consumption and population distribution. *International Journal of Remote Sensing*, 31(16):4459–4480, 2010. doi: 10.1080/01431160903261005. URL https://doi.org/10.1080/01431160903261005.

L. Wang, H. Fan, and Y. Wang. Improving population mapping using luojia 1-01 nighttime light image and location-based social media data. *Science of The Total Environment*, 730:139148, 2020. ISSN 0048-9697. doi: https://doi.org/10.1016/j.scitotenv.2020.139148. URL https://www.sciencedirect.com/science/article/pii/S0048969720326656.

N. A. Wardrop, W. C. Jochem, T. J. Bird, H. R. Chamberlain, D. Clarke, D. Kerr, L. Bengtsson, S. Juran, V. Seaman, and A. J. Tatem. Spatially disaggregated population estimates in the absence of national population and housing census data. *Proceedings of the National Academy of Sciences*, 115(14):3529–3537, 2018. doi: 10.1073/pnas.1715305115. URL https://www.pnas.org/doi/abs/10.1073/pnas.1715305115.

G. Warth, A. Braun, O. Assmann, K. Fleckenstein, and V. Hochschild. Prediction of socio-economic indicators for urban planning using vhr satellite imagery and spatial

analysis. *Remote Sensing*, 12(11), 2020. ISSN 2072-4292. doi: 10.3390/rs12111730. URL https://www.mdpi.com/2072-4292/12/11/1730.

X. Xing, Z. Huang, X. Cheng, D. Zhu, C. Kang, F. Zhang, and Y. Liu. Mapping human activity volumes through remote sensing imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:5652–5668, 2020. doi: 10.1109/JSTARS.2020.3023730.

J. Yang, L. Zhao, J. Mcbride, and P. Gong. Can you see green? assessing the visibility of urban forests in cities. *Landscape and Urban Planning*, 91(2):97–104, 2009. ISSN 0169-2046. doi: https://doi.org/10.1016/j.landurbplan.2008.12.004. URL https://www.sciencedirect.com/science/article/pii/S0169204608002314.

C. Yeh, A. Perez, A. Driscoll, G. Azzari, Z. Tang, D. Lobell, S. Ermon, and M. Burke. Using publicly available satellite imagery and deep learning to understand economic well-being in africa. *Nature Communications*, May 2020.

B. Yu, T. Lian, Y. Huang, S. Yao, X. Ye, Z. Chen, C. Yang, and J. Wu. Integration of nighttime light remote sensing images and taxi gps tracking data for population surface enhancement. *International Journal of Geographical Information Science*, 33 (4):687–706, 2019. doi: 10.1080/13658816.2018.1555642. URL https://doi.org/10.1080/13658816.2018.1555642.

C. Zeng, Y. Zhou, S. Wang, F. Yan, and Q. Zhao. Population spatialization in china based on night-time imagery and land use data. *International Journal of Remote Sensing*, 32(24):9599–9620, 2011. doi: 10.1080/01431161.2011.569581. URL https://doi.org/10.1080/01431161.2011.569581.

F. Zhang, D. Zhu, L. Wu, and Y. Liu. Social sensing from street-level imagery: A case study in learning spatio-temporal urban mobility patterns. *ISPRS Journal of Photogrammetry and Remote Sensing*, 153:48–58, 05 2019. doi: 10.1016/j.isprsjprs.2019.04.017.

H. Zhang. The optimality of naive bayes. volume 2, 01 2004.

Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. pages 89–98, 09 2011. doi: 10.1145/2030112.2030126.

B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

A | Data profiling

Before starting to extract the daily water consumption value from the corresponding time series, an analysis of the data was carried out to understand what relationships existed between the consumption patterns, i.e. whether buildings with similar physical characteristics had similar time series or to show unique insights regarding the factors influencing the water demand.

