UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS TRINDADE
ENGENHARIA DE PRODUÇÃO CIVIL
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS

Lorrany da Silva Mendes

DATA-DRIVEN PREDICTION OF ENERGY EFFICIENCY IN BUILDING RETROFIT

Florianópolis
2024

Lorrany da Silva Mendes

DATA-DRIVEN PREDICTION OF ENERGY EFFICIENCY IN BUILDING RETROFIT

Trabalho de Conclusão de Curso submetida ao Departamento de Engenharia de Produção e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de Grau de Engenheiro Civil com habilitação em Engenharia de Produção.
Supervisor:: Prof. Mauricio Uriona Maldonado, Dr.

Florianópolis
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Mendes, Lorrany da Silva
   DATA-DRIVEN PREDICTION OF ENERGY EFFICIENCY IN BUILDING RETROFIT / Lorrany da Silva Mendes ; orientador, Maurício Uriona Maldonado, 2024.
   166 p.

   Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Produção Civil, Florianópolis, 2024.

   Inclui referências.

   1. Engenharia de Produção Civil. 2. Machine Learning. 3. Eficiência Energética . 4. Retrofit. 5. Modelo de predição. I. Maldonado, Maurício Uriona. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Produção Civil. III. Título.

Lorrany da Silva Mendes

DATA-DRIVEN PREDICTION OF ENERGY EFFICIENCY IN BUILDING RETROFIT

O presente trabalho em nível de bacharelado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Grau de Engenheiro Civil com habilitação em Engenharia de Produção.

ICP-Edu
Documento assinado digitalmente
Monica Maria Mendes Luna
Data: 23/12/2024 09:28:22-0300
CPF: ***.104.993-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Monica Mendes Luna, Dra.
Coordenador do Programa

ICP-Edu
Documento assinado digitalmente
Mauricio Uriona Maldonado
Data: 23/12/2024 09:10:12-0300
CPF: ***.972.939-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Mauricio Uriona Maldonado, Dr.
Supervisor:

Florianópolis, 14 de dezembro de 2024.

This project is dedicated to my parents.

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to God for His Absolute presence and guIding in my life, which occurs to be in a small place and brief time lapse in this vast Universe.

I thank my parents, Michelle and Alcionei, for their love and encouragement. I thank my sister, Gabrielly for her sweet friendship, and my brother, Kauã, for his great humor. And finally, my supportive Antoine Convert for his constant care through this whole project.

I thank Luc Welfringer, Anthony Roux and Damien Racle for their support.

I thank professor, Mauricio Maldonado for his insights and revisions. I thank professor Ana Paula Melo and Professor Enzo Frazzon, for accepting being part of board of examiners.

I also thank my former LabEEE colleagues, Rayner, Letícia, Marcelo, Tiago, Larissa, Rafael, Leonardo, Leandra, Matheus(es), Greici, Renata, Andrea, Lucas, and professor Ana Paula and Lamberts for their support and all those who made my LabEEE journey so agreeable somehow.

I also thank my helpful colleagues that made this journey smoother, Luísa, Carolina and Maria Eduarda. To my colleagues of christian associations, I thank you all for being such a refreshment in the middle of the week.

To all those who crossed my way with some help, my most sincere thank you!

"And let us not grow weary while doing good, for in due season we shall reap if we do not lose heart."
(Galatians 6:9)

ABSTRACT

As auditorias energérticas desempenham um papel crucial na identificação de oportunidades de melhoria da eficiência energética em edifícios por meio de retrofit. Este projeto, conduzido dentro de uma empresa francesa especializada em auditorias energéticas para edifícios terciários, concentra-se no desenvolvimento de um método confiável e econômico para calcular a economia de energia em retrofits. Isso é essencial para melhorar a eficiência energética, reduzir as emissões de CO2 e alinhar-se à Estratégia Nacional de Baixo Carbono. Os métodos tradicionais para calcular melhorias na eficiência energética, como regressão estática e simulação térmica, têm limitações. A regressão estática simplifica demais os cálculos, enquanto a simulação térmica é muito cara. Para resolver isso, o projeto explora o aprendizado de máquina como uma solução orientada por dados capaz de fornecer estimativas precisas e rápidas. Com base em pesquisas anteriores, este projeto se concentra no desenvolvimento de modelos independentes de aprendizado de máquina para prever a economia de energia em vários retrofits. Essa abordagem reduz o número de entradas necessárias para cada modelo, melhorando a usabilidade para os funcionários sem comprometer o desempenho. Doze modelos independentes foram criados usando dados de duas bases de dados da empresa. O tratamento de dados e a engenharia de recursos foram empregados para preparar os dados para a apliacação do aprendizado de máquina. O estudo usou principalmente modelos de Gradient Boosting Machine (GBM) com transformação de regressão. Outros modelos, incluindo Redes Neurais Artificiais, Árvores de Decisão e Florestas Aleatórias, também foram explorados. Ajuste de hiperparâmetros, validação cruzada e múltiplos estados aleatórios foram implementados para todos os modelos, com o GBM demonstrando desempenho superior. Técnicas de clusterização e remoção de outliers foram aplicadas aos modelos GBM para retrofits específicos, resultando em melhorias de desempenho. Os resultados alcançados são considerados aceitáveis para aplicações do mundo real dentro da empresa. Os modelos foram integrados a um aplicativo da web, https://arcs-sevaia.streamlit.app/ , fornecendo aos funcionários da empresa uma ferramenta amigável para avaliar potenciais economias de energia de várias opções de retrofit.

Palavras-chave: Aprendizado de Máquina; Eficiência Energética; Retrofit; Modelo de predição.

