Data Mining for Smart Cities: Energy prediction for public buildings

Vasiliki Papanikolaou

SID: 3308180016

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

JANUARY 2021

THESSALONIKI – GREECE

Data Mining for Smart Cities: Energy prediction for public buildings

Vasiliki Papanikolaou

SID: 3308180016

Supervisor:	Prof. Christos Tjortjis
Supervising Committee Members:	Prof. Georgios Tsirigotis
	Prof. Panagiotis Bozanis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

JANUARY 2021

THESSALONIKI – GREECE

Acknowledgements

I would like to take this opportunity to express my gratitude to the following people who helped me during this project. First, I would like to thank my supervisor Professor Christos Tjortjis for his support and guidance. Also, I would like to thank the members of the Data Mining and Analytics laboratory of International Hellenic University who helped me to come up with new ideas and visualize my research. Finally, I would wish to thank my husband, my family and friends. This work would not have been completed without their valuable help, continuous support and interest.

Vasiliki Papanikolaou

January 4th, 2021

Abstract

This dissertation was a part of the program of MSc in Data Science in the International Hellenic University. The scope of the study is to use Big Data and data mining methods in the prediction of the energy consumption loads of non-commercial buildings in Smart Cities. This task was achieved through 9 cases of deployed models. The load prediction was made through Random Forest, Gradient Boosting, Linear and Extreme Gradient Boosting Regression models in 4 cases of the first stage and 3 cases on the second stage. Hyper-parameter tuning and model optimization through k-Fold Cross Validation and GridSearch CV methods took place in the second case scenarios. The results achieved for load prediction were 85.65% for the first case and 94.38% for the second case. For all the evaluations a dataset of 4.3 million datapoints was utilized, as part of the Building Data Genome Project database. For the building type prediction using load data, the Gradient Boosting and the Random Forest Classification methods were used. The score achieved for this case was 90.83% with some preprocess of the data and no parameter tuning.

Contents

ACKNOWLEDGEMENTS ..... **III**

ABSTRACT ..... **IV**

CONTENTS ..... **V**

LIST OF TABLES ..... **VII**

LIST OF FIGURES ..... **VIII**

1 INTRODUCTION **1**

2 THEORETICAL BACKGROUND **5**

2.1 SMART CITIES 5

2.1.1 Definitions and dimensions *5*

2.2 DATA MINING 8

2.2.1 Basic concepts *8*

2.2.2 Supervised learning *9*

2.3 BIG DATA 10

3 RELATED WORK **13**

3.1 BUILDING DATA GENOME PROJECT 13

4 METHODOLOGY **17**

4.1 PROBLEM 17

4.2 DATASET DESCRIPTION 18

4.3 MACHINE LEARNING MODELS 19

4.3.1 Random Forest Classification and Regression *20*

4.3.2 Gradient Boosting Classification and Regression *20*

4.3.3 Linear Regression *20*

4.3.4 Extreme Gradient Boosting Regression *20*

4.4 OPTIMIZATION 21

4.4.1 Hyper-parameters *21*

4.4.2 Optimization techniques *21*

4.5 EVALUATION AND METRICS 22
4.5.1 Regression evaluation and metrics *22*
4.5.2 Classification evaluation and metrics *23*

5 RESULTS 27

5.1 DATA DESCRIPTION AND PREPROCESSING 27
5.1.1 Timeseries data *27*
5.1.2 Data preprocess *30*
5.2 LOAD PREDICTION 32
5.2.1 Preprocess of data *33*
5.2.2 Case analysis *33*
5.3 PREDICTION OF THE BUILDING TYPE 36
5.3.1 Preprocess of data *36*
5.3.2 Case analysis *37*

6 DISCUSSION 41

7 CONCLUSION 45

7.1 STUDY REVIEW 45
7.2 THREATS ON VALIDITY 46
7.3 FUTURE WORK 46

BIBLIOGRAPHY 49

APPENDIX A 53

APPENDIX B 57

APPENDIX C 58

List of Tables

Table 1. The key categories and primary indicators of a smart city according to ITU-T FG-SSC analysis. 7
Table 2 Building energy consumption datasets 27
Table 3 Summary of the evaluation metrics of Case 1 models 34
Table 4 Summary of the evaluation results of Case 2 models 35
Table 5 Classes and number of tuples per class 37
Table 6 Mean accuracy of the classification models 38

List of Figures

Figure 1 The most important key words with reference to Smart Cities which were derived from the ITU – T FG- SSC. 7
Figure 2 The most widely used methods of machine learning per category. 10
Figure 3 The five main characteristics of big data. 11
Figure 4 Distribution of case study buildings among time zone, industry, sub-industry and primary use type. 19
Figure 5 MSE calculation formula 22
Figure 6 RMSE calculation formula. 23
Figure 7 R^2 calculation expression. 23
Figure 8 Confusion matrix. 24
Figure 9 Definition formula of accuracy 24
Figure 10 Definition formula of sensitivity 25
Figure 11 Definition formula of specificity 25
Figure 12 The accuracy formula expressed in terms of sensitivity and specificity. 25
Figure 13 Definition formula of precision. 25
Figure 14 Definition formula of recall. 26
Figure 15 Definition formula of F1 or F-score. 26
Figure 16 Energy consumption profiles in kWh, for 507 buildings for the period 01-01-2010 up to 01-01-2016 28
Figure 17 Hourly energy consumption values in kWh, of Primary Classroom Everett for a year 28
Figure 18 Hourly energy consumption values in kWh, for University Classroom Caitlyn for a year 29
Figure 19 Hourly energy consumption values in kWh, for University Dormitory Una for a year 29
Figure 20 Hourly energy consumption values in kWh, for University Laboratory Paul for a year 29
Figure 21 Hourly energy consumption values in kWh, for Office Elizabeth for a year 29
Figure 22 Dataset instance after the timeseries transformation. 31

Figure 23 Instance of the merged dataframe of temporal and meta-data. 32
Figure 25 Random Forest, classification report. 39
Figure 24 Gradient Boosting, classification report 39
Figure 26 Confusion matrix of the Gradient Boosting classifier 57
Figure 27 Confusion matrix of the Random Forest classifier 57

1 Introduction

Today, we are surrounded by smart cities. It is on the forefront of the media and on the imminent plans of technology companies and entrepreneurs. Societies and local governments have significantly increased their interest in them, through the last 10 years. Nevertheless, what is the actual interest in them, beyond the technological advancements that they showcase? Through new technologies, efficiency and interconnectivity, numerous cities around the world strive to provide the best quality of life for their citizens.

Interdisciplinary studies and various stakeholders investigate the smart city and view this topic from different perspectives. The University of Oxford and the Oxford Institute of Internet, describe the idea of the "smart city" as the act of "giving the policymakers real-time information on a whole variety of indicators about their city (traffic, environment, services, etc.) in order to improve decision making and optimize service delivery"[1]. A great institution such as the European Union, defines the smart city as "… a place where traditional networks and services are made more efficient with the use of digital and telecommunication technologies for the benefit of its inhabitants and business". However, this meaning is further expanded; beyond the information and communications technology (ICT) infrastructure, it includes not only the transport networks, the water and waste disposal facilities and the efficient ways of lighting and heating the buildings, but among others, also, the response of the city administration, the public space and the needs of the population [2].

Among the many stakeholders that shape and create the future of smart cities, governments are some of the most important. City leaders, such as mayors and local authorities, through their policies define guidelines and implement plans which transform a traditional urban environment into a smart city. Yet still, there is not a complete, universal recipe of what a government should do to make its cities smart. This is highly dependent on the unique context of each city. The engagement of the general government results in a combination of different political and technical roles, well beyond simply the implementation of the latest technology advancements. But this is a two-way relationship. Smart city initiatives provide opportunities to city authorities for long-term or immediate cost-savings. In this era of austerity measures and cuts on spending, the goal

for governments is to save the public resources and innovate, both of which are at the heart of smart cities.

The private domain is of equal importance. Enterprises and corporate organizations contribute greatly to the development of smart cities and benefit greatly, as well. The entrepreneurs are the pioneers who initiate many of the transitions during the development process of a smart city. In return, the technologies being adopted by a city produce large amounts of data, which boost business opportunities [3]. New jobs, new establishments, sustainability and efficiency in corporate operations and in the services provided are only a part of many benefits for the business sector [4].

The urban infrastructures play a crucial role in the quality of the services within a smart city, but also on their complexity, vulnerability or high cost. Therefore, specific developments are required, such as smart monitoring of systems and services, advanced communication technology which guarantees the safety and the integrity of the data transmission, advanced data analysis and machine learning tools for data processing and integrated platforms for the management of smart infrastructures [5]. Moreover, it is the potential interconnectivity between these infrastructures that enables the smart city applications to function in the most effective and useful way. Now, more than ever, the Internet of Things (IoT) with its increasing capabilities, qualifies for this. IoT being embedded and omnipresent adds to the integration of the real world to a network and eventually the Smart World.

Most smart city applications operate through IoT. This leads to large amounts of data, the so-called Big Data. The main characteristics of big data are the large volume, the increased velocity and the wide variety, while they are collected through various resources. Their analysis offers the city valuable insights; hence, they play a key role in transforming the lives of the citizens [6].

The large amounts of data generated continuously within the boundaries of a smart city, require at first, efficient data storage and then, effective processing methods, to be utilized in the decision-making process. The use of Data Mining (DM), Machine Learning (ML) and data analytics techniques provide a lot of tools that serve this cause [7]. Specifically, DM and ML techniques filter, analyze, process the data, and eventually extract high-quality information. Even so, the effectiveness or even suitability of traditional data mining methods and analytics platforms are sometimes challenged. Then, more ad-

vanced techniques, such as Deep Learning (DL) and Reinforcement Learning (RL) could sometimes be employed to accomplish the demanding task [8].

In this study, a part of great significance for smart cities will be assessed. The building sector, which is responsible globally for an estimated 40% of total energy consumption. In particular, this study focuses on non-residential buildings due to their unique characteristics and complexities in energy consuming systems. Publicly open and available data which consist of thousands of timeseries, are preprocessed and analyzed via data mining methods. The aim of this analysis is to describe and provide the energy consumption patterns of the selected public municipal buildings. Finally, several predictive models are employed and explored to provide accurate predictions of the energy consumption for certain types of public buildings. To achieve this goal, the research objectives for this study are the following:

- To deploy several simple and well-known data mining predictive models and evaluate their accuracy in load forecasting for certain types of public buildings.
- To further explore and evaluate these models by performing boosting techniques.
- To predict and classify a building type by using load as an attribute.

The dataset is a result of a project of Clayton Miller and Forrest Meggers and of the ETH University of Zurich to create an open dataset for non-residential buildings [9]. It consists of electrical meter data of 507 commercial buildings with at least 8760 timestamps for each one of them. The dataset is public, open and available online by the researchers, with the purpose of a repository creation suitable for benchmarking predictive models and help scholars in energy research.

The analysis of the data and the experiments on this study are executed in Python 3.7. Specifically, the packages Pandas, Matplotlib, are mainly used for the analysis and visualizations and Scikit-Learn is used for the implementation of the machine learning algorithms. For the deployment of the models Google Colab was used and Jupyter Notebooks. All the computational work was executed and processed on the Google Cloud.

The remainder of the study is structured as follows. Chapter 2 describes and sets the theoretical background for Smart Cities, Data Mining and Big Data. In Chapter 3, a literature review focused on the Building Data Genome dataset which was used is presented. Chapter 4 describes and analyses the methodology used to explore and prepro-

cess the data. Also, the theoretical description of the models that were used is illustrated along with the evaluation metrics and the hyperparameters that were tuned. In chapter 5, the experimental results for data preprocessing and the results of the models deploy are outlined. All the load forecasting scenarios and the building type prediction scenarios are presented here. Finally, the last two chapters 6 and 7 discuss and summarize the findings of the study, accordingly. Aspects regarding the validity of this work and future research directions are briefly explained in the last chapter.

2 Theoretical background

The starting point of a study is to define the subject. In this case, it is the question: What is a smart city? Nevertheless, any attempt to give a solid answer to that question results in ambiguities and finally a series of context-based definitions. As it seems the relationship between smart cities and data mining methods evolves through the years, so, that today, no further development of smart cities research can be made far from the foundational data mining, analytics and machine learning methods. Moreover, the data which are related to smart cities projects, are inevitably analyzed and explored through big data related processes.

2.1 Smart Cities

2.1.1 Definitions and dimensions

Identifying an operational definition for smart cities requires a closer examination of the theoretical background. Most of the researchers who study the concepts of smart cities, find common ground on the fact that there is not only one widely accepted definition of the term ‘smart city’ in bibliography. This is apparent to the scholars and in either case clearly stated by many of them in their works [10]–[14]. According to researchers [15], ‘smart city’ as a notion appeared for the first time in 1998 by Van Bastelaer. However, Dameri and Cocchia in 2013 stated that the concept was introduced in 1994 [13]. A study of relevant research showed that, the definition of this term is an ongoing process. As ‘smart city’ is directly linked to the evolution of technology and newer sources reflect that.

At one of the very first attempts to provide a definition for ‘digital city’, Van Bastelaer concludes that the term may have several definitions, some of which differ greatly from one another. However, in this study, the rapidly growing information and communication technologies are proven to be the driving forces that transformed the advanced industrial cities. The online services are managed by the municipal authorities or citizens and present useful information that make people’s lives easier in the city [16]. From that early notion of the digital city, the terminology has changed and during the last decade

researchers explored alternative approaches of the former ‘digital city’ which is now a ‘smart city’. Cocchia in her work [10] describes the smart city appearance as a sequent that came after the emerging urbanization phenomenon in this century. Elaborating on this idea, this trend flourished due to the information and technology improvements. At most times, either ICT attributes of the city are highlighted (i.e digital, broadband, wireless, etc.) or the information flow through the urban space is [15]. Others argue that ranking a characteristic (i.e. digital infrastructure) of a smart city higher than another, only gives a unilateral perspective to the meaning of a smart city. Instead, multiple issues should be considered such as: awareness, flexibility, transformability, synergy, individuality, strategic behavior, self-decisive [17]. In their attempt to shed light on this fuzzy concept, the authors T. Nam and T. A. Pardo approach the term “smart” from a linguistic, a marketing, a technological and an urban planning field perspective in their work [18]. It is further inferred that a smart city can be called by many names (Digital City, Intelligent City, Information City, Knowledge City or Smart Community) and all of them are equally effective. It is mostly the modern technology and marketing stresses that have resulted in the prevalence of the term ‘smart city’.