A.1. Data driven approach

54 water consumption time series of 5-story buildings were grouped for analysis. For the first step, missing values and outliers present in the data were replaced using the method already described in the data chapter 4. The correlation matrix of the time series was calculated: it is a table that shows the *Pearson* correlation coefficients between pairs of variables in the dataset. Each cell in the table represents the correlation coefficient between two variables, with values ranging from -1 to 1. A positive correlation coefficient indicates a positive relationship between two variables, meaning that when one variable increases, the other also tends to increase. A negative correlation coefficient indicates a negative relationship between two variables, meaning that when one variable increases, the other tends to decrease. A correlation coefficient of 0 indicates no linear relationship between the two variables. Figure A.2 shows the matrix composed by the first 25 time series of the 5-story building set analyzed, each one associated with a specific PDR number.

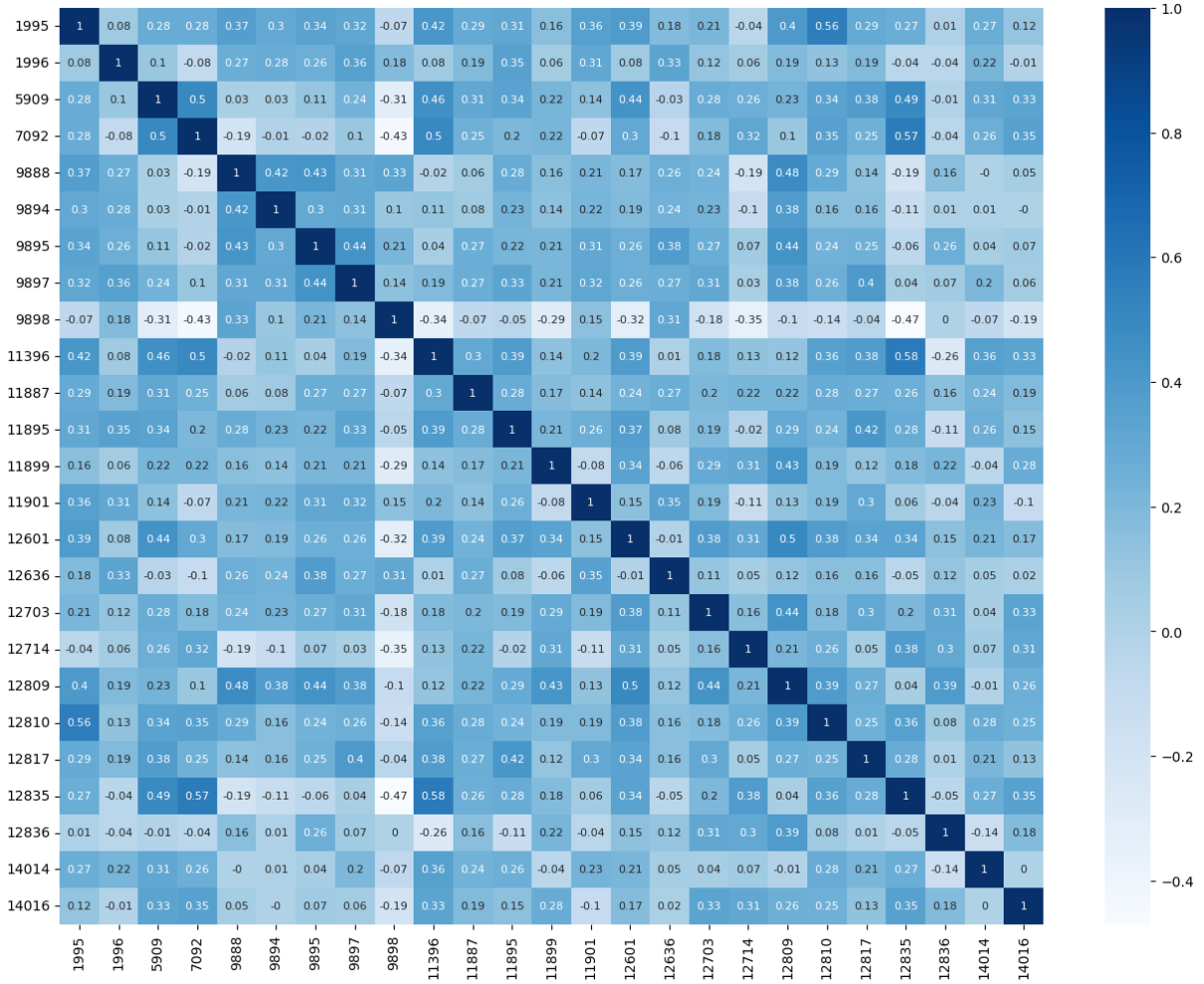


Figure A.1: Correlation matrix of the first 30 time series of the 5-story building set.

As can be seen, the time series are weakly related to each other even though the buildings have similar heights and areas. For that reason, instead of directly analyzing the time series, the focus of the analysis has been carried on their seasonality. The additive decomposition assumes that the time series is composed of the sum of three components: residuals, trend and seasonality. Seasonality refers to the pattern of regular and periodic fluctuations in a time series that occur at fixed intervals of time, typically, for water consumption, over the week. Figure A.2 highlights the relevant correlations that exist between the weekly periodicity of water usage of different buildings.