ABSTRACT

Energy audits play a crucial role in identifying energy efficiency improvement opportunities in buildings through retrofitting. This project, conducted within a French company specializing in energy audits for tertiary buildings, focuses on developing a reliable and cost-effective method for calculating energy savings from retrofits. This is essential for improving energy efficiency, reducing CO2 emissions, and aligning with the National Low Carbon Strategy. Traditional methods for calculating energy efficiency improvements, such as static regression and thermal simulation, have limitations. Static regression oversimplifies calculations, while thermal simulation is very expensive. To address this, the project explores machine learning as a data-driven solution capable of providing accurate and rapid estimations. Building upon previous research, this project focuses on developing independent machine learning models to predict energy savings for various retrofits. This approach reduces the number of inputs required for each model, enhancing usability for employees without compromising performance. Twelve independent models were created using data from two company databases. Data treatment and feature engineering were employed to prepare the data for machine learning applications. The study primarily utilized Gradient Boosting Machine (GBM) models with regression transformation. Other models, including Artificial Neural Networks, Decision Trees, and Random Forests, were also explored. Hyperparameter tuning, cross-validation, and multiple random states were implemented for all models, with GBM demonstrating superior performance. Clusterisation and outlier removal techniques were applied to GBM models for specific retrofits, resulting in performance improvements. The results achieved are deemed acceptable for real-world applications within the company. The models have been integrated into a web application, https://arcs-sevaia.streamlit.app/, providing the company's employees with a user-friendly tool to assess potential energy savings from various retrofitting options.

Keywords: Machine Learning, Energy Efficiency, Retrofit, Prediction Model.

LIST OF FIGURES

CONTENTS

1 INTRODUCTION

The energy transition is a continuing process requiring long-term energy strategies and planning, with a country-tailored focus on applying appropriated energy technologies to reach net-zero emissions (UNDP, 2024). Particularly in France, the Energy Transition politics have put in place The Eco-Energy Tertiary Scheme (Décret tertiaire), which requires tertiary buildings larger than 1000 m² to reduce their final energy consumption by 40% by 2030, 50% by 2040 and 60% by 2050 (ADEME, 2020).

To implement these actions in the existing buildings, there are decision-support studies (pre-diagnoses, energy audits, feasibility studies) that aim to enable managers and project owners to identify energy-saving opportunities and quickly implement economically viable energy consumption control measures by considering the potential dynamics of energy price evolution over medium term (ADEME, 2020).

In particular, the energy audits, which are conducted through a visit to the building to collect data, should enable the project owner to make informed decisions by providing estimations of the energy efficiency to be acquired through building updates. Due to the complexity of physical phenomena, obtaining reliable results of these energy efficiency estimations can be a complex problem, leading to the research for rapid but reliable methods to make these predictions.

1.1 RESEARCH PROBLEM

The building sector is a critical component of the global economy, not only as a driver of economic growth and employment but also account for around 30% of global energy consumption and nearly 40% of CO2 emissions (IEA, 2024a). Consequently, reducing energy usage in buildings is central to mitigate climate change.

Approximately 75% of the buildings expected to exist in 2050 have already been constructed (EEA, 2024). Therefore, updating existing structures in order to improve their energy efficiency and reducing their environmental impact is necessary.

> Within the context of the built environment, this process is called retrofit, [...] used to imply substantive physical changes to a building (e.g. mitigation activities to improve energy efficiency), and often linked to the concept of "adaptation" (i.e. intervention to adjust, reuse or upgrade a building to suit new conditions or requirements (DIXON, 2014).

Retrofit might include a variety of improvements, like improving insulation, modernizing HVAC systems, or improving lighting. The impact of each update on energy consumption varies depending on a number of interrelated factors, including the building's construction year, usage patterns, location, etc.

Despite its evident potential, retrofit solutions are challenging to implement adequately due to difficulties in planning, estimating, and quantifying energy efficiency.

To be able to accurately predict the effects of these modifications is crucial to correctly calculate the return on investment and environmental benefits of retrofit, allowing to prioritize the ones with the best cost-benefits.

There are many different strategies to calculate the energy efficiency acquired by a retrofit. The most basic ones consist of making predictive analysis based on different forms of regressions. However, mathematical modeling can easily become unfeasible due to constraints of time, computational power and most of all, the complexity of the physical phenomenons.

Another solutions are static point estimation calculations, linear or multiple regressions and the main method used nowadays is thermodynamic simulations. Recently, the use of artificial intelligence through machine learning algorithms is also increasing.

These methods have different levels of problem and application depending on the context in which they are inserted. According to Versage (2015), dynamic simulations are the most advanced methods for predicting the energy performance of buildings, being a popular tool for analyzing potential energy savings by modeling the physical interactions within a building. While these simulations can be highly accurate, they are computationally demanding and often require specialized knowledge to create a satisfactory model and interpret the results, making them time-intensive and costly.

In the other hand, Versage (2015) states that statistical methods for sample-based inference functions are faster and simpler to use. However these models often lack the necessary precision to capture the complex relationships between retrofit interventions and energy outcomes. These approaches might oversimplify the relationships, producing inaccurate projections that could lead to resource misallocation and missed savings opportunities.

Particularly, the company to which this project is addressed, faces this problem daily as audits rely on static estimations and the opinion of experts, getting to a point of dependence upon the employees. Their knowledge may be carried as they depart from the company, or even not be acquired as they arrive into the company. Also, it can be highly simplified and not standardized at a company level.