Giffinger, who is a widely cited researcher of this topic, in his work in 2007 [17] builds upon the ‘smartness’ of a city and gives the following definition: “a city well performing in a forward-looking way in Smart Economy, Smart People, Smart Governance, Smart Mobility, Smart Environment and Smart Living, built on the ‘smart’ combination of endowments and activities of self-decisive, independent and aware citizens”. This is a holistic approach, which recognizes that various aspects should form the basis for a complete definition. In the same direction, the International Telecommunications Union (ITU) in 2015, following an assessment of 116 available definitions on smart cities, agreed on this definition: *“*A smart sustainable city is an innovative city that uses information and communication technologies (ICTs) and other means to improve quality of life, efficiency of urban operation and services, and competitiveness, while ensuring that it meets the needs of present and future generations with respect to economic, social, environmental as well as cultural aspects”. [19] The ITU-T Focus Group on Smart Sustainable Cities (FG -SSC) identified a total of 8 different key categories, 6 primary indicators and 30 key words as representative of a smart city. The following Table 1 summarizes the findings of the categories and indicators, while on Figure 1 the word cloud reflects a quantitative analysis of the different keywords and the number of occurrences that these keywords have from the 116 documents studied [20].

Table 1 The key categories and primary indicators of a smart city according to ITU-T FG-SSC analysis. Source [20]

Key categories	Primary Indicators
Quality of life and lifestyle	Smart living
Infrastructure and services	Smart people
ICT, communications, intelligence and information	Smart environment and sustainability
People, citizen and society	Smart governance
Environment and sustainability	Smart mobility
Governance, management and administration	Smart economy
Economy and finance	
Mobility	

Figure 1 The most important keywords with reference to Smart Cities which were derived from the ITU – T FG- SSC. Source: [20]

Although researchers and academics recognize the great importance of intellectual and social capital, for example smart governance and smart strategic planning, [14], [17] the private corporate sector acts differently. In organizations and companies, it is anticipated that ICT provides a means of improving productivity through automation of processes and enhance decision making, planning and control activities. In a city, it is the same contribution that ICTs have in simplifying the urban living complexities. The abun-

dance of data and the active use of them in decisions can make the difference in a city's traffic or energy problem [21].

2.2 Data Mining

Data mining is a subject that falls into many disciplines and can be defined in many ways. The basic concepts of data mining are similar to the core concepts of the Knowledge Discovery from Data (KDD) field. In general, the KDD process is a sequence of the following steps:

i. Data cleaning, removal of noise and inconsistencies in data
ii. Data integration, combination of multiple sources of data
iii. Data selection, retrieval of data relevant to the analysis
iv. Data transformation, transformation processes apply to data such as aggregations or summaries
v. Data mining, application of algorithmic methods to extract patterns from data
vi. Pattern evaluation, identification of interesting patterns in data
vii. Knowledge presentation, visualization and representation techniques are utilized to showcase the extracted knowledge to the users [22].

2.2.1 Basic concepts

Some very basic concepts about data mining consider the foundational blocks of it. A very basic concept in data mining is the idea of Class and databases. All the entries in a database can be categorized in a number of classes. A class could be defined as a concept which is described and characterized by specific data [22] Although, many kinds of data can be mined, e.g. data streams, sequences of data, graphs etc., the most usual are databases or data warehouses. The databases can hold several thousand or millions of organized records.

The Data Mining techniques can be divided in two categories: descriptive or predictive. The descriptive methods provide information which are properties of the data. The predictive methods extract inferences from the data to predict the information which are not evident. The different techniques that fall into these two categories are:

i. Data classification
ii. Data prediction (regression methods)
iii. Data clustering

iv. Outliers analysis
v. Association rule mining [23]

In the present work the first two categories are of interest for the analysis purposes. Data classification is the process of classifying the available dataset in classes. This category incorporates methods that are called supervised learning. These methods exploit a set of given input – output to learn a function that maps the input to output. The other techniques available at the data classification category, are unsupervised learning, where no outputs are provided, and reinforcement learning methods, where the output is a set of instructions rather than a class.

2.2.2 Supervised learning

The supervised learning methods are the most used in machine learning applications. The general idea is that a set of labelled data is provided and the data mining technique learns from the relationship between the inputs and the outputs the function that maps from the input data to the output data. The closer the approximation of this relationship the better the performance of the method.

A further categorization in supervised learning methods, divides them in classification and regression:

i. Classification: A classification problem is when the output variable is a category
ii. Regression: A regression problem is when the output variable is a real value [24]

The following figure shows graphically some of the most common machine learning methods, Figure 2 The most widely used methods of machine learning per category. Source [25]

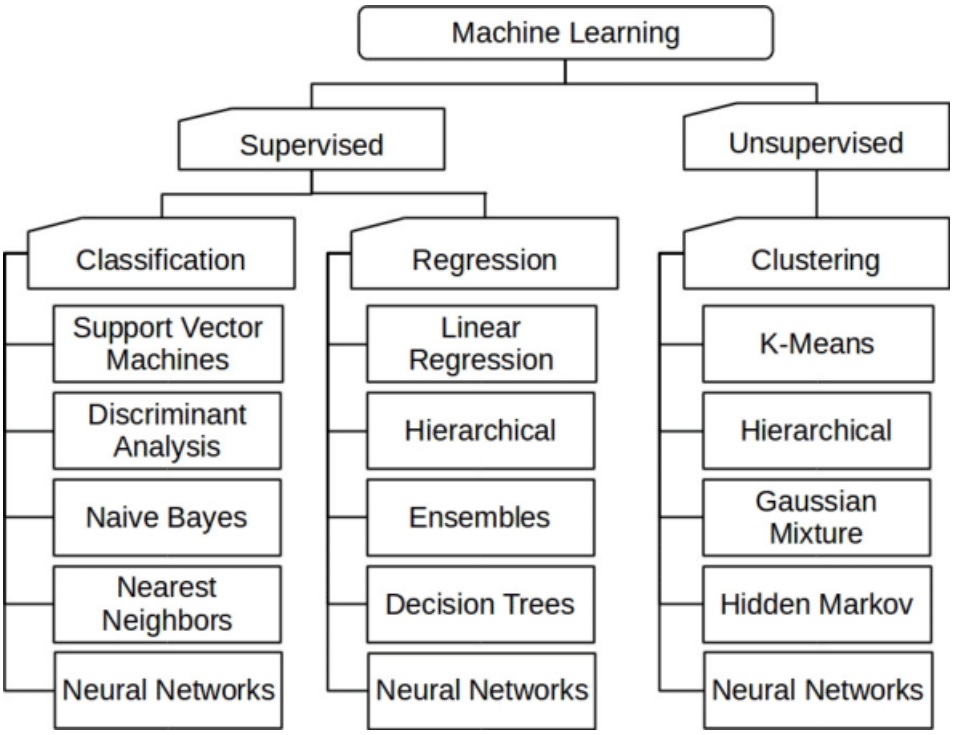

Figure 2 The most widely used methods of machine learning per category. Source [25]

2.3 Big Data

One definition of Big Data comes from the McKinsey Global report from 2011: " Big Data is data whose scale, distribution, diversity, and/ or timeliness require the use of new technical architectures and analytics to enable insights that unlock new sources of business value."

Although the volume of Big Data tends to attract the most attention, generally the variety and velocity of the data provide a more precise definition of Big Data. Some years earlier big data used to be described by three main characteristics:

i. Huge volume of data, not thousands or millions of rows, but billions of rows and millions of columns.
ii. Complexity of data types and structures, big data reflect the variety of available data sources, formats, and structures.
iii. Speed of new data creation and growth, they can describe high velocity data, with rapid data ingestion and near real time analysis.

This is no longer the case. Big data is more than these three aspects. A new revised 5 point definition tries to incorporate all the characteristics than define big data. But today, this is surpassed as well. In Figure 3 The five main characteristics of big data. Source: [26]the 5 aspects of big data are illustrated.

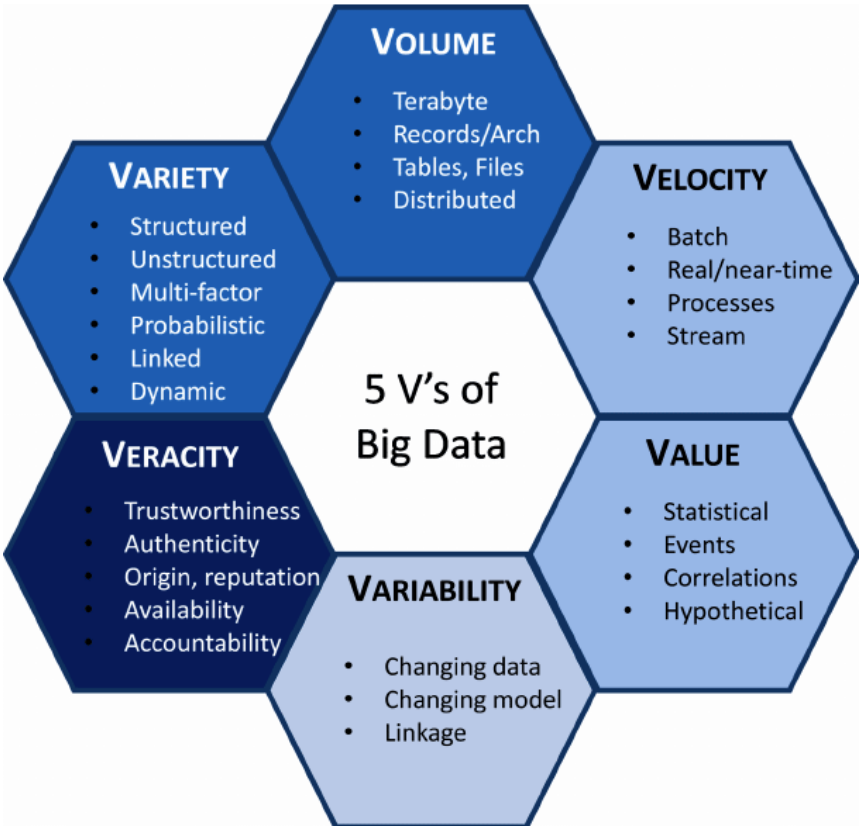

Figure 3 The five main characteristics of big data. Source: [26]

On another point of view, big data can be structured in databases, unstructured in text documents, images, videos, etc. or semi-structured in textual data with pattern XML, etc.

3 Related work

The research on machine learning applications for smart buildings is mainly focused on two large groups: i) solutions focusing on occupancy, e.g. estimating the number of occupants, recognizing their activities or estimating about their preferences or behaviors and, ii) solutions focusing on energy or devices, e.g. energy or device profiling and estimation, fault detection, inference from sensors etc. [27].

3.1 Building Data Genome Project

Several related studies have attempted to develop models that could support advanced building energy systems (BES) and provide measurable improvement in the energy efficiency in buildings. Accurate energy predictions are the most important factor that can optimize the operation and control of BES. The new advancements in big data analytics and machine learning fields equip the researchers with the necessary tools to describe the complex relationships that the data of the buildings form. A large toolset of models is available now, and it keeps growing. However, the development of repositories that may serve as benchmarking tools for the plethora of the new models proposed is of great importance. It is a common problem that a proposed model may be tested against a specific dataset but may not generalize well when it is tested against another. The Building Data Genome Project (BDG) dataset which was utilized in this study, is aimed to serve as a benchmarking tool by its creators. [9] There have been several studies published about it, even though it is relatively new. The following review is focused on the use of the dataset by researchers from the year of publication in 2017 until now.

In 2017, Miller C., proposes in his study [28] a preliminary methodology about non-residential buildings that have advanced metering (AMI) meters. This work falls into the building retrofit area of interest and results in statistics, model and pattern-based temporal features extraction from over 36,000 smart meters. Classification models, such as Random Forest are deployed for this purpose, which showed an 18.3% increase in accuracy of predicting if a building would perform well after a retrofit is made and an 27.6% increase in accuracy in predicting the industry type of a building.

Taheri M. et al., in their work [29] present the idea of efficiency factors which are calculated from timeseries of energy, weather, and occupancy (represented by usage). Their research explores the prediction of what are the modulations on the building design side to modulate the effect of the weather and finally provide a comfortable environment to its occupants. Linear and non-linear regression models such as Linear Regression (LR), Polynomial Regression (PR) and Gradient Boosted Trees Regression (GBR) were utilized to present a method that helps in benchmarking the new buildings, on the basis that variation in weather and human usage, creates variation in the energy usage.

Park J.Y. et al, in their research [30] deal with load profiling and benchmarking for buildings. The main idea is based on the new data-driven approaches that emerge where the shape of the load profiles is used as a means of comparison. A total of 3829 buildings in this work are analyzed with clustering methods followed by entropy calculation for each building. This approach contradicts the traditionally used classification methods for such applications. The results may be of use to portfolio management applications, building and urban energy simulations, demand response and renewable energy integration in buildings and more.

Also in 2019, Miller C. in his work [31] used the BDG dataset to evaluate several feature engineering and data-driven classification models. With the aim of the study being to provide explainable machine learning models for prediction and classification purposes in building applications, this study is the first one to focus on this field of smart meter data from non-residential buildings.