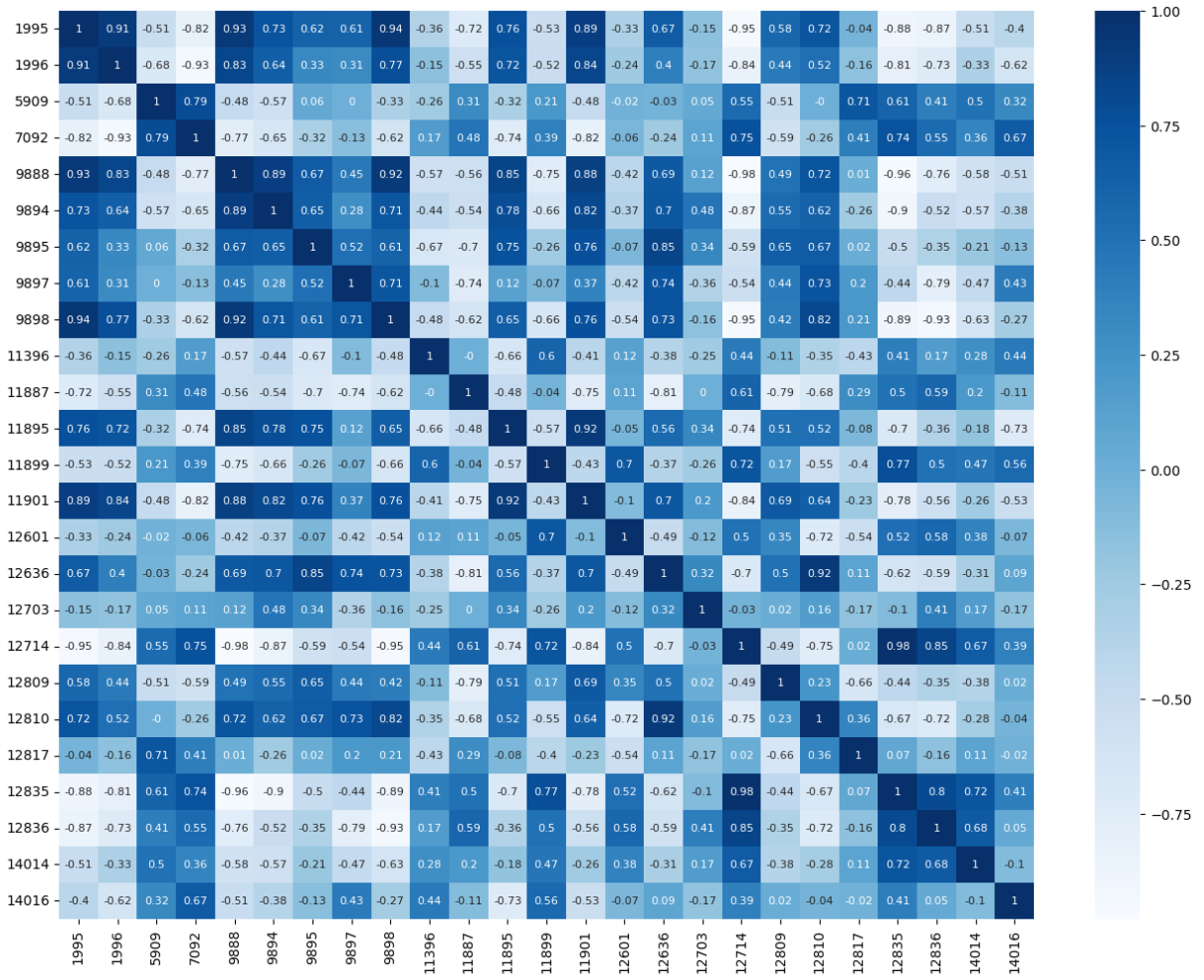


Figure A.2: Correlation matrix of the first 30 time series seasonality of the 5-storey building set.

In order to identify patterns of water usage that are common across multiple buildings, which can be used to gain insights into factors that influence water consumption, the seasonality of the time series was clustered.

A.1.1. SOM: Self-Organizing maps

Self-Organizing Maps (SOM) (Kohonen, 1990) is a type of unsupervised neural network algorithm that can be used for clustering and visualization of high-dimensional data. The algorithm uses a two-dimensional grid of nodes, where each node corresponds to a weight vector with the same dimensionality as the input data. The nodes are organized in such a way that neighboring nodes in the grid correspond to similar weight vectors. To cluster time series data using SOM, each time series is first converted into a feature vector using a suitable feature extraction method. The feature vectors are then presented to the SOM

algorithm, which maps them to a two-dimensional grid of nodes. The SOM algorithm uses a competitive learning approach to adjust the weights of the nodes to best match the input feature vectors. Once the SOM algorithm has been trained, the nodes in the grid can be used to cluster the input time series data. Time series that are mapped to neighboring nodes in the grid are likely to be similar to each other and can be grouped together into clusters.