Regarding, thermal simulation, they are resource and time-consuming, requiring training, experience and very detailed information, that are frequently not available even after visiting the building and collecting data, also it is not very popular among clients, as it is expensive. Therefore, the company looks for a fast and reliable method able to improve their energy audit process by estimating the energy efficiency of their retrofit suggestions, which raises the following problematic: "How to predict the energy efficiency impact of retrofit on third sector buildings audited by the company?"

1.2 OBJECTIVE

1.2.1 General Objective

The main objective is to improve the energy audit process by creating an Energy savings calculation prediction tool.

1.2.2 Specific Objectives

The specific goals are:

a) Build machine learning models to predict the energy efficiency of 12 retrofit actions;

b) Execute permutation feature importance analysis;

c) Make the hyper-parameters tuning;

d) Simplify the inputs to the essentials.

1.3 JUSTIFICATION

The field of machine learning is sufficiently young that it is still expanding at an accelerating pace, lying at the crossroads of computer science and statistics, and at the core of artificial intelligence (AI) and data science. In just the last five or ten years, machine learning has become a critical way, arguably the most important way, most parts of AI are done. (BROWN, 2021).

Recent progress in ML has been driven both by the development of new learning algorithms theory, and by the ongoing explosion in the availability of vast amount of data and low-cost computation (PUGLIESE; REGONDI; MARINI, 2021), but also by the development of new theories about learning algorithms and the continued explosion in the availability of big data and low-cost computing.

Machine learning identifies correlations and makes predictions where humans would not be able to. It is adaptable and non-parametric, and can more successfully deal with observations that include complex phenomena, which are features of real and complex data (DAVE; DUTTA, 2012).

In the field of buildings in France, alternative energy, environmental or economic scenarios proposed by designers are increasingly often modeled throughout the life cycle. Beyond that, scenarios to model future climate are also being used to create public policies, which shows the importance of having good models to buildings, as they are part of a common public interest, which is also reliable on data.

Bonte, Thellier, and Lartigue (2014) developed a method based on an artificial intelligence algorithm to model occupant behavior, taking into account individual preferences such as set temperature, blinds, windows, lighting and dress code.

Again, Paudel (2016) used artificial intelligence to estimate the load curve of its network based on weather forecasts for low-consumption buildings. He chose the method because of the complex interactions between the outside temperature, solar radiation, the inertia of the building, the use and control of the heat supply.

In Canada, a study carried out by Le Cam, Daoud, and Zmeureanu (2016), presented a model for predicting energy demand over the next twenty-four hours by non-parametric regression. A genetic algorithm is then used to optimize the size of the variation range of each parameter characteristic of the similarity of conditions so as to minimize the error in predicting demand over a week.

In Brazil, machine learning is used in the certification and energy regulation of buildings through the construction of powerful models capable of predicting thermal consumption for several climates and building typologies. According to Souza (2022), the model input parameters are described based on geometry, constructive aspects and climatic factors. As a result, the model predicts the cooling consumption density for the building and is constantly improved.

These studies are examples of the possible of the viable use of machine learning to create building energy predictions models, and their capacity to contribute to the design of more sustainable and energy-efficient buildings. Each method has its advantages and disadvantages and as the field is constantly evolving, different studies emerge in order to solve specific demands and needs of the field.

As shown, machine learning is able to leverage the strengths of both thermodynamic simulations and static regression models to improve the accuracy and efficiency of energy prediction in retrofit projects. Accurate predictions are made possible without the high processing costs of comprehensive simulations by combining these benefits by discovering connections and understanding patterns from massive data sets that conventional approaches could overlook.

1.4 PROJECT STRUCTURE

The structure of this project is divided in five chapters. The first one presents the research problem, goals and subject justification. The second one presents the literature review, including the context of energy transition and climate change, the key concepts and also a section presenting some of the similar studies literature.

The third chapter is about the methodology, which is divided in two main parts: data treatment and the construction of predictions models. Finally, the fourth chapter presents the results, analyzing both the databases and the performance of the machine learning models created. The fifth chapter presents the conclusion.

2 LITERATURE REVIEW

2.1 ENERGY TRANSITION

Given the severity of the threat posed by anthropogenic climate change, which is driven in large part by fossil fuel combustion, it is becoming widely recognized that societies need a transition in how they produce and consume energy. The energy transition aims to prepare for the post-oil era and establish a resilient and sustainable energy model in the face of challenges in energy supply, price fluctuations, resource depletion, and environmental protection imperatives (MTE, 2017).

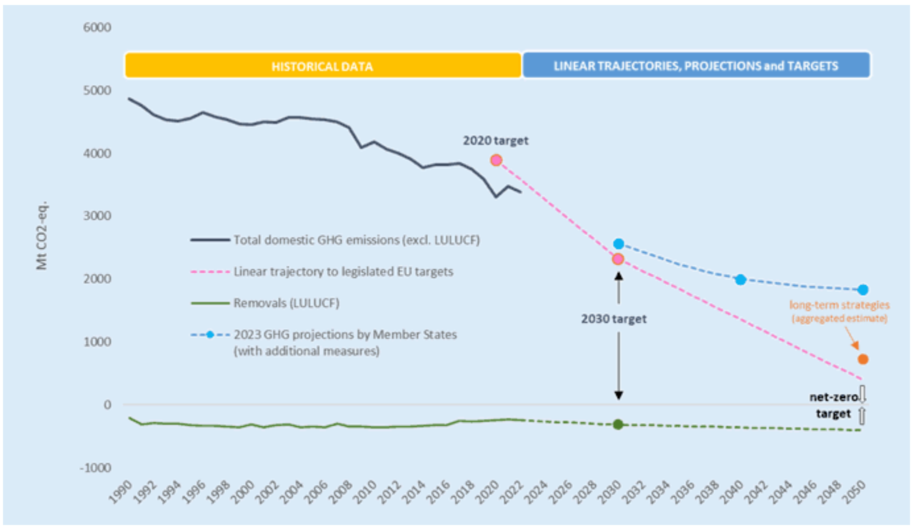
Europe is invested in being the main leader globally in this front, setting the goal of achieving carbon neutrality by 2050. Carbon neutrality implies a radical change in energy production systems, transformation and consumption, notably the challenge of replacing hydrocarbons with decarbonized energy sources (MONTAIGNE, 2021).