Fang X. et al, in their study [32] propose a novel hybrid deep transfer learning strategy for short-term cross-building energy prediction. They make use of long short memory (LSTM) for feature extraction and domain adversarial neural networks (DANN) for finding domain invariant features that could be adapted to the unknown target buildings. The results have shown that the building energy prediction can be improved significantly.

In another new study, Nichiforov C. et al. [33] utilize the BDG database for testing and evaluating their proposed method. With the purposes of extracting information, anticipating possible future faults and finally performing domain-specific load profiling, their Matrix Profile (MP) technique is applied, which is based on a model free approach. The results justify the scope of their study as their analysis provides higher level information

which speed up the analysis with more advanced methods and provide a baseline for online implementation or real time energy building management systems. Adding up to the findings of the previous study, Nichiforov C. et al, in continuing their research by utilizing the MP introduced a technique for feature extraction in the time series of BDG dataset and anomaly detection. On a second stage, several classification algorithms such as decision trees, nearest neighbors, support vector machines and regression trees were utilized to discriminate among the dominant usage patterns of the buildings. The diversity of the used dataset helped in proving that, the unusual behavior in energy consumption patterns is sufficient enough for differentiating between usage patterns. This leads to fast approaches in decision making and control systems where historical data reach a possible minimum.

Li A. et al,[34] in their research make use of transfer learning based artificial neural network (ANN) methods that could act as a baseline for building energy prediction models when a limited amount of data is available. Their analysis on 400 buildings of the BDG database revealed a significant improvement in accuracy of back-propagation neural network (BPNN)- based building energy models for buildings with little available training data. Moreover, through the analysis of the available features of the buildings, it was derived that the most influential buildings features were the building usage and industry.

In 2019, ASHRAE hosted the Great Energy Predictor III (GEPIII) machine learning competition on the Kaggle platform and the BDG dataset was one among the 16 different data sources. The result was 2380 energy meters for over 1448 buildings and over 20 million of training data that were provided to the competitors. The results showed that with great difference the Gradient Boosting methods such as Light GBM, CatBoost, XGBoost, and LiteMORT resulted in greater accuracy in the final prediction models. Moreover, some of the top solutions used Multi-Layer Perceptron, Feed-Forward Neural Networks and Random Forest models with very good results.[35] The ASHRAE successful competition was followed by a new version of the BDG dataset, which incorporated the new timeseries that were added for the competition purposes. A BDG 2 dataset is available with 3,053 energy meters from 1,636 non-residential buildings with a range of two full years (2016 and 2017) at an hourly frequency (17,544 measurements per meter resulting in approximately 53.6 million measurements) [36].

In 2020, another research team by Wang Z. et al., [37] proposed a new method for generating realistic electrical load profiles of buildings through the Generative Adversarial Network (GAN). This is a machine learning technique which is used to extract an unknown probability distribution from plain data. The results showed that with the proposed model, the general trend and the random variations of the actual electrical loads are captured. Moreover, new building electrical loads can be generated, other profile generation models can be verified, changes to load profiles can be detected and smart meter data can be anonymized for research promoting reasons.

4 METHODOLOGY

The aim of this study is to utilize the open dataset of the Building Data Genome Project for predicting the consumption load for several commercial buildings. In the present work this was attempted with the use and evaluation of well-known, conceptually simple, yet diverse data mining models. At first, the raw dataset is explored and preprocessed. The first case is explored via four different regression models. At a second case hyperparameter tuning methods are utilized to improve the performance of the selected algorithms. Lastly, another task which is performed, is utilizing a classification model: predicting the type of a building via classification methods.

4.1 Problem

Buildings are essentially a system that consists of inflows and outflows and their high performance is achieved through carefully measurement, regulation and control. These processes include day-to-day operations which finally, influence occupants' health and comfort, energy performance and the cost of utilities. Although the application of machine learning methods and data-driven processes in the building life cycle has been extensively researched, there are still some problems that remain unsolved [38].

It has been observed that most of the research results and novel methods will not reach the industry in the imminent future or even never. This problem is attributed to several reasons. It is certain that the low availability in open labeled data as training sets for the models, affects the number of experiments and evaluations which are performed. This could also lead to another possible obstacle, that of model transferability. There have been transfer learning research approaches for solving this. [32] However, the paradigms where a model is deployed and explored based on a building's data and is then transferred and applied to another building are limited. One more possible cause for the slow process towards the industrialization of the research, is that the cost benefits are not measured or clear enough to promote this. Finally, it is evident that in every proposed novelty there comes an estimated possibility of success. The estimated accuracy of most models and their capacity to generalize is an aspect that usually needs to be examined further.

Relevant to the open training data availability, is the fact that the majority of published research presents different machine learning methods and models which are trained and validated on different training data. This causes great difficulty in the comparison between them, as no safe conclusion can be drawn. A benchmarking process is of great importance in this situation. There need to be large scale open datasets available upon which different machine learning approaches could be evaluated. ASHRAE Global Thermal Comfort Database [39], ASHRAE Great Energy Predictor III [40], Building Data Genome Project [9] and the revised Building Data Genome Project 2 [36] datasets have offered a preliminary solution on that. Even more attempts about different types of buildings would be beneficial.

Finally, it is known that many of machine learning solutions are difficult to interpret and explain. This poses restrictions on how a machine learning data-driven, or so called black-box, model can provide interpretable results. Some of the proposed solutions focus on the integration with the physical models and implementation of physical domain knowledge into the data-driven models [38].

This study is dedicated to deploy non-complex, interpretable and well-known but different in their core models, and measure their performance in energy load prediction. The selected dataset is public, open and accessible to everyone. It is examined how targeted hyperparameter tuning affects the accuracy of the model and how the special characteristics of the buildings used as features can add to accuracy.

4.2 Dataset description

As only a few public data sources of hourly non-residential meter data exist for the purpose of testing algorithms, the Building Data Genome dataset was chosen for this analysis. It is a collection of 507 whole building electrical meters. The majority of which come from university campuses. This dataset is the result of the ‘‘Building Data Genome Project’’ of Clayton Miller and Forrest Meggers [9] and it serves as a repository of open, non-residential data sources which can be built upon by other researchers.

For each one of the buildings of the dataset a set of about a year hourly electrical energy consumption values are gathered. The time period of the measurements spans from 01-01-2010 to 01-01-2016. Along with the raw data file, there is a metadata file available providing information on some of the buildings’ characteristics, e.g. surface area, primary heating type, as well as the location of the buildings, e.g. city, continent, and the

primary use of them, e.g. educational, office etc. It is very interesting that the researchers were able to incorporate weather files for each one of the locations of the buildings, which add to the potential analysis tasks. Snapshots of the raw data file and the metadata file may be found at Appendix A. Some characteristics for the available data are found in Figure 4 Distribution of case study buildings among time zone, industry, sub-industry and primary use type. Source [9].

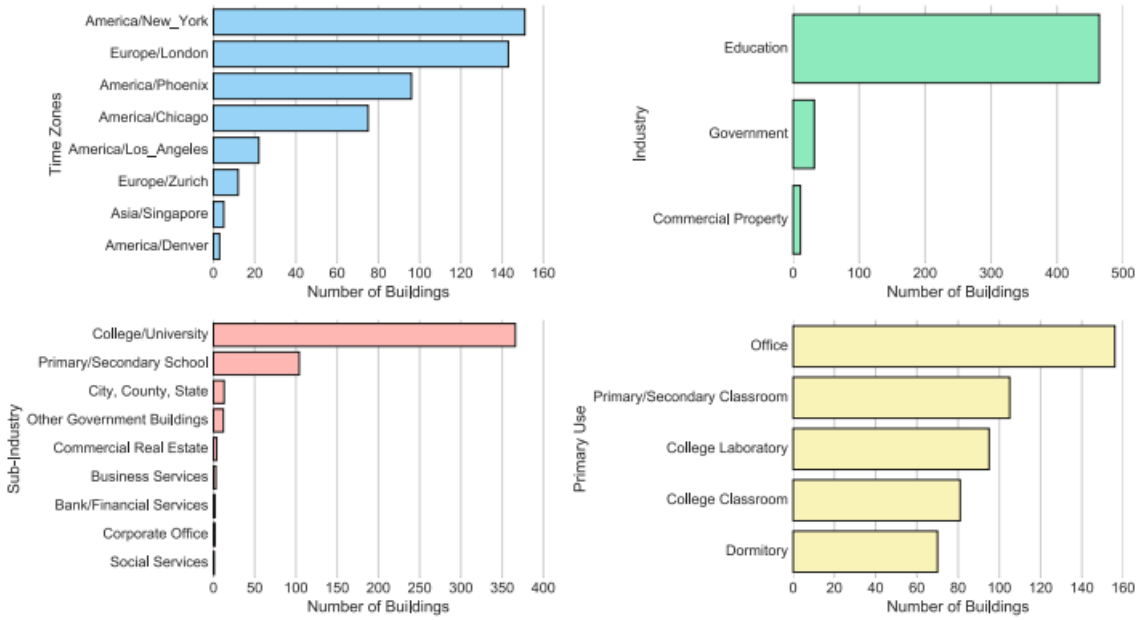

Figure 4 Distribution of case study buildings among time zone, industry, sub-industry and primary use type. Source [9]

4.3 Machine learning models

The load forecasting strategy of this study is based on several regression models. The length and variety of data in the dataset pointed towards this direction since, the more data available for a regressor the better it is trained. The models which were used are Random Forest Regression, Linear Regression, Gradient Boosting Regression and Extreme Gradient Boosting Regression. For the building type classification task, the Random Forest Classifier and Gradient Boosting Classifier were used.

4.3.1 Random Forest Classification and Regression

Random Forest (RF) is a combination of decision trees. Decision trees are splitting a dataset depending on the feature value in different trees and branches (directions). The RF method consists of a large number of such decision trees that operate as a group. The model's prediction is similar to the one that the majority of the trees have predicted. What need to be noted at this point is that bagging and feature randomness are used to build each of the trees. As a result, the uncorrelated forest of trees that this process results in has increased accuracy in comparison to the individual trees [41].

4.3.2 Gradient Boosting Classification and Regression

The Gradient Boosting (GB) or the Gradient Boosting Decision Trees (GBDT) is an accurate method that generalizes in the boosting of arbitrary differentiable loss functions.[42] It is a machine learning technique used for regression or classification. The main idea is to produce a model constructed of decision trees, which are weak prediction models. It incorporates different stages in the model construction and generalizes them by optimizing an arbitrary differentiable loss function. [43] The model performance is measured by computing the cost of predicting $\hat{y_i}$ instead of actual value y_i. The loss across all N observations is just the average of all the individual observation losses: [44]

4.3.3 Linear Regression

Linear regression is a method for modelling the linear relationship between a dependent value and one or more independent variables. It can be categorized in simple or multiple linear regression depending on the number of independent variables. [45] The linear relationship between the dependent and independent variables is modeled by using linear predictor functions. The parameters of those functions are unknown and are estimated from the training data of the model. A variety of methods for fitting linear regression models to data may be used, but the most usual method used is Least Squares Estimation.

4.3.4 Extreme Gradient Boosting Regression

The Extreme Gradient Boosting regression method or XGBoost, belongs to the decision trees methods. It is an ensemble method, like GB. This method has several advantages which make it one of the most preferred in machine learning model deployment. It is

based on a really powerful algorithm that shows great speed and good performance. This aspect is empowered by the ability to run in multicore computers and exploit the most of the processing power available. Moreover, large sets are handled well in the training phase. These characteristics make this model outperform the more simple algorithms. [46]

4.4 Optimization

Each of the algorithms has parameters that can be appropriately selected for great performance of the model. Besides the hyper-parameters there are also techniques that boost performance of the models when carefully selected. In this work both were implemented in different cases.

4.4.1 Hyper-parameters

In Random Forest regressor, the parameters that were used to boost performance of the model are the 'n_estimators', which represents the number of the trees in the forest. The default value is 100 and the 'max_depth', which is the maximum depth of the tree. A value of 'None' makes the algorithm keep iterating until pure leaf is reached.

The Gradient Boosting regressor could be optimized by tuning a few parameters like the 'loss'. This parameter chooses the loss function to implement. Another parameter to tune is the 'max_features', which chooses the number of features to consider when looking for the best split. In this study, the 'max_depth' was chosen, which defines the maximum depth of the individual regression estimators and the nodes of the tree.

In the XGBoost model, the booster parameter was selected for tuning. This parameter controls the booster that is implemented in the model and it can be linear or tree based.

4.4.2 Optimization techniques

There are several boosting and optimization techniques available for better performance of the models. In this study two of them were applied: Kfold cross validation and Grid Search CV.

Kfold cross validation is basically a statistical method which is used mainly to compare the evaluation of models and choose the best one for a particular problem. The idea is to perform shuffling on data and take several samples from them to train the model. This

way the algorithm is able to generalize well. The number of samples is defined by the k folds. There exist multiple shuffling choices for the data, such as the stratification option, which is used for data with imbalanced classes. In this option the data are sampled from each class according to the length of the class.

Grid Search CV is mostly used for tuning the parameters of a model. It can be exhaustive and explore all the possible combinations of the parameters or it may be exploring only the ones given by user. In Scikit Learn Library is implemented together with the k-fold cross validation method.

4.5 Evaluation and metrics

4.5.1 Regression evaluation and metrics

In regression problems the predicted values are continuous real numbers. The main idea in evaluation of regressor performance is to measure the distance between the real value and the predicted value. Although a variety of measures is available for evaluating the performance of a regressor, this work uses only three of them. These are the Coefficient of Determination or Adjusted R denoted as R^2 , the Mean Square Error or MSE and the Root Mean Square Error or RMSE.