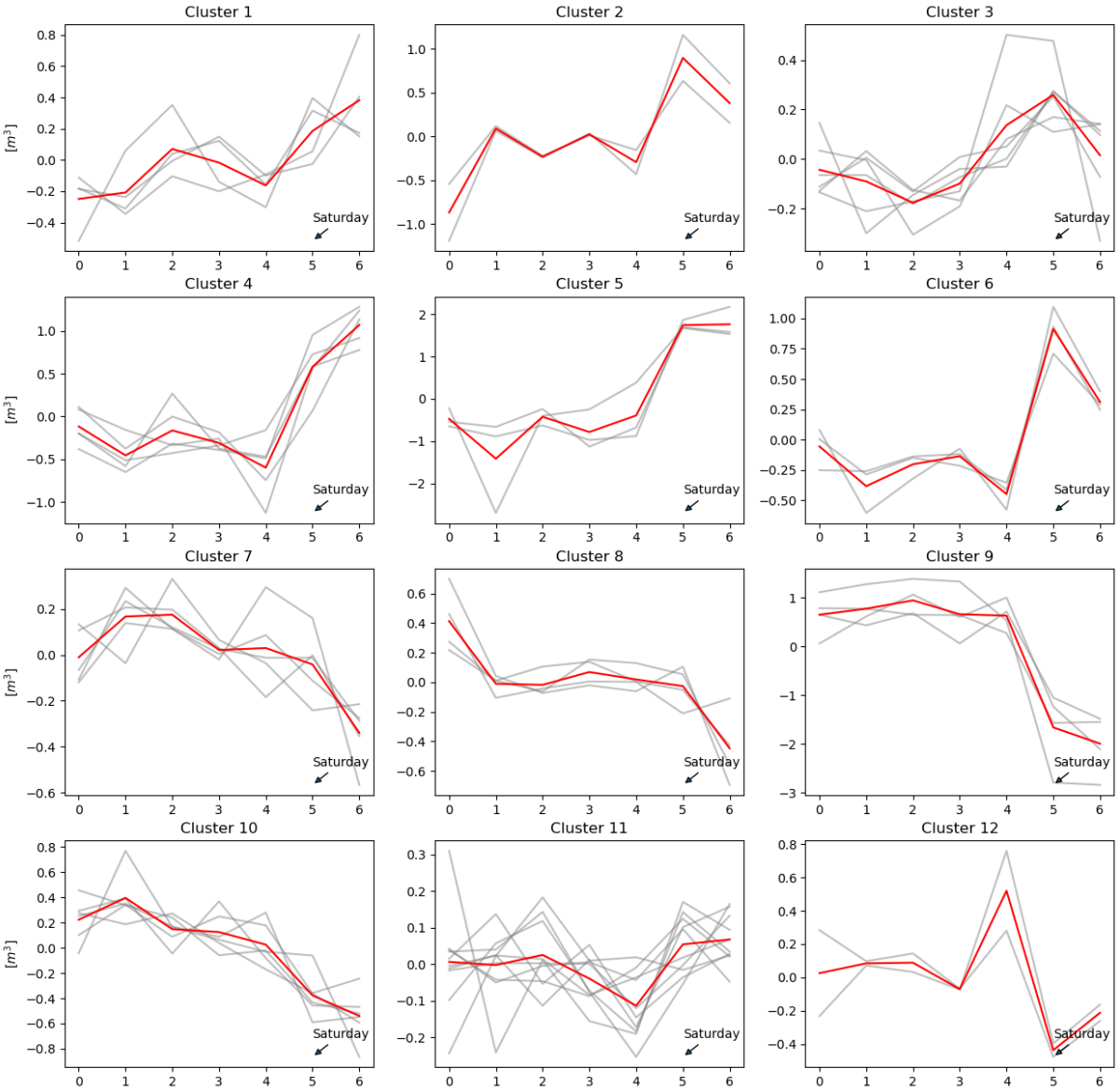


Figure A.3: SOM clusters of water time series seasonality.

The Figure A.3 represents the 12 clusters identified by the SOM algorithm; the water demand fluctuations have a weekly periodicity: 0 on the x-axis encodes Monday while 6 encodes Sunday. The different behaviours regarding water usage on weekends and during

the working weekday may depend on several factors, including the type of building, the location, the weather, and the demographic characteristics of the population.

A.2. Commercial activities influence on building water demand

During the study, it was hypothesized that one of the influencing factors could be the presence of commercial activities(e.g.,shops) at the base of the building. Commercial activities can potentially have an impact on the water usage of the entire structure. The impact will depend on the type of activity and the purpose carried out within it. For instance, if the shop is a grocery store or a restaurant, it may require significant amounts of water for cleaning and food preparation. This can result in higher water usage within the building, particularly during business hours and on weekends. To confirm the hypothesis, out of the 54 5-story buildings, 20 with a store at the base were identified. The weekly periodicity of water usage fluctuations was extracted and the *boxplots* involving all the data for each weekday was calculated. A boxplot, also known as a box and whisker plot, is a graphical representation of the distribution of a dataset. It displays the median, quartiles, and outliers of a set of values.