These measures were defined through The European Green Deal in 2020 after the Paris Agreement of 2015, which provided a durable framework guiding the global effort for decades to come, marking the beginning of a shift towards net-zero emissions.

The European Green Deal was put in the form of law called European Climate Law to reach climate neutrality by 2050, signing a commitment to negative emissions after 2050, beyond the establishment of European Scientific Advisory Board on Climate Change, that will provide scientific advice and means to climate change adaptation.

According to the 2023 report as the Figure 1 below, the EU has steadily decreased its greenhouse gas emissions since 1990, reaching a total –32.5% in 2022. COVID lockdown measures in 2020 caused an unprecedented fall in emissions, followed by a strong rebound in 2021, with subsequent decrease at a slow rate, turning EU not on track to reach its 2030 objective of carbon removal.

Figure 1 – 2023 Survey of emissions



Source: European Commission (2023)

At the French level, the country has put into place The Energy Transition for Green Growth Act (LTECV) and the National Low Carbon Strategy (SNBC), which determines and formalizes the necessary measures to achieve carbon neutrality by 2050, to more effectively contribute to the fight against climate change and the preservation of the environment, as well as to enhance its energy independence while providing its businesses and citizens with access to energy at a competitive cost (MTE, 2017).

One of the ten goals of SNBC is to achieve an energy performance level in line with "low-energy buildings" standards for the entire housing stock by 2050. The reason behind it is the high significance of the building sector energy consumption both at global and national scales. According to the International Energy Agency, the operations of buildings account for 30% of global final energy consumption and 26% of global energy-related emissions, 8% being direct emissions in buildings and 18% indirect emissions from the production of electricity and heat used in buildings (IEA, 2023). In France, it represents 44% of the energy consumption and more than 123 million tonnes of CO2 emissions each year (MTE, 2021).

Particularly, the tertiary buildings represent one quarter of all the French existing structures, more than 940 million square meters, which accounts for one third of the total energy consumption and greenhouse gas emissions from all buildings (OPERA, 2023). In this context, the Eco-Energy Tertiary Scheme (Décret tertiaire) whose origin is the Grenelle II Law in 2010, later incorporated in 2017 into the Energy Transition and Green Growth Act and then in the ELAN law in 2018 (VERTIGO, 2023), requires tertiary buildings larger than 1000 m^2 to reduce their final energy consumption by 40% by 2030, 50% by 2040 and 60% by 2050 (ADEME, 2020).

> The impact study guided by ADEME (2020) shows that around 68% of all tertiary buildings in France are concerned by the decree, being included all the domains of the third sector, with very few exceptions, which are temporary constructions (temporary building permits), places of worship, and activities for operational purposes related to defense, civil security, or domestic security.

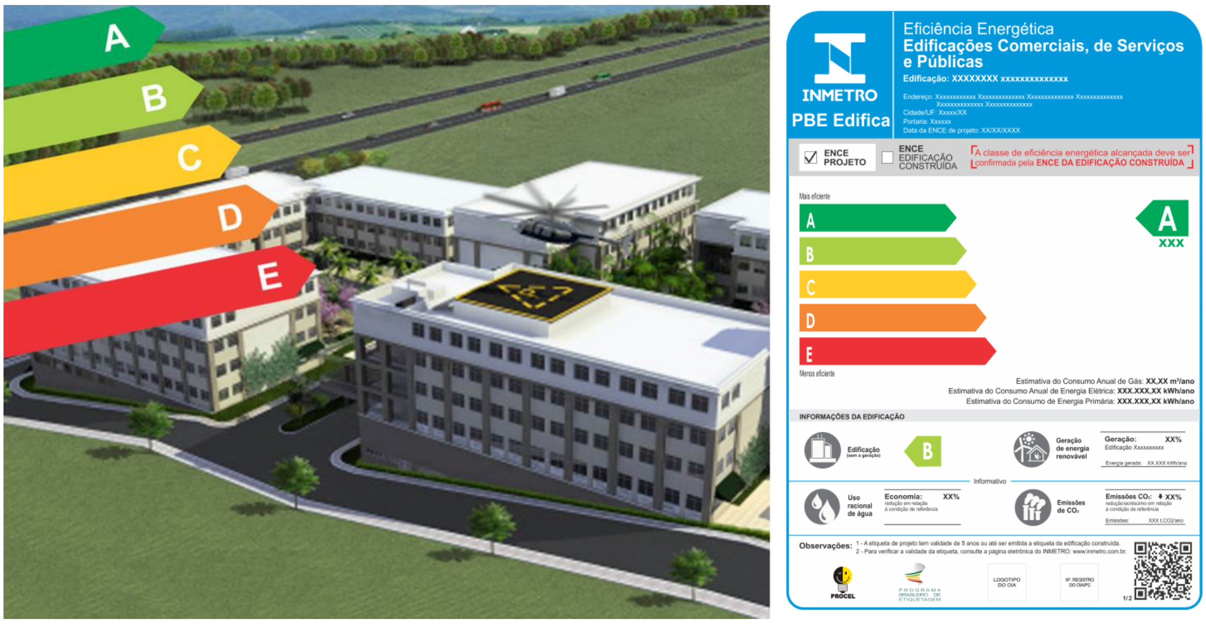
In order to achieve the Decree goals, there are five levers of actions: improvement of building energy performance, installation of efficient equipment, optimization of equipment operation, adaptation of spaces for energy efficiency and encouraging sustainable occupant behavior.