The Mean Square Error, MSE measures the average Euclidean distance. The optimal number is the minimum number for this metric. If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the mean squared error (MSE) estimated over n_{samples} is defined as shown in Figure 5

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2$$

Figure 5 MSE calculation formula Source: [47]

Root Mean Square Error (RMSE) is one of the most widely used measures of the error of a model in predicting quantitative data. In essence it is the standard deviation of the residuals and shows how spread they are. It is a measure that tells how concentrated the data are around the line of fit. Mathematically is expressed as the square root of MSE, Figure 6:

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2}$$

Figure 6 RMSE calculation formula. Source: [48]

The coefficient of determination, R^2, expresses the proportion of the variance that can be explained by the independent variables in the model. It is a measurement of goodness of fit of the model to new samples. The optimum score is 1 and it can also take negative values when the model is worse. A value close to zero would mean that the model does not change regardless of what input features it has. If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value for total n samples, the estimated R^2 is defined as, Figure 7:

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

Figure 7 R^2 calculation expression. Source: [49]

where $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$

4.5.2 Classification evaluation and metrics

The most widely known and frequently used metrics to evaluate a classifiers performance are: accuracy (or recognition rate), sensitivity (or recall), specificity, precision, F1, and Fβ. In this study only accuracy, sensitivity, specificity and F1 are utilized. Some basic terminology to classification metrics is needed before the previous mentioned metrics are further explained.

Regarding the classifier's label recognition, there are the positive tuples (those of the main class of interest) and the negative tuples (all the other tuples). The positive ones are denoted as P and the negative ones are denoted as N. When evaluating a tuple a

comparison is made between the classifier's class label prediction and the tuple's known label.

The evaluation result is a summary of the metrics above, explained in terms of positive and negative tuples and it is called confusion matrix. The following explanation of the terms is essential for the understanding of the confusion matrix. The True Positives (or TP) refer to the positive tuples that were labeled correctly by the classifier, while the True Negatives (or TN) are the negative tuples that were correctly labeled by the classifier. False Positives (or FP) are called the negative tuples that were incorrectly labeled as positive and likewise False Negatives (or FN) are called the positive tuples that were mislabeled as negative. The confusion matrix is illustrated in Figure 8

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

Figure 8 Confusion matrix. Source: [22]

The confusion matrix is used to illustrate how good a classifier is performing in recognizing the different classes. The TP and TN values tell if a classifier is classifying the right way while the FP and FN showcase the wrong way that a classifier is working.

With reference to the evaluation metrics discussed at the beginning of this paragraph, these are defined using the formulas in Figure 9 – Figure 15.

The accuracy of a classifier on a given test set is defined as the test set tuples which are correctly classified. The expression is:

$$accuracy = \frac{TP + TN}{P + N}.$$

Figure 9 Definition formula of accuracy Source [22]

Sensitivity is also called true positive recognition rate and is defined as the proportion of positive tuples that are correctly classified. The expression is:

$$sensitivity = \frac{TP}{P}$$

Figure 10 Definition formula of sensitivity Source [22]

Similarly, specificity is the true negative recognition rate and is used to measure the proportion of negative tuples that are correctly identified.

$$specificity = \frac{TN}{N}.$$

Figure 11 Definition formula of specificity Source [22]

There is a relationship between accuracy and specificity and sensitivity measures, which can be expressed as:

$$accuracy = sensitivity\frac{P}{(P+N)} + specificity\frac{N}{(P+N)}.$$

Figure 12 The accuracy formula expressed in terms of sensitivity and specificity. Source: [22]

Precision can be defined as the percentage of tuples which are positive and are actually classified as such. Precision is a measure of exactness and is expressed as:

$$precision = \frac{TP}{TP+FP}$$

Figure 13 Definition formula of precision. Source: [22]

The recall measure expresses the completeness is a measure of completeness and it expresses the percentage of positive tuples that are labelled as positive. The definition formula is:

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}.$$

Figure 14 Definition formula of recall. Source: [22]

Finally, the F1 or F-score measure is just a combination of the precision and recall measures and it is expressed as:

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Figure 15 Definition formula of F1 or F-score. Source: [22]

When a classifier is evaluated, besides the mentioned measures, additional characteristics may be considered. Some of them are robustness, scalability, speed and interpretability. These are more qualitative and give are used to describe the generalization and stability of the classifier when is deployed with different datasets [22].

5 RESULTS

5.1 Data description and preprocessing

5.1.1 Timeseries data

The effectiveness of load forecasting with the use of the proposed machine learning models were evaluated on a whole implementation of the BDG dataset. The 507 different timeseries with hourly resolution were combined in a large dataset. Statistical information on the 507 buildings of the dataset are presented on Table 2.

Table 2 Building energy consumption datasets

Dataset	Building type	Count of timestamps	Hourly energy consumption in kWh			
			Mean	Min	Max	Std
Office	Office	1348456	112.688400	0.005517	2649.300000	177.114300
PrimClass	Primary School Classroom	885145	16.689776	0.000658	298.69999	25.954297
UnivClass	University Classroom	699716	86.935176	0.010000	570.66705	79.512861
UnivDorm	University Dormitory	606554	94.608103	0.017931	641.180000	87.770776
UnivLab	University Laboratory	823396	300.146676	0.720000	3150.060000	349.349919
Combined	All	4363267	121.94590	0.000658	3150.060000	208.601100

Figure 16 illustrates the 507 building energy consumption profiles for the period 01-01-2010 up to 01-01-2016. It is obvious that there are missing values for the year 2011. Moreover, several buildings regardless of their type, do not have consistent data values for a whole year, almost half the days of 2015. This triggered a missing values exploration of the data which resulted in dropping all the missing values of the dataset and keeping for analysis those with meter values. From a total of 20,756,580 rows only 4,363,267 (~21%) of them form the dataset that was used for the machine learning models. This is no surprise, as the creators of the dataset clearly state in their research

paper [9] that for every building only a total of 8760 (a year) of metered data is available.

Figure 16 Energy consumption profiles in kWh, for 507 buildings for the period 01-01-2010 up to 01-01-2016

Unfortunately, the general overview of the raw dataset did not reveal much information and a more in-depth visualization process was conducted to know the timeseries. In Figure 17 – Figure 21, five samples of the timeseries are illustrated. They illustrate the hourly energy consumption values of one building per type, for a year. This is only a sample of the 507 timeseries that are present in this dataset, but it is enough for an overview of what these timeseries may look like.

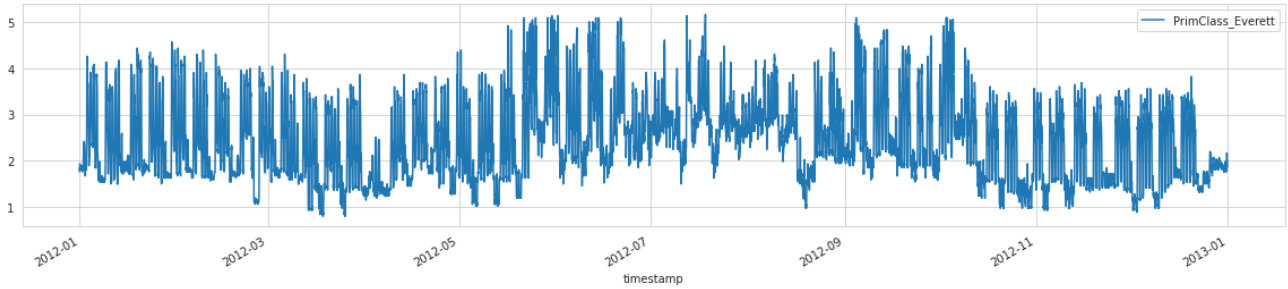

Figure 17 Hourly energy consumption values in kWh, of Primary Classroom Everett for a year

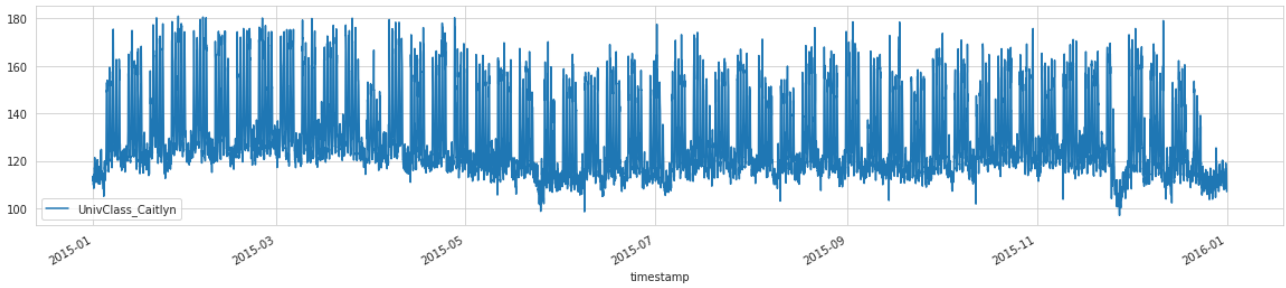

Figure 18 Hourly energy consumption values in kWh, for University Classroom Caitlyn for a year

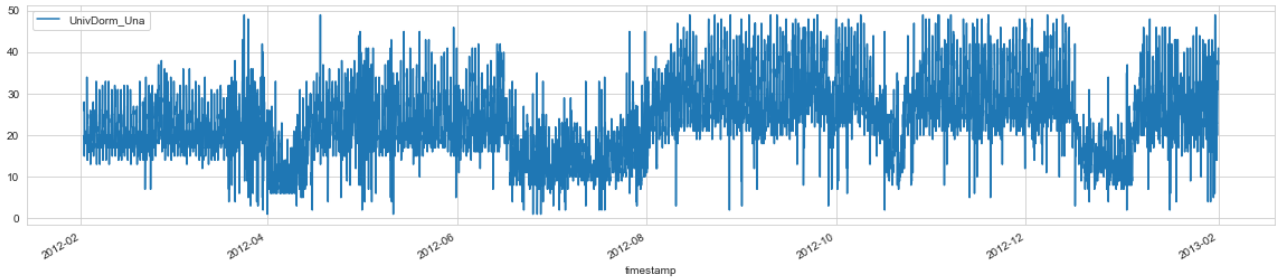

Figure 19 Hourly energy consumption values in kWh, for University Dormitory Una for a year

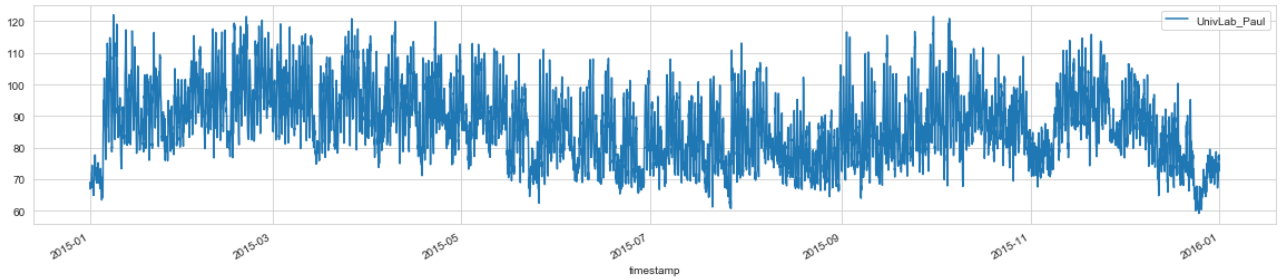

Figure 20 Hourly energy consumption values in kWh, for University Laboratory Paul for a year

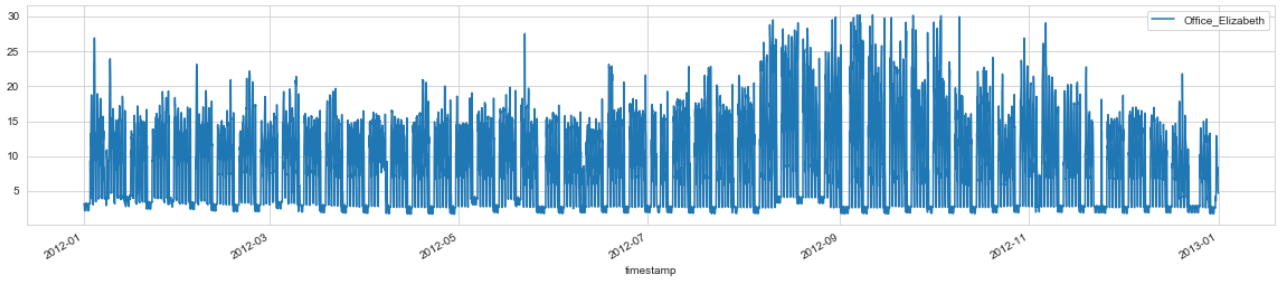

Figure 21 Hourly energy consumption values in kWh, for Office Elizabeth for a year.

Some of the profiles show random patterns while some others, especially the office and classroom profiles, show periodicity and specific patterns. Of course, this is anticipated since the operation hours of these buildings is usually standard.

5.1.2 Data preprocess

As explained above, the exploration of the dataset revealed certain characteristics of the timeseries that could jeopardize the accuracy in analysis. The strategy and the decisions made on how these peculiarities were handled are explained below.

Missing data

First, the large number of missing values was a point of interest, especially the strategy that should be followed for them. A percentage of 21% is relatively low for the keeping instances. From another point of view, there are plenty of timesteps that are kept for analysis, enough for the models to be properly trained and validated. There are no values metered for the year 2011 and for year 2015, a research showed that recorded values exist after May. No model could handle accurately such a disturbance in a timeseries data, even more when the available historical data cover only a year for each building. Having that in mind, the strategy that was followed for missing data was to exclude them from the analysis. After that, the timeseries of many buildings were left incomplete and, in some cases, they represented only half of the year.