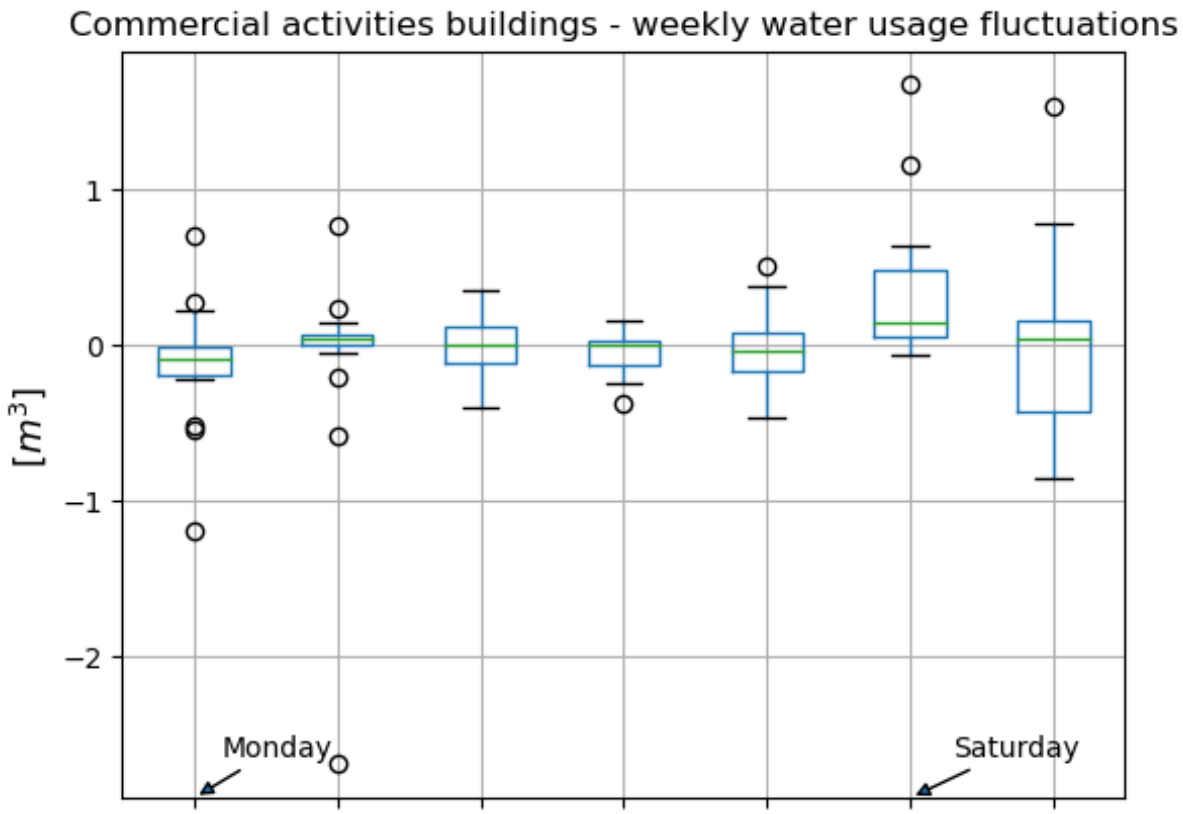


Figure A.4: Weekly water consumption fluctuations in building having a commercial activity.

As can be seen from the Figure A.4, the buildings with a commercial activity consume more water during the weekend, specifically on Saturdays. To complete the analysis, the water usage fluctuations during the week in buildings without a shop were also calculated and expressed in the form of boxplots (Figure A.5).