To implement these actions, pre-diagnoses, energy audits, feasibility studies are used. In this project, calculate the energy savings, a representative indicator of energy efficiency, to building audits is the main focus.

2.2 ENERGY EFFICIENCY

Energy efficiency is the use of less energy to perform the same task or produce the same result (U.S.GOVERNMENT, 2024). In the context of buildings, it is based on the establishment of standards for evaluating and classifying buildings in terms of energy performance, and it is important to note that the benefits of energy efficiency go beyond reducing energy consumption, but also in the efficient use of resources (FOSSATI et al., 2016; SCHUTZE; HOLZ; ASSUNÇÃO, 2022).

Figure 2 – PBE Edifica label (Energy Efficiency Classification)


A
B
C
D
E
INMETRO
PBE Edifica
Eficiência Energética
Edificações Comerciais, de Serviços e Públicas
Edificação: XXXXXXXX xxxxxxxxxxxxxx
Endereço: Xxxxxxxxxxxx Xxxxxxxxxxxxxx Xxxxxxxxxxxxxx Xxxxxxxxxxxxxx Xxxxxxxxxxxxxx Xxxxxxxxxxxxxx
Cidade/UF: Xxxxx/XX
Portaria: Xxxxxx
Data da ENCE de projeto: XX/XX/XXXX
✓ ENCE PROJETO
ENCE EDIFICAÇÃO CONSTRUÍDA
A classe de eficiência energética alcançada deve ser confirmada pela ENCE DA EDIFICAÇÃO CONSTRUÍDA
Mais eficiente
Menos eficiente
A XXX
Estimativa do Consumo Anual de Gás: XX,XX m³/ano
Estimativa do Consumo Anual de Energia Elétrica: XXX.XXX,XX kWh/ano
Estimativa do Consumo de Energia Primária: XXX.XXX,XX kWh/ano
INFORMAÇÕES DA EDIFICAÇÃO
Edificação (sem a geração) B
Geração de energia renovável
Geração: XX% Edificação Xxxxxxxxx
Energia gerada: XX.XXX kWh/ano
Informativo
Uso racional de água
Economia: XX% redução em relação à condição de referência
Emissões de CO₂
Emissões CO₂: ⬇ XX% redução/acréscimo em relação à condição de referência
Emissões: XXX t.CO2/ano
Observações: 1 - A etiqueta de projeto tem validade de 5 anos ou até ser emitida a etiqueta da edificação construída. 2 - Para verificar a validade da etiqueta, consulte a página eletrônica do INMETRO: www.inmetro.com.br.
PROCEL
PROGRAMA BRASILEIRO DE ETIQUETAGEM
LOGOTIPO DO OIA
Nº. REGISTRO DO OIA/PC
1/2

Source: PBE Edifica (2020)

Reducing CO2 emissions is a global trend that requires more than small changes, it demands significant transformations in the built environment to promote a low-carbon path, with investments in smart technologies. Buildings are primarily responsible for the increase in CO2 emissions, mainly due to excessive energy consumption. Therefore, the construction sector needs to increase investment in energy efficiency to reduce its carbon footprint (AZOUZ; ELARIANE, 2023).

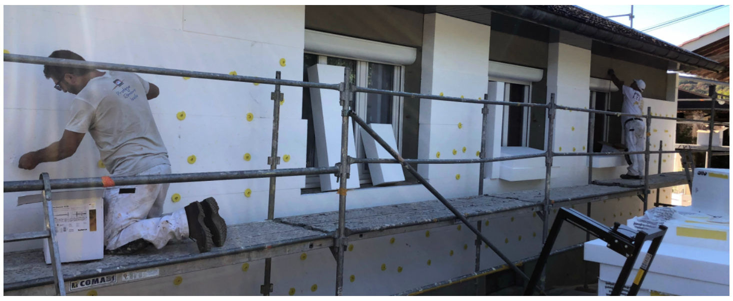
The energy efficiency is directly correlated with energy consumption. Therefore, the increase of energy efficiency, among many other aspects, can be evaluated through the energy savings they generate. Energy saving correspond to the amount of energy that is not necessary anymore when energy efficiency solution are applied, creating a relative economy regarding the previous or standard consumption. They underpin the multiple benefits of EE, and are associated with economic, social and environmental benefits (IEA, 2024b).

2.3 RETROFIT SUGGESTIONS

Retrofitting makes older structures safe and sustainable (EIB, 2024). As mentioned in the problem section, approximately 75% of the buildings expected to exist in 2050 have already been constructed (EEA, 2024). Thus, the practical challenge of renovating existing buildings is considered one of the most important problems in reducing energy consumption and CO2 emissions (ZUNE et al., 2020; SHARMA et al., 2022).

Therefore, appropriate decision-making is long-term, including the refurbishment and efficiency of buildings, which can significantly increase thermal performance and therefore reduce energy use (ABOELATA, 2021). As the retrofit is made on an already built structure, most of the time there are many more constraints compared to the implementation of the same solutions in a new building, which could become a high-complexity renovation, as shown in Figure 3 below.