Transformation of timeseries data

The previous described decision on the missing values led to another decision. The question was, if we would decide to proceed with the buildings that had metered data for a complete year as a timeseries or proceed otherwise. Careful consideration of the limitations of one or the other choice led to the decision that the most possibly available data points would be the best for the models. The aim of the study is to evaluate the preferred models on big data, meaning also large and diverse. Moreover, the literature review showed that most of the studies proceed with timeseries analysis. Also, the need for explainable and non-complex models, some computation and time limitations which also apply here, pointed towards a different perspective. Finally, the decision was to continue with a combined dataset, were the name of each building would not play an important role. This dataset implemented the notion of time as a feature, the timestamp was split in ‘day of year’ categorical values and the hour of the timestep was a categorical value under the feature ‘hour of day’. If there had been historical data available for

more than a year, this choice would have been different. This transformation resulted in a somehow anonymized dataset where the 'time' was present as a feature. The benefits of this are clearly shown on the low complexity of the analysis, the relatively low computation needs, the fast execution times for this many points and the variety of available methods for feature engineering and hyperparameter tuning methods. On the disadvantages comes the fact that the periodicity and trends of the timeseries are lost. Any information that could be extracted through timeseries analysis are not shown. This last argument though, may not be entirely accurate. In the dataset there are no meta-data available for the buildings' occupancy or the occupants' habits or even the operation shifts. This means, that the predicted total load values could not be explained in a context of specific needs of occupants, or types of loads that are used, or types of devices and the use routine of them. So, the timepoints in the timeseries of the building would carry that information but we could not exploit it in a way.

Creation of a complete dataset

Following the previous steps, the dataset was holding 4,363,267 rows of energy consumption meter described by the building's name, the day of the year and hour it occurred. Figure 22 illustrates an instance of this dataset.

	uid	load	day_of_year	hour
0	PrimClass_Jolie	0.7	335	0
1	PrimClass_Jolie	0.6	335	1
2	PrimClass_Jolie	0.8	335	2
3	PrimClass_Jolie	0.7	335	3
4	PrimClass_Jolie	0.8	335	4
...	...	...	...	...
4363262	PrimClass_Ulysses	37.0	31	11
4363263	PrimClass_Ulysses	39.0	31	12
4363264	PrimClass_Ulysses	44.0	31	13
4363265	PrimClass_Ulysses	34.0	31	14
4363266	PrimClass_Ulysses	10.0	31	15

4363267 rows × 4 columns

Figure 22 Dataset instance after the timeseries transformation.

The next step was to combine the available meta-data for the buildings with the processed temporal data. Instance of the meta-data available for the buildings may be found at Appendix A. The primary meta-dataset contains 19 columns. Unfortunately, not all of them hold data for all buildings. Thus, the features that were not present for all the buildings had to be dropped. A feature called ‘timezone’ held the names of city and continent in which the building is located. This feature was split in two: ‘city’ and ‘continent’. From the 19 columns only 6 remained. The merge process of the two resulted in 4,363,267 rows and 9 columns. Figure 23 showcases the resulting dataframe.

	uid	load	day_of_year	hour	industry	usespace	sqm	continent	city
0	PrimClass_Jolie	0.7	335	0	Education	PrimClass	2927.0	Europe	London
1	PrimClass_Jolie	0.6	335	1	Education	PrimClass	2927.0	Europe	London
2	PrimClass_Jolie	0.8	335	2	Education	PrimClass	2927.0	Europe	London
3	PrimClass_Jolie	0.7	335	3	Education	PrimClass	2927.0	Europe	London
4	PrimClass_Jolie	0.8	335	4	Education	PrimClass	2927.0	Europe	London
...	...	...	...	...	...	...	...	...	...
4363262	PrimClass_Ulysses	37.0	31	11	Education	PrimClass	12000.0	Asia	Singapore
4363263	PrimClass_Ulysses	39.0	31	12	Education	PrimClass	12000.0	Asia	Singapore
4363264	PrimClass_Ulysses	44.0	31	13	Education	PrimClass	12000.0	Asia	Singapore
4363265	PrimClass_Ulysses	34.0	31	14	Education	PrimClass	12000.0	Asia	Singapore
4363266	PrimClass_Ulysses	10.0	31	15	Education	PrimClass	12000.0	Asia	Singapore

4363267 rows × 9 columns

Figure 23 Instance of the merged dataframe of temporal and meta-data.

The last steps of the preprocess were to drop the ‘uid’ column of the data, as this is not actually a feature. The anonymized dataset was left with load values as numerical data and the rest of the columns were converted to dummy categorical values. This transformation is necessary for the models to be trained properly.

5.2 Load prediction

There were 4 different regression models evaluated for load prediction in 7 scenarios.

5.2.1 Preprocess of data

At the first stage of analysis, the dataset had to be split in train and test set. Primary attempts with the models, showed that the best performance is achieved with a 25% test set 75% train set split, which is the default for the Scikit Learn. Other attempts used 20% for the test set or 33% for the test set. Moreover, although the thought of normalization of the data was present, in fact the results from the first round showed that such strategy would not be necessary.

However, this strategy for splitting in test/train data was changed at the second stage. A split of 50/50 was chosen for the second case examined. The reason is that this could save in computational time and complexity as the dataset was very large. Also, Scikit Learn documentation suggested that the test set should be 50% in Grid Search CV for optimal use.

5.2.2 Case analysis

In the first case scenarios for all four regression models, the models were using the dataset at its primary stage, without any feature engineering or optimization techniques. The second case scenarios included 5 Fold cross validation for the train split for three models. This case was combined with hyperparameter tuning through the Grid Search CV method.

Case 1 - models

Random Forest Regressor

The effectiveness of the Random Forest (RF) regressor was evaluated in predicting the load by utilizing the default parameters. The max_depth of the regressor was set to 2 and the number of estimators was 100. The performance of the regressor was evaluated using three metrics: R^2 , MSE and RMSE.

Linear Regression

The second model was Linear (LR) regressor. The parameters were left at their default values and the dataset was at its primary form. The performance of the regressor was evaluated by using the same metrics: R^2, MSE and RMSE.

Gradient Boosting Regressor

The third model which was deployed is based upon a Gradient Boosting (GB) regressor. The dataset was at its primary form and the parameters of the model were set at their default values. Again, the performance was evaluated by utilizing the same metrics as before: R^2, MSE and RMSE

Extreme Gradient Boosting Regressor

The last model in this first case was an Extreme Gradient Boosting (XGB) Regressor. Parameter tuning did not take place in this attempt and the dataset which was at its primary form. The performance was again evaluated by the same metrics as the previous models: R^2, MSE, RMSE.

Case 1 – Results analysis

The results of the case 1 scenarios are summarized in Table 3.

Table 3 Summary of the evaluation metrics of Case 1 models

CASE 1		MODELS			
		RF	**LR**	**GB**	**XGB**
METRICS	**R^2**	0.4098	0.4184	0.8548	**0.8565**
	MSE	25731.5677	25299.6714	6275.8881	**6202.7785**
	RMSE	160.4106	159.0587	79.2205	**78.7577**

The fact that the first stage of model deployment resulted in such good metrics for some models, is interesting. The RF regressor reached a medium score of 40.98%, which is quite good for this length of dataset. The Linear regressor reached a close 41.84% and the RMSE was very close to the Random Forest as well. The great difference comes with the two boosting algorithms. The GB regressor reached 85.48% at this first stage without parameter tuning and the XGB regressor reached a little higher, 85.65%. Their RMSE results were very close, also. The two boosting algorithms are based in the same principles in their core and such a close result was expected. On the other hand, good scores like this were not anticipated, for a difficult dataset.

The transformation of the categorical features with dummy values, played a significant role in this outcome. The binary codes of 1 and 0 after the transformation may result in more features but do not complicate the training phase of the model.

The RF model performed lower than the linear model, in all metrics. However, it is the RF regressor that has more possibilities of reaching higher scores with proper tuning of the parameters. The Linear model has narrow space for improvement.

Case 2 – Models

Random Forest Regressor

The effectiveness of the RF regressor was evaluated in predicting the load by utilizing the default parameters, but this time, a 5 fold cross validation was used for the train and test split of data. The 'max_depth' parameter of the regressor was tuned for 2 or 3 via the use of Grid Search CV method and the number of estimators was set to 100. The performance of the regressor was evaluated using the metrics: average R2, MSE and RMSE.

Linear Regression

In the second case, the LR regressor was left out. The reasons behind this decision are the low performance of the regressor in case 1 and the minimum available parameters for hyper-tuning via Grid Search method.

Gradient Boosting Regressor

The GB regressor. The dataset was split in train and test sets using 5 fold cross-validation. The 'max_depth' parameter was tuned for values 3 and 5. Again, the performance was evaluated by utilizing the same metrics as before.

Extreme Gradient Boosting Regressor

The last method was the XGB regressor. The dataset was used with a 5 fold cross validation at train-test split. The booster parameter of the model was set to 'gblinear' and 'gbtree', to test the effect of the linear and the tree booster in the performance of the model. The performance was again evaluated by the same metrics as the previous models: average R2, MSE, RMSE.

Case 2 – Results analysis

The results of the Case 2 models are summarized in Table 4

Table 4 Summary of the evaluation results of Case 2 models

CASE 2		**MODELS**		
		RF	**GB**	**XGB**
METRICS	**Avg R^2**	0.5486	**0.9438**	0.8574
	MSE	19630.4523	**2503.9616**	6285.0809
	RMSE	140.1087	**50.0396.**	79.2785

The results of the second case after the hyper-parameter tuning and the 5 fold cross-validation implementation have changed compared to the first case. All models have performed better. The RF regressor had an amazing 33.87% increase compared to the first case. The best parameter was the 'max_depth' equal to 3 and number of estimators was chosen at 100. The increase is attributed partially to the 'max_depth' change from 2 to 3 and partially to the 5 fold cross validation. The 'max_depth' parameter controls the depth of the tree, thus the available nodes, so an increased number is sure to result in greater separation of the data. The XGB regressor had a small increase, since the score at first round was already high. It reached 85.74% and was increased by 0.11%. The optimization through 5 fold cross validation was responsible for that increase. The best booster parameter although tuned, was already the best as default, of 'gbtree'. The alternative linear booster did not perform better than the tree-based one. A very interesting change in the second stage is the increase in GB regressor. In the first case reached 85.48% by learning from the primary data. The second stage was different in the 5 fold cross validation which was applied to the training dataset and the change of the max_depth parameter which resulted in 5, as the best. These two alterations increased the score by 10.42%.

5.3 Prediction of the building type

In research, clustering methods are prevailing in building type prediction. However, in this case, two classification algorithms were deployed and evaluated in predicting the building type. A positive aspect towards the selection of these classifiers, was the fact that they have performed well as regressors in load prediction. More complex classifiers were left out of the selection. A problem like this, with the complexity and computational cost that the large number of datapoints poses, would be very difficult to handle.

5.3.1 Preprocess of data

For this classification task, the data had to be transformed suitably. Although, the major preprocess part of the dataset remained the same as for the regression tasks, a few final preprocess steps were added. First, the label column had to be created in data. On the regression tasks the target values were the load values, while in this case the target value is the type of the building. The two features, 'industry' and 'usespace', were concat-

enated resulting in the ‘label’ feature. Next, the two old features had to be dropped. The ‘continent’ and ‘city’ features had to be transformed into dummy values for simplicity. Finally, the ‘label’ had to be transformed from string to numbered categories for simplicity, also. The result was 9 classes. Table 5 shows the resulting classes with the number of tuples they contain. It is evident that the problem is imbalanced.

Table 5 Classes and number of tuples per class

Class	**Number of tuples**
Class 1	1040847
Class 2	835753
Class 5	823396
Class 3	699716
Class 4	598219
Class 6	212203
Class 0	95406
Class 7	49392
Class 8	8335
9 classes in total	**4,363,267 tuples**

The train/test split in this case used the default values of 25% test set and 75% train test. The imbalance of classes was considered by setting the’ stratify’ option in test/train split to ‘Yes’. This adjustment would result in a more balanced problem. They were not cross validation or other optimizing techniques applied in this step.

5.3.2 Case analysis

Case 1 – Models

Gradient Boosting Classifier

The first scenario was to deploy a GB classifier model to predict the building type. At this stage the parameters of the model were used with their default values. The metrics

of the classification with the GB model sere a full classification report, with accuracy, precision, recall, f1-score values calculated.

Random Forest Classifier

This scenario used an RF classifier as a model. The dataset was at its preprocessed form as described above and the 'max_depth' parameter was set to 3, all other parameters were set at their default values. A full classification report was produced, also.

Case 1 – Results analysis

Table 6 below, illustrates the mean accuracy score of the two models. Clearly, the GB classifier has performed better than the RF. Moreover, the difference between the two is significant, to a point that RF would need wide hyper-parameter tuning.

Table 6 Mean accuracy of the classification models

CASE 1	**MODELS**	
	RF	**GB**
mean accuracy	0.4192	**0.9083**

The full classification reports of both classifiers are illustrated at Figure 25 Gradient Boosting, classification report and Figure 24 Random Forest, classification report.. The complete confusion matrices for both models are available at Appendix B, Figure 26 Confusion matrix of the Gradient Boosting classifier- Figure 27 Confusion matrix of the Random Forest classifier For the GB classifier the precision scores are very good in total, and only in the fourth class seems to be low. Of course, the recall score tells that the estimation for great performance is a little optimistic for some classes, for example the third class has a high precision, but the recall value is lower, meaning that some tuples which are classified in class were not actually true. The f1-scores of the model are quite high for most of the classes, which is a measure of balanced values between the precision and recall for this problem. In overall, most of the classes are well classified by the model and this makes it a successful choice of algorithm for this problem.