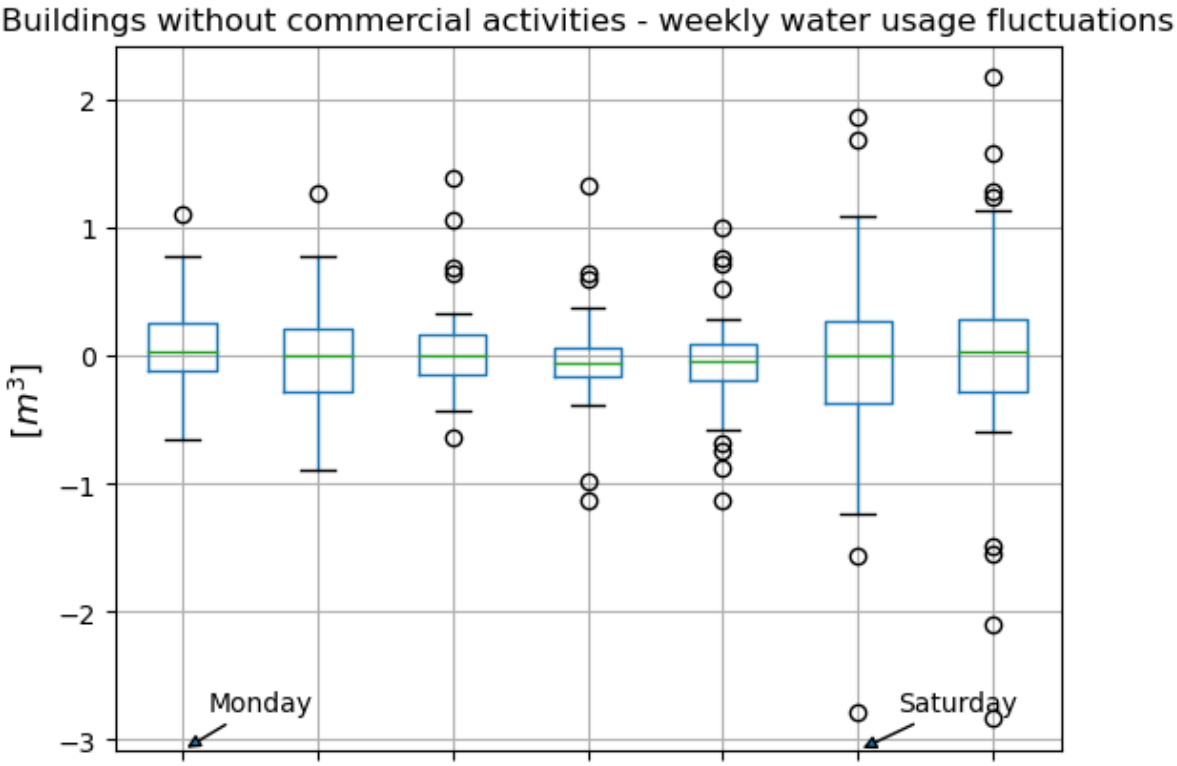


Figure A.5: Weekly water consumption fluctuations in building not having a commercial activity.

The difference is significant, especially on Saturdays and on the first days of the week, where residential consumption in buildings without commercial activities is slightly higher.

List of Figures

List of Tables

Acknowledgements

Ringrazio il progetto ide3a che, tramite finanziamenti dell DAAD (Servizio Tedesco per lo Scambio Accademico), mi ha offerto una borsa di studio per un periodo di ricerca di 3 mesi alla Technische Universität Berlin. Il supporto di questa borsa mi ha permesso non solo di sviluppare il lavoro di tesi all'estero ma di vivere una delle esperienze più significative della mia vita. Sarò sempre grato per l'ospitalità riservatami da tutto il gruppo.

Ringrazio MM S.p.A e il Geoportale del comune di Milano per i dati offerti fondamentali ai fini della realizzazione della tesi.

Ringrazio il Prof. Andrea Cominola che mi ha guidato e coinvolto durante tutto il lavoro di tesi e il periodo di studi a Berlino con preziosi consigli non solo dal punto di vista tecnico ma anche umano.

Ringrazio il Prof. Andrea Castelletti per i continui confronti e le interessanti suggestioni offerte per condurre il lavoro di ricerca al meglio.

Ringrazio Wenjin and Siling, la possibiltà di confronto e il vostro continuo supporto anche nelle fasi più complesse mi ha reso più sereno sapendo che c'erano due persone come voi a correggere i miei sbagli.