Figure 3 – Retrofit example



Source: Services en Bâtiment (2024)

Therefore, taking into account the level of complexity of each structure, a comprehensive approach is essential (SHARMA et al., 2022). The counterpart of this complexity is the high variability in databases that keep records of retrofits, which is equilibrated with the relatively low number of options available. In the company, the most frequent retrofit improvements are described below.

1. Lighting - Relamping (LED) + office settings
2. Lighting - Installation of management equipment (dimming and/or presence detection)
3. Ventilation - Replacement or installation of dual-flow air unit with heat exchanger
4. Ventilation - Optimization of dual-flow heat exchanger (temperature and/or schedule)
5. Envelope - Strengthening insulation from the inside/outside
6. Envelope - Insulation of the ground floor
7. Envelope - Installation of high-performance exterior joinery as a complete replacement
8. Heating - Replacing the current system with a heat pump
9. Heating - Optimization of terminal emitters (temperature and/or schedule)
10. Heating - Replacement of terminal emitters
11. Management - Setting virtuous temperature guidelines
12. Management - Reducing office and reprography equipment operation during inactivity

2.4 ENERGY AUDITS

Energy audits should enable the project owner to make informed decisions regarding the interventions required for improving the energy performance of their building. These energy audits are mandatory for companies with more than 250 employees and/or an annual revenue exceeding 50 million euros (excluding taxes) and an annual balance sheet exceeding 43 million euros [1].

Additionally, at the beginning of 2023, it was announced that the tertiary energy audit will be mandatory for companies with an annual energy consumption ranging from 10 to 100 terajoules. If the company follows ISO 50001[2], it is exempt from this requirement (OPERA, 2023).

An energy audit is expected to have five stages. The first one is an inventory of fixtures: a visit to the building and a detailed note-taking; the second one is an energy's assessment: a critical condition analysis; the third is the enhancement initiatives: the prepositions for improvement, which must be showed in different scenarios. Finally, there is a financial analysis, used in deciding whether to implement an improvement, and if so, when to do it (ADEME, 2020).

Figure 4 – Equipments

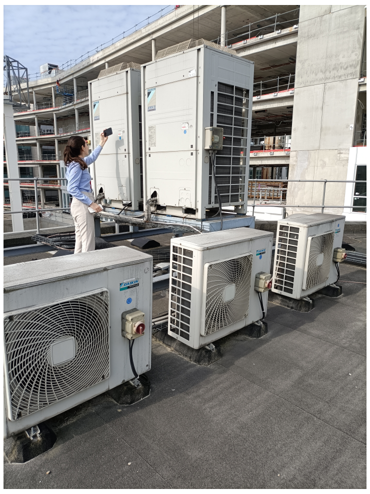


Figure 5 – IR camera

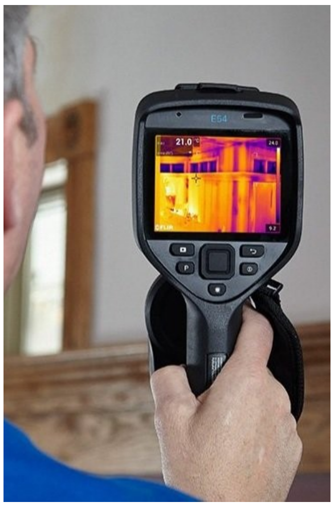


Figure 6 – Envelope

Source: Author

The visit is very detailed. In the envelope information collect, the auditor is supposed to investigate the wall materials, the type of window's glass, if there is any gas to insulate it, type and thickness of all insulation, colors to compute absorbance, etc. by the use of proper tools and equipment, but also their experience. Also, the installation of the smart meter 'Linky' is mandatory since 2021, allowing the collect of energy consumption values directly from its source.

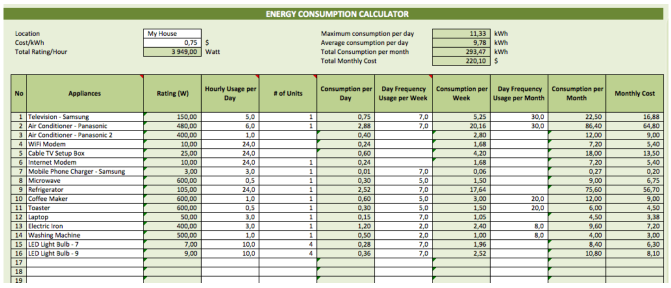
[1] Because the quantity of buildings is huge, to assure the buildings are following the regimentation, it is mandatory to insert the energy consumption of all buildings in the Platform OPERAT (MTE, 2023).

[2] ISO 50001 is an international standard for energy management systems that provides organizations with a framework to establish, implement, maintain, and improve their energy performance.

Beyond that, the auditor is supposed to collect all possible information about the equipments: model, year of installation, date of start and frequency of utilization, how many there are, how they are connected and how they feed up the building, if there are forms of technical building management (TBM); if there are people responsible for the verification in case of malfunction or even full-time. In the lighting, the quantity of lamps in each room, the type of lighting.