For the RF classifier the results on the classification matrix are revealing. The precision scores are extremely low, and the recall are at a medium level. There are classes that have not been correctly classified, at all. For example, class 0, class 3, class 4, class 7,

class 8, have a recall value of 0.00. Only class 1 and class 2 are somehow correctly classified. The bad performance of the classifier for most of the classes, is shown in the f1-scores, too. Most of them fall at 0.00 level. The overall performance of RF classifier in this problem make it a bad choice for this problem.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	23851
1	0.89	0.89	0.89	260212
2	0.92	0.98	0.95	208938
3	0.90	0.86	0.88	174929
4	0.86	0.87	0.87	149555
5	0.91	0.89	0.90	205849
6	1.00	0.97	0.98	53051
7	1.00	1.00	1.00	12348
8	1.00	1.00	1.00	2084
accuracy			0.91	1090817
macro avg	0.94	0.94	0.94	1090817
weighted avg	0.91	0.91	0.91	1090817

Figure 25 Gradient Boosting, classification report

	precision	recall	f1-score	support
0	0.00	0.00	0.00	23851
1	0.34	0.81	0.48	260212
2	0.50	0.81	0.62	208938
3	0.00	0.00	0.00	174929
4	0.00	0.00	0.00	149555
5	0.54	0.34	0.42	205849
6	1.00	0.12	0.21	53051
7	0.00	0.00	0.00	12348
8	0.00	0.00	0.00	2084
accuracy			0.42	1090817
macro avg	0.26	0.23	0.19	1090817
weighted avg	0.33	0.42	0.32	1090817

Figure 24 Random Forest, classification report.

6 DISCUSSION

In this study 9 models in total were deployed exploring various load prediction and building type prediction tasks. For load prediction tasks, 4 different machine learning algorithms were utilized in 2 different cases and 7 model deployments in total. For the building type prediction task, 2 different machine learning algorithms were evaluated in 1 case.

The first case in load prediction, includes 4 regressor models, Random Forest, Gradient Boosting, Linear and XgBoost which were evaluated on a train set of the dataset which was lightly preprocessed. The results showed that among the four models the XGB regressor outperformed the others with a score very close to the GB regressor. The XGB reached a 85.65% score in performance and the GB 85.48%. This is almost a difference of ~0.2% between them. In comparison to the other metrics, where their differences reach ~1.1% for the MSE and ~1.2% at the RMSE. In total, it could be deducted that the two models performed similarly in this problem without even minor optimizing handlings. The other two algorithms, RF with 40.98% and LR with 41.84% also performed similarly to one another. In comparison to the two boosting algorithms, though, they showed a difference of more than 50% decreased performance. This is clear evidence that boosting techniques or hyper-parameter tuning is needed for them to perform better.

In the second case, the three RF, GB, XGB regressors were slightly tuned and boosted through 5 fold cross validation of the train data, different train/test set split and hyper-parameter tuning. The results showed that for the XGB the margin for improvement was narrow, for the GB a little wider and for the RF it was a lot better. In particular, the XGB model reached 85.74% by increasing this score by ~0.10% in comparison to case 1. In this case, the GB outperformed the other two, even the XGB, by reaching 94.38% score. This is a very high score and the total increase compared to case 1 is close to 10%. The evidence that the RF model would benefit from optimization techniques was proven to be right, since the score reached 54.86% increased by almost 34% from the first case. Although, the RF algorithm performed poorly in this second case compared to

the other two, it is clear that there is probably still room for improvement. Of course, it is highly unlikely that it will reach the high scores of the boosting algorithms.

The final task of building type prediction via classification methods, was evaluated as successful due to the high score of the GB classifier. In this case, only two models were deployed, RF and GB. The data was again slightly preprocessed with most of the values transformed to numerical categories for simplicity. The GB achieved a score of 90.83% and the RF reached 41.92%. Again, as it was found at the load prediction part of the study, RF would have probably performed better if more optimizing had been done before the deployment. The classification report results show the degree of the GB classifier's superiority in this classification task compared to the RF. All scores, are increased with the f1-scores of GB reaching the level of 1 for most classes. In contrast to the RF results, where most of the tuples are misclassified and the corresponding f1-scores are at the 0.0 level. The GB classifier has performed better both on the precision and on recall metrics, revealing a balance between the two.

Regarding the preprocess and analysis of the dataset, the choice of mild preprocess before the first case model deployments is justified. This way, the power of the boosting algorithms was shown from the very first stage. On the second case scenarios were hyper-parameter tuning had been applied and cross validation and train/test split alternatives were explored, it was shown that the decision tree algorithms had potential for better performance. The same applies for both the load prediction and the building type prediction tasks.

In total, the models have performed as expected from the published research. Although, not many benchmarking datasets are available, and the ones that exist have not been tested thoroughly with non-complex machine learning algorithms, many researchers have pointed out the superiority of boosting algorithms in large datasets.

With respect to the research questions of the present study, the deployment of several models with specific characteristics regarding their simplicity, non-complexity and performance was achieved and the results were consistent with the research. Evaluation and exploration of them and their ability to perform better was investigated through hyper-parameter tuning and optimization techniques. A prediction for building type was achieved reaching a high level of accuracy from a large dataset through a simple model approach. Moreover, it could be deducted from the above analysis that the models could

generalize well when faced with new data. The extend of the dataset was large enough to support this.

7 CONCLUSION

7.1 Study review

A benchmarking data set which is public and open was utilized in this study with the aim to evaluate the performance of specific machine learning algorithms in load prediction and to test two classification algorithms in building type prediction. The algorithms that were used performed as it was anticipated from the research review. In load prediction the two of the regression models Gradient Boosting and Extreme Gradient Boosting performed very well while the Random Forest and Linear model performed at a medium level. The second attempt, after hyper-parameter tuning and optimization techniques were used, revealed that the Gradient Boosting and Extreme Gradient Boosting algorithms stile performed at great levels, while the Random Forest algorithm showed significant improvement but still at a lower level than the other two. In building type prediction from load data, the Gradient Boosting and Random Forest models were evaluated in classification with the first one showing great performance. The Random Forest would perform better if some hyper-parameter tuning took place.

Although, in this study it was not the case to test thoroughly and exhaustively the capabilities of the algorithms, an overview of them is shown. The idea of testing them with a large dataset with a significant number of non-complex features has proven to be important. The research questions posed at the beginning of this study are answered and the results, agree with the general trend in research.

With respect to the knowledge gained from this study, it could be stated that it gave a clear perspective of how a very big dataset behaves. Moreover, the accompanying limitations in time and computational power limitations have proven to be beneficial for the decision-making process. Not many random choices were made and a thorough research for the parameters and methods to be employed was necessary.

7.2 Threats on validity

It is common for studies that are constrained by time limitations and computational resources availability to reach conclusions that are subjected to further analysis. The work presented here is to the best knowledge of the researcher, so far.

Regarding missing values strategy, the exclusion from the beginning of a few missing rows in the middle of a full dataset could be avoided with proper imputation methods that result in a complete timeseries. This would result in greater generalization abilities for the models.

The weather data were not utilized as attributes in the studied dataset and this probably has an impact in the evaluation result of the models. Several studies have incorporated such data as features and the results were improved. From a realistic point of view this would be expected to happen in an energy management system that predicts daily or monthly load consumption.

Extensive feature research was not conducted in this study, although it would supply interesting alternatives in the training datasets for the models. This is a possible threat to generalization ability of the models.

Finally, due to limited time only a simple classification task was performed. The dataset is suitable for extensive classification tasks for energy prediction or building type label prediction.

7.3 Future work

This study was conducted with the aim to evaluate some of the most well-known regression and classification models in load prediction and complementary to test the performance of two models in building type label prediction. Of course, the evaluation process is not complete, nor the selection of models is exhaustive. The dataset has been extensively studied since 2017 with a variety of proposed models and with a variety of predicting tasks. If it were for future research, the following points would be a good start.

- A wider variety of base models should be evaluated by using this dataset. This proposal includes a large pool of novel or hybrid models that could work on

such a type of data. This would result in a database of evaluation results of different models on a benchmarking dataset such as BDG dataset.

- The new revised BDG 2 should be utilized in replicating the process presented in this study, as well as more models as proposed above. The new dataset is larger and incorporates more types of buildings. These new timeseries would add to the computation time of the training phase but the resulting models should generalize better in unknown data.
- The decent number of features which are provided as meta-data with the open dataset, is suitable for experimentation and extended feature engineering. More features could be extracted from the present ones and interesting combination would probably give to low score models, a new perspective and increase their scores.
- An interesting approach would be for the dataset to be further analyzed as a timeseries. The resulting models would provide for comparison with the proposed one in this study.
- Finally, as it is studied in research building type label prediction should be also tested through a variety of clustering methods.

Bibliography

[1] Oxford Institute, “Smart Cities Research.” https://smartcities.oii.ox.ac.uk/ (accessed Oct. 05, 2020).

[2] European Commission, “Smart cities,” *European Commission*. https://ec.europa.eu/info/eu-regional-and-urban-development/topics/cities-and-urban-development/city-initiatives/smart-cities_en (accessed Oct. 05, 2020).

[3] R. K. R. Kummitha, “Smart cities and entrepreneurship: An agenda for future research,” *Technological Forecasting and Social Change*, vol. 149, p. 119763, Dec. 2019, doi: 10.1016/j.techfore.2019.119763.

[4] “5 Ways Businesses Benefit From Smart Cities | Articles | Chief Innovation Officer.” https://channels.theinnovationenterprise.com/articles/5-ways-businesses-benefit-from-smart-cities (accessed Oct. 05, 2020).

[5] I. Shahrour, “Smart City – Isam shahrour.” http://ishahrour.com/en/smart-cityen/ (accessed Oct. 05, 2020).

[6] I. A. T. Hashem *et al.*, “The role of big data in smart city,” *International Journal of Information Management*, vol. 36, no. 5, pp. 748–758, Oct. 2016, doi: 10.1016/j.ijinfomgt.2016.05.002.

[7] H. Habibzadeh, A. Boggio-Dandry, Z. Qin, T. Soyata, B. Kantarci, and H. T. Mouftah, “Soft Sensing in Smart Cities: Handling 3Vs Using Recommender Systems, Machine Intelligence, and Data Analytics,” *IEEE Communications Magazine*, vol. 56, no. 2, pp. 78–86, Feb. 2018, doi: 10.1109/MCOM.2018.1700304.

[8] X. He, K. Wang, H. Huang, and B. Liu, “QoE-Driven Big Data Architecture for Smart City,” *IEEE Communications Magazine*, vol. 56, no. 2, pp. 88–93, Feb. 2018, doi: 10.1109/MCOM.2018.1700231.

[9] C. Miller and F. Meggers, “The Building Data Genome Project: An open, public data set from non-residential building electrical meters,” *Energy Procedia*, vol. 122, pp. 439–444, Sep. 2017, doi: 10.1016/j.egypro.2017.07.400.

[10] A. Cocchia, “Smart and Digital City: A Systematic Literature Review,” in *Smart City: How to Create Public and Economic Value with High Technology in Urban Space*, R. P. Dameri and C. Rosenthal-Sabroux, Eds. Cham: Springer International Publishing, 2014, pp. 13–43.

[11] M. Angelidou, “Smart cities: A conjuncture of four forces,” *Cities*, vol. 47, pp. 95–106, Sep. 2015, doi: 10.1016/j.cities.2015.05.004.

[12] V. Albino, U. Berardi, and R. M. Dangelico, “Smart Cities: Definitions, Dimensions, Performance, and Initiatives,” *Journal of Urban Technology*, vol. 22, no. 1, pp. 3–21, Jan. 2015, doi: 10.1080/10630732.2014.942092.

[13] H. Ahvenniemi, A. Huovila, I. Pinto-Seppä, and M. Airaksinen, “What are the differences between sustainable and smart cities?,” *Cities*, vol. 60, pp. 234–245, Feb. 2017, doi: 10.1016/j.cities.2016.09.009.

[14] R. P. Dameri, *Smart City Implementation: Creating Economic and Public Value in Innovative Urban Systems*. Cham: Springer International Publishing, 2017.

[15] L. G. Anthopoulos, “Understanding the Smart City Domain: A Literature Review,” in *Transforming City Governments for Successful Smart Cities*, M. P. Rodríguez-Bolívar, Ed. Cham: Springer International Publishing, 2015, pp. 9–21.

[16] B. van Bastelaer, “Digital cities and transferability of results,” p. 15, 1998.
[17] R. Giffinger, C. Fertner, H. Kramar, and E. Meijers, “City-ranking of European medium-sized cities,” *Cent. Reg. Sci.*, pp. 1–12, Jan. 2007.
[18] T. Nam and T. A. Pardo, “Conceptualizing smart city with dimensions of technology, people, and institutions,” in *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*, New York, NY, USA, Jun. 2011, pp. 282–291, doi: 10.1145/2037556.2037602.
[19] International Telecommunications Unit, “Focus Group on Smart Sustainable Cities.” https://www.itu.int/en/ITU-T/focusgroups/ssc/Pages/default.aspx (accessed Oct. 06, 2020).
[20] ITU - T Focus Group on Smart Sustainable Cities, “TR Smart sustainable cities an analysis of definitions,” Oct. 2014. https://www.itu.int/en/ITU-T/focusgroups/ssc/Documents/website/web-fg-ssc-0100-r9-definitions_technical_report.docx (accessed Oct. 06, 2020).
[21] P. Neirotti, A. De Marco, A. C. Cagliano, G. Mangano, and F. Scorrano, “Current trends in Smart City initiatives: Some stylised facts,” *Cities*, vol. 38, pp. 25–36, Jun. 2014, doi: 10.1016/j.cities.2013.12.010.
[22] J. Han, M. Kamber, and J. Pei, *Data Mining. Concepts and Techniques*, 3rd ed. USA: Morgan Kaufmann Publishers - Elsevier, 2011.
[23] S. Agarwal, “Data Mining: Data Mining Concepts and Techniques,” in *2013 International Conference on Machine Intelligence and Research Advancement*, Dec. 2013, pp. 203–207, doi: 10.1109/ICMIRA.2013.45.
[24] J. Brownlee, “Supervised and Unsupervised Machine Learning Algorithms,” *Machine Learning Mastery*, Mar. 15, 2016. https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ (accessed Jan. 04, 2021).
[25] “(PDF) Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey.” https://www.researchgate.net/publication/335810150_Machine_Learning_Methods_for_Reliable_Resource_Provisioning_in_Edge-Cloud_Computing_A_Survey (accessed Jan. 04, 2021).
[26] J. Moura and C. Serrão, *Handbook of Research on Trends and Future Directions in Big Data and Web Intelligence*, vol. Security and Privacy Issues of Big Data. IGI Global, 2015.
[27] D. Djenouri, R. Laidi, Y. Djenouri, and I. Balasingham, “Machine Learning for Smart Building Applications: Review and Taxonomy,” *ACM Comput. Surv.*, vol. 52, no. 2, p. 24:1-24:36, Mar. 2019, doi: 10.1145/3311950.
[28] C. Miller, “Predicting success of energy savings interventions and industry type using smart meter and retrofit data from thousands of non-residential buildings,” in *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, Delft Netherlands, Nov. 2017, pp. 1–4, doi: 10.1145/3137133.3137160.
[29] M. Taheri, P. Rastogi, C. Parry, and A. Wegienka, “Benchmarking Building Energy Consumption Using Efficiency Factors,” Rome, Italy, pp. 3863–3870, doi: 10.26868/25222708.2019.210575.
[30] J. Y. Park, X. Yang, C. Miller, P. Arjunan, and Z. Nagy, “Apples or oranges? Identification of fundamental load shape profiles for benchmarking buildings using a large and diverse dataset,” *Applied Energy*, vol. 236, pp. 1280–1295, Feb. 2019, doi: 10.1016/j.apenergy.2018.12.025.