Un grazie a tutti quelli che ci sono stati durante questi anni, non solo alla mia famglia e gli amici ma anche a quelle persone che indirettamente mi hanno ispirato o dato un motivo per non arrendermi. Vi voglio bene e vi auguro ogni successo.

A mia madre, la persona più forte che conosco. Il tuo sorriso ha sempre illuminato anche il più cupo dei miei giorni, grazie per avermi insegnato il valore della libertà, del viaggio e della lettura. Grazie per rendermi felice ogni giorno.

A mio padre che mi ha insegnato a guardare entrambi i lati della medaglia di ogni aspetto. Grazie per il continuo aiuto nei momenti di difficoltà e per avermi mostrato il valore della vita.

A mia sorella, una personalità coraggiosa e dinamica ma altrettanto sensibile e giocosa. Nel viaggio verso i tuoi sogni, ti fermi sempre a sostenermi con mille parole o anche con

un semplice sguardo.

A nonna e nonno, custodi dei miei primi ricordi. Quando le radici sono più profonde non c'è motivo di temere il vento. Grazie per l'affetto che mi avete dato e continuate a dare.

A Martina, il mio piccolo grande amore. Come ben sai, le parole non sono il mio forte ed è per questo che mi avvarrò di un film che ti piace tanto: l'amore più bello è quello che risveglia l'anima e ci fa desiderare di arrivare più in alto, è quello che incendia il nostro cuore e porta la pace nella nostra mente, questo è quello che tu mi hai dato e che spero di darti per sempre. Starò sempre al tuo fianco, grazie ancora per avermi insegnato ad amare.

A Marco. Siamo ciò che pensiamo, ma a definirci sono anche le persone di cui ci circondiamo. Tutte le pagine della tesi non basterebbero a descrivere le esperienze condivise e cos'altro faremo. Se ho intrapreso i bivi che mi hanno portato a raggiunger questo obiettivo è in parte merito tuo. Ricerco sempre la felicità e spensieratezza di quando eravamo bambini all'elementari e, quando sono insieme a te, le avverto come se il tempo non fosse passato. Ti voglio tanto bene e ti auguro di raggiungere i tuoi sogni. Stai sicurò che sarò li con te, pronto a festeggiare.

Ai miei amici di infanzia e adolescenza. Chi l'avrebbe mai detto che da quel giorno in cui ho conosciuto ognuno di voi saremmo arrivati fino a qui. Ognuno con i suoi sogni abbiamo intrapreso vie differenti ma è sempre bello incontrarci ai punti di sosta. Quando ci riuniamo i ricordi vengono a galla ricordandomi da dove sono partito e, forse, dove sono diretto.

Ai miei amici dell'Unisa. Se oggi ho raggiunto questo traguardo è anche per le esperienze affrontate insieme a voi durante quegli anni memorabili all'inizio del nostro percorso. Non sapete ancora quanto valete e vi auguro di scoprirlo raggiungendo i vostri obiettivi ma, citando un nostro professore, ricordatevi sempre: pancia a terra.

Ai miei amici del Politecnico, compagni di faticosi progetti ed esami. Ricordo ancora i nostri sguardi stanchi dopo lunghe ore di programmazione ma anche la soddisfazione nei nostri occhi a compito svolto. Questo percorso di studi è stato impegnativo ma se è questo il prezzo da pagare per aver conosciuto persone come voi allora sono stato ben lieto di aver saldato il conto.

Ai miei amici conosciuti Berlino. Si dice che il viaggiare cambi l'uomo, io sicuramente non ho esplorato il mondo così tanto ma l'avervi conosciuto e l'aver parlato con voi mi ha arricchito come se avessi visitato i vostri luoghi di origine. Dalla Costa Rica all'Olanda, dal Belgio alla Grecia, grazie per avermi fatto sentire un cittadino del mondo ed espanso

i miei orizzonti. Spero, un giorno, di potervi rincontrare e ridere davanti ad una birra tedesca dei ricordi passati.