Also, through interviews with the maintenance responsible or employees, the auditor needs to collect the occupation schedule and operation of the building to have the most precise information to suggest retrofit suggestions. All of these information are going to be used in the subsequent phases, which discriminate two audit types: by simulation or by expert opinion. Both of them are illustrated in the Figures below:

Figure 7 – Calculation Table: Analyse Conso

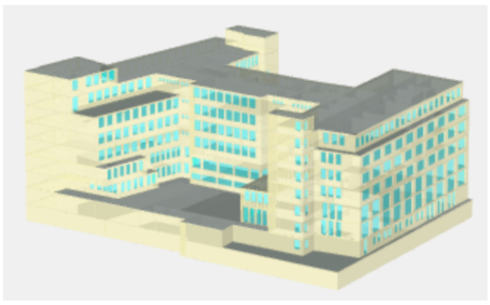


ENERGY CONSUMPTION CALCULATOR

Location	My House		Maximum consumption per day	11,33	kWh
Cost/kWh	0,75	$	Average consumption per day	9,78	kWh
Total Rating/Hour	3 949,00	Watt	Total Consumption per month	293,47	kWh
			Total Monthly Cost	220,10	$

No	Appliances	Rating (W)	Hourly Usage per Day	# of Units	Consumption per Day	Day Frequency Usage per Week	Consumption per Week	Day Frequency Usage per Month	Consumption per Month	Monthly Cost
1	Television - Samsung	150,00	5,0	1	0,75	7,0	5,25	30,0	22,50	16,88
2	Air Conditioner - Panasonic	480,00	6,0	1	2,88	7,0	20,16	30,0	86,40	64,80
3	Air Conditioner - Panasonic 2	400,00	1,0		0,40		2,80		12,00	9,00
4	WiFi Modem	10,00	24,0		0,24		1,68		7,20	5,40
5	Cable TV Setup Box	25,00	24,0		0,60		4,20		18,00	13,50
6	Internet Modem	10,00	24,0	1	0,24		1,68		7,20	5,40
7	Mobile Phone Charger - Samsung	3,00	3,0	1	0,01	7,0	0,06		0,27	0,20
8	Microwave	600,00	0,5	1	0,30	5,0	1,50		9,00	6,75
9	Refrigerator	105,00	24,0	1	2,52	7,0	17,64		75,60	56,70
10	Coffee Maker	600,00	1,0	1	0,60	5,0	3,00	20,0	12,00	9,00
11	Toaster	600,00	0,5	1	0,30	5,0	1,50	20,0	6,00	4,50
12	Laptop	50,00	3,0	1	0,15	7,0	1,05		4,50	3,38
13	Electric Iron	400,00	3,0	1	1,20	2,0	2,40	8,0	9,60	7,20
14	Washing Machine	500,00	1,0	1	0,50	2,0	1,00	8,0	4,00	3,00
15	LED Light Bulb - 7	7,00	10,0	4	0,28	7,0	1,96		8,40	6,30
16	LED Light Bulb - 9	9,00	10,0	4	0,36	7,0	2,52		10,80	8,10
17										
18										
19										

Figure 8 – Simulation Pleiades



Source: Author

The simulation type has the energy assessment and enhancement initiatives described in a very detailed manner, by the use of building thermal dynamic simulations. It helps to improve the reliability of both the breakdown of energy consumption by category and the estimation of energy savings, but it still presents significant uncertainties.

The expert audit type is the one in which savings estimates are established according to the experts experience and static calculations, supplemented by the use of the database of energy audit results conducted by the company on similar buildings.

However, both the categories have limitations. The audit by simulation is precise but very time-consuming and financially expensive. The expert type is fast, but less precise and extremely reliable on the experts experience, leading the company to be dependent of the employees. Beyond that, it can have high variability depending on the person responsible for the study. To improve the precision, efficiency and standardization of these expert predictions, the use of machine learning can be recommended and it is expected to generate fast, reliable and uniform results.

The data used in this study is acquired by the audits, which are made through a single or multiple visits in the building, the collection of documentation, the analysis of actual consumption of the buildings and finally the development of a thermal simulation.

2.5 MACHINE LEARNING

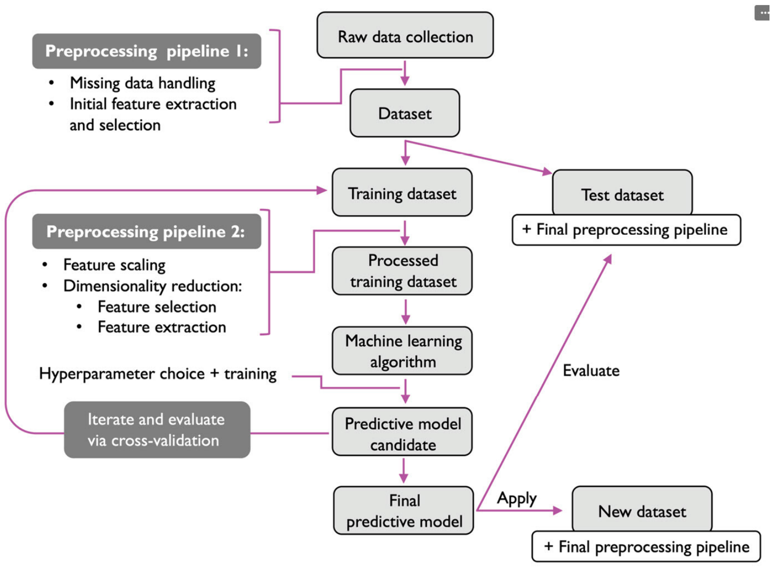
Generally seen as a sub-field of artificial intelligence, machine learning algorithms allow the systems to make decisions autonomously without any external support. Such decisions are made by finding valuable underlying patterns within complex data (SAH, 2020). An interesting definition is made by Zhou (2021):

> Machine learning is the technique that improves system performance by learning from experience via computational methods. In computer systems, experience exists in the form of data, and the main task of machine learning is to develop learning algorithms that build models from data. By feeding the learning algorithm with experience data, we obtain a model that can make predictions on new observations (ZHOU, 2021).