[31] C. Miller, “What’s in the box?! Towards explainable machine learning applied to non-residential building smart meter classification,” *Energy and Buildings*, vol. 199, pp. 523–536, Sep. 2019, doi: 10.1016/j.enbuild.2019.07.019.
[32] X. Fang, G. Gong, G. Li, L. Chun, W. Li, and P. Peng, “A hybrid deep transfer learning strategy for short term cross-building energy prediction,” *Energy*, vol. 215, p. 119208, Jan. 2021, doi: 10.1016/j.energy.2020.119208.
[33] C. Nichiforov, I. Stancu, I. Stamatescu, and G. Stamatescu, “Information Extraction Approach for Energy Time Series Modelling,” in *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, Oct. 2020, pp. 886–891, doi: 10.1109/ICSTCC50638.2020.9259635.
[34] A. Li, F. Xiao, C. Fan, and M. Hu, “Development of an ANN-based building energy model for information-poor buildings using transfer learning,” *Build. Simul.*, vol. 14, no. 1, pp. 89–101, Feb. 2021, doi: 10.1007/s12273-020-0711-5.
[35] C. Miller *et al.*, “The ASHRAE Great Energy Predictor III competition: Overview and results,” *Science and Technology for the Built Environment*, vol. 26, no. 10, pp. 1427–1447, Nov. 2020, doi: 10.1080/23744731.2020.1795514.
[36] C. Miller *et al.*, “The Building Data Genome Project 2, energy meter data from the ASHRAE Great Energy Predictor III competition,” *Scientific Data*, vol. 7, no. 1, Art. no. 1, Oct. 2020, doi: 10.1038/s41597-020-00712-x.
[37] Z. Wang and T. Hong, “Generating realistic building electrical load profiles through the Generative Adversarial Network (GAN),” *Energy and Buildings*, vol. 224, p. 110299, Oct. 2020, doi: 10.1016/j.enbuild.2020.110299.
[38] T. Hong, Z. Wang, X. Luo, and W. Zhang, “State-of-the-art on research and applications of machine learning in the building life cycle,” *Energy and Buildings*, vol. 212, p. 109831, Apr. 2020, doi: 10.1016/j.enbuild.2020.109831.
[39] V. Földváry Ličina *et al.*, “Development of the ASHRAE Global Thermal Comfort Database II,” *Building and Environment*, vol. 142, pp. 502–512, Sep. 2018, doi: 10.1016/j.buildenv.2018.06.022.
[40] “ASHRAE - Great Energy Predictor III.” https://kaggle.com/c/ashrae-energy-prediction (accessed Jan. 03, 2021).
[41] T. Yiu, “Understanding Random Forest,” *Medium*, Aug. 14, 2019. https://towardsdatascience.com/understanding-random-forest-58381e0602d2 (accessed Jan. 04, 2021).
[42] “1.11. Ensemble methods — scikit-learn 0.24.0 documentation.” https://scikit-learn.org/stable/modules/ensemble.html#gradient-tree-boosting (accessed Jan. 04, 2021).
[43] “Gradient boosting,” *Wikipedia*. Dec. 31, 2020, Accessed: Jan. 04, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Gradient_boosting&oldid=997459645.
[44] “How to explain gradient boosting.” http://explained.ai/gradient-boosting/index.html (accessed Jan. 04, 2021).
[45] D. Freeman, *Statistical Models: Theory and Practice*. Cambridge University Press, 2009.
[46] “A Brief Introduction to XGBoost. Extreme Gradient Boosting with XGBoost! | by Neetika Khandelwal | Towards Data Science.” https://towardsdatascience.com/a-brief-introduction-to-xgboost-3eaee2e3e5d6 (accessed Jan. 04, 2021).
[47] D. J. Hand and R. J. Till, “[No title found],” *Machine Learning*, vol. 45, no. 2, pp. 171–186, 2001, doi: 10.1023/A:1010920819831.

[48] “RMSE: Root Mean Square Error,” *Statistics How To*. https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/ (accessed Jan. 04, 2021).
[49] “R2 score.” [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score.

Appendix A

Instance of raw temporal data set

	timestamp	Office_Dawn	UnivLab_Dianna	Office_Dorian	UnivLab_Ali	UnivLab_Albert	UnivLab_Alaina	UnivLab_Ana	UnivClass_Anika	UnivLab_Annette	Office_Alannah
0	2010-01-01 08:00:00+00:00	313.75	763.550	426.8835	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2010-01-01 09:00:00+00:00	357.25	741.625	428.1145	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2010-01-01 10:00:00+00:00	348.00	726.775	425.0000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2010-01-01 11:00:00+00:00	340.75	715.025	414.0000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2010-01-01 12:00:00+00:00	319.50	703.600	417.0000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
40935	2016-01-01 02:00:00+00:00	NaN	NaN	NaN	81.52	196.36	188.39	106.58	222.79	237.31	17.05
40936	2016-01-01 03:00:00+00:00	NaN	NaN	NaN	80.11	190.36	184.47	108.14	214.99	186.23	19.74
40937	2016-01-01 04:00:00+00:00	NaN	NaN	NaN	NaN	NaN	176.00	27.07	NaN	NaN	6.05
40938	2016-01-01 05:00:00+00:00	NaN	NaN	NaN	77.32	191.37	175.05	95.14	197.71	160.17	18.16
40939	2016-01-01 06:00:00+00:00	NaN	NaN	NaN	76.86	190.10	173.62	78.64	199.46	156.56	16.71

40940 rows × 16 columns

Instance of meta data set

	uid	dataend	datastart	energystarscore	heatingtype	industry	mainheatingtype	numberoffloors	occupants	primaryspaceusage	rating	sqft	
0	PrimClass_Everett	31/12/12 23:00	01/01/12 00:00	NaN	NaN	Education	NaN	NaN	NaN	Primary/Secondary Classroom	NaN	105530.0000	9804
1	UnivClass_Clifford	31/12/15 23:00	01/01/15 00:00	NaN	NaN	Education	NaN	NaN	NaN	College Classroom	NaN	56969.0000	5292
2	Office_Elizabeth	31/12/12 23:00	01/01/12 00:00	NaN	NaN	Commercial Property	NaN	NaN	NaN	Office	NaN	294651.0000	27373
3	Office_Ellie	31/12/12 23:00	01/01/12 00:00	NaN	NaN	Commercial Property	NaN	NaN	NaN	Office	NaN	496517.0000	46127
4	PrimClass_Elisabeth	31/12/12 23:00	01/01/12 00:00	NaN	NaN	Education	NaN	NaN	NaN	Primary/Secondary Classroom	NaN	233062.0000	21652
...	...	...	...	...	...	...	...	...	...	...	...	...	
502	Office_Lane	30/11/15 23:00	01/12/14 00:00	NaN	Heat network	Education	Heat Network	8.0	NaN	Office	NaN	34455.2439	3201
503	Office_Cameron	31/12/15 23:00	01/01/15 00:00	NaN	NaN	Education	NaN	NaN	NaN	Office	NaN	53303.0000	4952
504	UnivLab_Lea	30/11/15 23:00	01/12/14 00:00	NaN	Gas	Education	Gas	6.0	NaN	College Laboratory	NaN	16802.4479	1561
505	UnivLab_Carlos	31/12/15 23:00	01/01/15 00:00	NaN	NaN	Education	NaN	NaN	NaN	College Laboratory	NaN	30143.0000	2800
506	UnivLab_Aoife	31/12/15 23:00	01/01/15 00:00	NaN	NaN	Education	NaN	NaN	NaN	College Laboratory	NaN	261188.0000	24265

507 rows × 19 columns

Instance of meta data set (continued)

›ffloors	occupants	primaryspaceusage	rating	sqft	sqm	subindustry	timezone	yearbuilt	nickname	primaryspaceuse_abbrev	newweatherfilename
NaN	NaN	Primary/Secondary Classroom	NaN	105530.0000	9804.053590	Primary/Secondary School	America/New_York	NaN	Everett	PrimClass	weather12.csv
NaN	NaN	College Classroom	NaN	56969.0000	5292.591007	College/University	America/New_York	1967	Clifford	UnivClass	weather2.csv
NaN	NaN	Office	NaN	294651.0000	27373.961850	Commercial Real Estate	America/Los_Angeles	NaN	Elizabeth	Office	weather22.csv
NaN	NaN	Office	NaN	496517.0000	46127.918850	Bank/Financial Services	America/Los_Angeles	NaN	Ellie	Office	weather28.csv
NaN	NaN	Primary/Secondary Classroom	NaN	233062.0000	21652.158990	Primary/Secondary School	America/New_York	NaN	Elisabeth	PrimClass	weather23.csv
...	...	...	...	...	...	...	...	...	...	...	...
8.0	NaN	Office	NaN	34455.2439	3201.000000	College/University	Europe/London	1907	Lane	Office	weather5.csv
NaN	NaN	Office	NaN	53303.0000	4952.008609	College/University	America/New_York	1981	Cameron	Office	weather2.csv
6.0	NaN	College Laboratory	NaN	16802.4479	1561.000000	College/University	Europe/London	1995	Lea	UnivLab	weather5.csv
NaN	NaN	College Laboratory	NaN	30143.0000	2800.375129	College/University	America/New_York	1951	Carlos	UnivLab	weather2.csv
NaN	NaN	College Laboratory	NaN	261188.0000	24265.148760	College/University	America/Phoenix	NaN	Aoife	UnivLab	weather0.csv

Appendix B

Confusion matrices for the two classification models.

```
array([[ 23844,      5,      0,      2,      0,      0,      0,      0,
             0],
       [     0, 231243,   9499,   2751,  10571,   6148,      0,      0,
             0],
       [     0,   1965, 205622,      0,   1108,    218,     25,      0,
             0],
       [     0,  11328,   2833, 150634,   3849,   6285,      0,      0,
             0],
       [     0,   5791,   2254,   6482, 130750,   4278,      0,      0,
             0],
       [     0,   8958,    873,   7730,   5510, 182778,      0,      0,
             0],
       [     0,      0,   1412,      0,      0,    173,  51466,      0,
             0],
       [     0,      0,      0,      0,      0,      0,      0,  12348,
             0],
       [     0,      0,      0,      0,      0,      0,      0,      0,
          2084]])
```

Figure 26 Confusion matrix of the Gradient Boosting classifier

```
array([[     0,   6588,   3645,      0,      0,  13618,      0,      0,
             0],
       [     0, 211871,  33476,      0,      0,  14865,      0,      0,
             0],
       [     0,  39793, 168263,      0,      0,    882,      0,      0,
             0],
       [     0, 135832,  25945,      0,      0,  13152,      0,      0,
             0],
       [     0, 111945,  19628,      0,      0,  17982,      0,      0,
             0],
       [     0, 112462,  22631,      0,      0,  70756,      0,      0,
             0],
       [     0,      0,  46461,      0,      0,    222,   6368,      0,
             0],
       [     0,      0,  12348,      0,      0,      0,      0,      0,
             0],
       [     0,      0,   2084,      0,      0,      0,      0,      0,
             0]])
```

Figure 27 Confusion matrix of the Random Forest classifier

Appendix C

The Python scripts that were deployed are shown below.

```
# Setting up
# Import libraries
from pathlib import Path
import numpy as np
import time
import pickle
import pandas as pd
import matplotlib.pyplot as plt

# Directories and Paths
DIR_WORK = Path(....)
DIR_DATA = Path(....)

TS_DATA = "temp_open_utc.csv"
BUILDING_DATA = "meta_open.csv"



# ETL
#Timeseries Data

#Load data
ts_data_raw = pd.read_csv(Path(DIR_DATA, TS_DATA))\
ts_data_raw.tail(2)

# Check & Convert datatypes
ts_data_raw.dtypes

ts_data_raw["timestamp"] = pd.to_datetime(ts_data_raw["timestamp"])
print(ts_data_raw.dtypes)
ts_data_raw.tail(2)

# Visualise the timeseries
# Convert the timestamp column to DatetimeIndex and plot the graph
ts_data_raw.set_index(pd.DatetimeIndex(ts_data_raw["timestamp"]))
.drop("timestamp", axis=1)
.plot(figsize=(20,10), legend=False, title='Hourly building energy consumption profiles in kWh')

# Unpack & transform the dataframe
ts_data = pd.melt(ts_data_raw, value_vars=ts_data_raw.columns.values[1:], id_vars=['timestamp'])
ts_data
```

```
ts_clean = ts_data.copy()

# Drop nan rows
ts_clean = ts_clean.dropna()

# Extract the ordinal day of the year
ts_clean["day_of_year"] = ts_clean["timestamp"].dt.dayofyear

# Extract the hour from the timestamp
ts_clean["hour"] = ts_clean["timestamp"].dt.hour

# Rename columns
ts_clean = ts_clean.rename(columns={"variable": "uid", "value": "load"})

# Drop irrelevant columns and reset index
ts_clean = ts_clean.drop("timestamp", axis=1).reset_index(drop=True)

print(ts_clean.shape)
ts_clean.tail(6)

# Building data
building_data_raw = pd.read_csv(Path(DIR_DATA, BUILDING_DATA))
building_data_raw

building_data_raw.dtypes

# Keep relevant columns and creating derivative features
building_data = building_data_raw[["uid", "industry", "primaryspaceuse_abbrev", "sqm", "timezone"]]

# Create 2 new features from Timezone
building_data[["continent", "city"]] = building_data["timezone"].str.split("/", expand=True)

# Drop Timezone as it is irrelevant
building_data = building_data.drop("timezone", axis=1)

# Rename columns for consistency
building_data = building_data.rename(columns={"primaryspaceuse_abbrev": "usespace"})
building_data

# Primary dataframe
primary = ts_clean.merge(building_data, on="uid")
primary.head()

# Models
# Prepare data for model
df = primary.copy()
df.head()
```

```
df.info()

# Drop the uid
df = df.drop("uid", axis=1)

# Convert categorical to dummies
df = pd.get_dummies(df)

# Modelling
# Split data in Train, Test
# Primary dataset

# Import library
from sklearn.model_selection import train_test_split

# Train/test split
x = df.loc[:, df.columns != 'load']
x.head()

y = df.loc[:, "load"]

# CASE 1 -Use of shuffle in split - random state=10 - train/test 75/25 defaults
x_train, x_test, y_train, y_test = train_test_split(x , y, random_state=10, shuffle=True, test_size=0.25)

# CASE 2 -Use of shuffle in split - random state=10 - train/test 50/50
x_train, x_test, y_train, y_test = train_test_split(x , y, random_state=10, shuffle=True, test_size=0.50)

# Regression
# Import libraries
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
from math import sqrt
import xgboost as xgb
import time

# Random forest
# Case 1
start_time = time.time()

rf_reg = RandomForestRegressor(max_depth=2, random_state=1, n_estimators=100, n_jobs=-1)
rf_reg.fit(x_train, y_train)

end_time = time.time()
```

```
print("--- %s seconds ---" % (end_time - start_time))

# Predicted values
y_pred = rf_reg.predict(x)

# Print metrics of the model 75/25 split
# R2
print("Train score:", rf_reg.score(x_train, y_train))
print("Test score:", rf_reg.score(x_test, y_test))

# MSE
rf_reg_mse = mean_squared_error(y, y_pred)
print("MSE:", rf_reg_mse)

# RMSE
rf_reg_rmse = sqrt(rf_reg_mse)
print("RMSE:", rf_reg_rmse)

# Case 2 - Grid Search - 5 FOLD

#Split data in train and test set
x_train, x_test, y_train, y_test = train_test_split(x , y, random_state=10, shuffle=True, test_size=0.50)

start_time = time.time()

# Set the parameters
max_depths = (2,3)
n_estimators = (100)
tuned_parameters = [{'max_depth': max_depths}]
n_folds = 5

# Random Forest Regressor
rf_reg = RandomForestRegressor(random_state=1, n_jobs=-1)

RFREG = GridSearchCV(rf_reg, param_grid=tuned_parameters, cv=5, verbose=10)
RFREG.fit(x_train, y_train)
print('Best parameters: ', RFREG.best_params_)
print('Average score: ', RFREG.best_score_)
print(RFREG.cv_results_['mean_test_score'])

# Predict
y_true, y_pred = y_test, RFREG.predict(x)

end_time = time.time()

print("--- %s seconds ---" % (end_time - start_time))
```

```
#Print the scores
# R2
print("Train score:", RFREG.score(x_train, y_train))
print("Test score:", RFREG.score(x_test, y_test))
# MSE
RFREG_mse = mean_squared_error(y, y_pred)
print("MSE:", RFREG_mse)
# RMSE
RFREG_rmse = sqrt(RFREG_mse)
print("RMSE:", RFREG_rmse)

# Linear Regression
# Case 1
start_time = time.time()

lr_reg = LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=-1)
lr_reg.fit(x_train, y_train)

end_time = time.time()

print("--- %s seconds ---" % (end_time - start_time))

# Predicted values
y_pred = lr_reg.predict(x)

# Print metrics of the model
# R2
print("Train score:", lr_reg.score(x_train, y_train))
print("Test score:", lr_reg.score(x_test, y_test))

# MSE
lr_reg_mse = mean_squared_error(y, y_pred)
print("MSE:", lr_reg_mse)

# RMSE
lr_reg_rmse = sqrt(lr_reg_mse)
print("RMSE:", lr_reg_rmse)

# Case 2 - GridSearchCV - 5 fold
#Split data in train and test set
x_train, x_test, y_train, y_test = train_test_split(x , y, random_state=10, shuffle=True, test_size=0.50)

start_time = time.time()

# Set the parameters
max_depths = (3, 5, 7)
```

```
tuned_parameters = [{'max_depth': max_depths}]
n_folds = 5

lr_reg = LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=-1)
lr_reg.fit(x_train, y_train)

LRREG = LinearRegression(random_state = 1, n_jobs=-1)

LRREG = GridSearchCV(LRREG, param_grid=tuned_parameters, cv=5, verbose=10)
LRREG.fit(x_train, y_train)
print('Best parameters: ', LRREG.best_params_)
print('Average score: ', LRREG.best_score_)
print(LRREG.cv_results_['mean_test_score'])

# Predict
y_true, y_pred = y_test, LRREG.predict(x)

end_time = time.time()

print("--- %s seconds ---" % (end_time - start_time))

#Print the score
# R2
print("Train score:", XGBREG.score(x_train, y_train))
print("Test score:", XGBREG.score(x_test, y_test))
# MSE
XGBREG_mse = mean_squared_error(y, y_pred)
print("MSE:", XGBREG_mse)
# RMSE
XGBREG_rmse = sqrt(XGBREG_mse)
print("RMSE:", XGBREG_rmse)

# Gradient Boosting
# Case 1

start_time = time.time()

gb_reg = GradientBoostingRegressor(random_state=0)
gb_reg.fit(x_train, y_train)

end_time = time.time()

print("--- %s seconds ---" % (end_time - start_time))

# Predicted values
y_pred = gb_reg.predict(x)
```

```
# Print metrics of the model
# R2
print("Train score:", gb_reg.score(x_train, y_train))
print("Test score:", gb_reg.score(x_test, y_test))

# MSE
gb_reg_mse = mean_squared_error(y, y_pred)
print("MSE:", gb_reg_mse)

# RMSE
gb_reg_rmse = sqrt(gb_reg_mse)
print("RMSE:", gb_reg_rmse)

# Case 2 - GridSearchCV - 5 FOLD
#Split data in train and test set
x_train, x_test, y_train, y_test = train_test_split(x , y, random_state=10, shuffle=True, test_size=0.50)

start_time = time.time()

# Set the parameters by cross-validation
max_depths = (3,5)
tuned_parameters = [{'max_depth': max_depths}]
#n_folds = 5

GBREG = GradientBoostingRegressor(max_depth=5, random_state = 1)

GBREG = GridSearchCV(GBREG, param_grid=tuned_parameters, cv=5, verbose=10)
GBREG.fit(x_train, y_train)
print('Best parameters: ', GBREG.best_params_)
print('Average score: ', GBREG.best_score_)
print(GBREG.cv_results_['mean_test_score'])

# Predict
y_true, y_pred = y_test, GBREG.predict(x)

end_time = time.time()

print("--- %s seconds ---" % (end_time - start_time))

#Print the score
# R2
print("Train score:", GBREG.score(x_train, y_train))
print("Test score:", GBREG.score(x_test, y_test))
# MSE
GBREG_mse = mean_squared_error(y, y_pred)
print("MSE:", GBREG_mse)
# RMSE
```

```
GBREG_rmse = sqrt(GBREG_mse)
print("RMSE:", GBREG_rmse)

# XgBoost
# Case 1

start_time = time.time()

xgb_reg = xgb.XGBRegressor(n_jobs=1)
xgb_reg.fit(x_train, y_train)

end_time = time.time()

print("--- %s seconds ---" % (end_time - start_time))

# Predicted values
y_pred = xgb_reg.predict(x)

# Print metrics of the model
# R2
print("Train score:", xgb_reg.score(x_train, y_train))
print("Test score:", xgb_reg.score(x_test, y_test))

# MSE
xgb_reg_mse = mean_squared_error(y, y_pred)
print("MSE:", xgb_reg_mse)

# RMSE
xgb_reg_rmse = sqrt(xgb_reg_mse)
print("RMSE:", xgb_reg_rmse)

# Case 2 - Grid Search - 5 FOLD
#Split data in train and test set
x_train, x_test, y_train, y_test = train_test_split(x , y, random_state=10, shuffle=True, test_size=0.50)

start_time = time.time()

# Set the parameters
boosters = ('gblinear', 'gbtree')
tuned_parameters = [{'booster': boosters}]

XGBREG = xgb.XGBRegressor(random_state = 1, n_jobs=1)

XGBREG = GridSearchCV(XGBREG, param_grid=tuned_parameters, cv=5, verbose=10)
XGBREG.fit(x_train, y_train)
print('Best parameters: ', XGBREG.best_params_)
print('Average score: ', XGBREG.best_score_)
```

```
print(XGBREG.cv_results_['mean_test_score'])

# Predict
y_true, y_pred = y_test, XGBREG.predict(x)

end_time = time.time()

print("--- %s seconds ---" % (end_time - start_time))

#Print the score
# R2
print("Train score:", XGBREG.score(x_train, y_train))
print("Test score:", XGBREG.score(x_test, y_test))
# MSE
XGBREG_mse = mean_squared_error(y, y_pred)
print("MSE:", XGBREG_mse)
# RMSE
XGBREG_rmse = sqrt(XGBREG_mse)
print("RMSE:", XGBREG_rmse)

# Classification
# Preprocess data
# Import libraries from regression part too
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Repeat the first case preprocess steps to transform the dataset for classification
ts_data_raw.head()

ts_data_raw["timestamp"] = pd.to_datetime(ts_data_raw["timestamp"])
ts_data = pd.melt(ts_data_raw, value_vars=ts_data_raw.columns.values[1:], id_vars=['timestamp'])
ts_clean = ts_data.copy()

# Drop nan rows
ts_clean = ts_clean.dropna()
# Extract the ordinal day of the year
ts_clean["day_of_year"] = ts_clean["timestamp"].dt.dayofyear
# Extract the hour from the timestamp
ts_clean["hour"] = ts_clean["timestamp"].dt.hour
# Rename columns
ts_clean = ts_clean.rename(columns={"variable": "uid", "value": "load"})
# Drop irrelevant columns and reset index
ts_clean = ts_clean.drop("timestamp", axis=1).reset_index(drop=True)
```

```
#building_data_raw = pd.read_csv(Path(DIR_DATA, BUILDING_DATA))
building_data = building_data_raw[["uid", "industry", "primaryspaceuse_abbrev", "sqm", "timezone"]]
# Create 2 new features from Timezone
building_data[["continent", "city"]] = building_data["timezone"].str.split("/", expand=True)
# Drop Timezone as it is irrelevant
building_data = building_data.drop("timezone", axis=1)
# Rename columns for consistency
building_data = building_data.rename(columns={"primaryspaceuse_abbrev": "usespace"})

primary = ts_clean.merge(building_data, on="uid")

df = primary.copy()
df = df.drop("uid", axis=1)
df

# Create the 'label' for classification
df["label"] = df["industry"] + "-" + df["usespace"]
# Drop the two features after the concatenation
df = df.drop(["industry", "usespace"], axis=1)
# Get dummy values for the features left
df = pd.get_dummies(df, columns=["continent", "city"])
df.head()

# Use of Lable Encoder for the transformation of the label from string to numbered categories
df['label'] = LabelEncoder().fit_transform(df['label'])
df.head()

# Count the classes and check the balance
df["label"].value_counts()

# Split in train and test
x = df.loc[:, df.columns != 'label']
y = df.loc[:, "label"]

# Use of shuffle in split and flag stratify to YES because of imbalanced dataset
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=10, shuffle=True, stratify=y)

# Gradient Boosting
# Case 1
start_time = time.time()

gb_class = GradientBoostingClassifier(random_state=0)
gb_class.fit(x_train, y_train)
end_time = time.time()

print("--- %s seconds ---" % (end_time - start_time))
```

```
print("Train score:", gb_class.score(x_train, y_train))
print("Test score:", gb_class.score(x_test, y_test))

y_pred = gb_class.predict(x_test)

# Classification report
print(classification_report(y_test, y_pred))

# Confusion matrix
confusion_matrix(y_test, y_pred)

# Random Forest
# Case 1
start_time = time.time()
rf_class = RandomForestClassifier(max_depth = 3, random_state=0, n_jobs=-1)
rf_class.fit(x_train, y_train)
end_time = time.time()

print("--- %s seconds ---" % (end_time - start_time))

print("Train score:", rf_class.score(x_train, y_train))
print("Test score:", rf_class.score(x_test, y_test))

y_pred = rf_class.predict(x_test)

# Classification report
print(classification_report(y_test, y_pred))

# Confusion matrix
confusion_matrix(y_test, y_pred)
```