2.5.1 Machine Learning Cycle

The goal of Machine Learning algorithms is to automatically learn from data by using general procedures (RIBEIRO; SINGH; GUESTRIN, 2016). Scientific ML combines data-driven techniques with specialized domain knowledge, following a cyclical workflow where scientists hypothesize, design experiments, analyze results, and refine assumptions (SOUZA et al., 2022).

Figure 9 – Machine learning cyclic steps

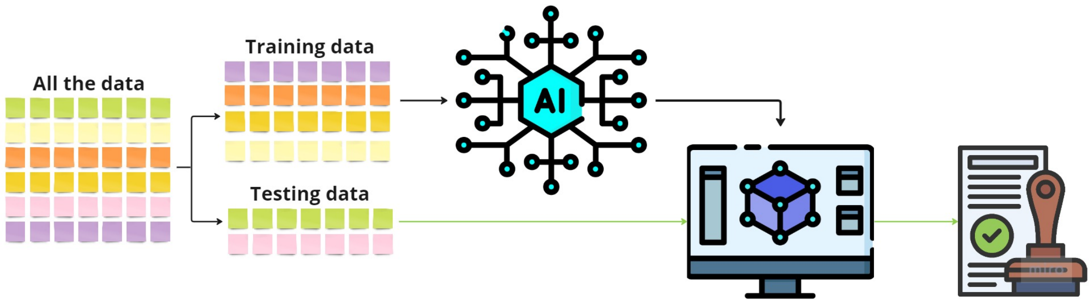

Preprocessing pipeline 1:
• Missing data handling
• Initial feature extraction and selection
Raw data collection
Dataset
Training dataset
Test dataset
+ Final preprocessing pipeline
Preprocessing pipeline 2:
• Feature scaling
• Dimensionality reduction:
• Feature selection
• Feature extraction
Processed training dataset
Machine learning algorithm
Hyperparameter choice + training
Evaluate
Iterate and evaluate via cross-validation
Predictive model candidate
Final predictive model
Apply
New dataset
+ Final preprocessing pipeline

Source: Raschka, Liu, and Mirjalili (2022)

It generally starts in the data acquisition phase, where relevant information are gathered in order to form the bases necessary to the predictions. Data quality is extremely influent to a well precise model behavior.

Following acquisition, data preparation as shown in Pipeline I is conducted, which involves cleaning the data, handling missing values, and possibly transforming variables to enhance data quality and consistency. Once the data is prepared, it is divided into training and test subsets to assess model performance and prevent overfitting, allowing for a reliable evaluation of how the model will generalize to unseen data. The training and validation phase are the core of the IA implementation that requires the ensemble of the parameters (the set of variables associated with the model). The overall scheme of the training and validation is illustrated in the Figure 10 below:

Figure 10 – Training and testing - How to assure the validation of the model



Source: Author

Additionally, this phase requires determining the most suitable hyperparameters, as these settings control the behavior of the model and can significantly impact its performance and accuracy. Once the model has been trained and optimized, the evaluation phase follows. By evaluating the model on test data, practitioners gather insights into its accuracy, precision, and ability to generalize to unseen data (RASHIDI et al., 2023). These insights feed back into the model tuning phase, where hyperparameters—configuration settings that influence the model's learning behavior — are adjusted to optimize performance.

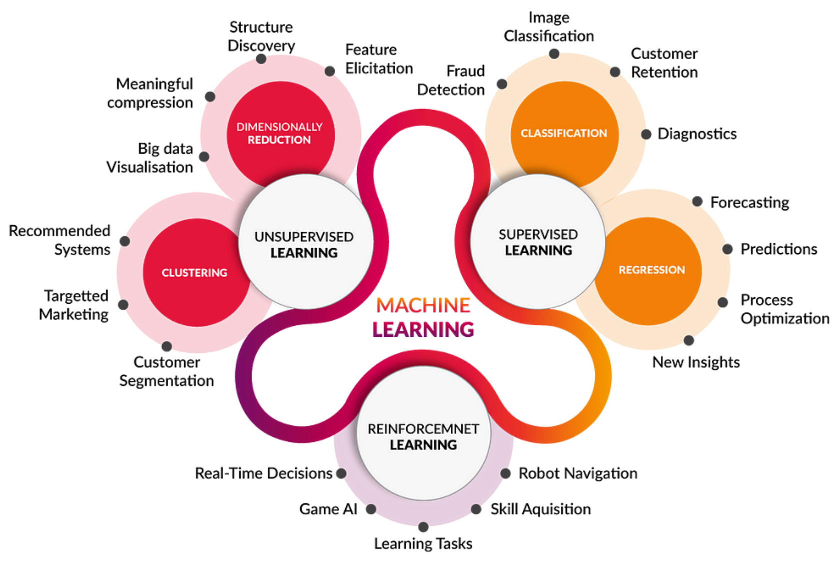
Inside Pipeline II occurs the cyclical operation of verifying the quality of the model versus its simplicity, as the goal is to have the most precise and more user-friendly model possible. To ensure that the model is interpretable and that predictions align with do-main knowledge, SHAP (SHapley Additive exPlanations) analysis is often performed after tuning. SHAP values assess the impact of each feature on individual predictions, supporting transparency and interpretability (LUNDBERG, 2017). This interpretative insight is then cycled back to refine feature selection and model tuning, making the model progressively more robust and transparent.

Finally, there's the final preprocessing pipeline, which occurs as a remodeling of the variables to their original values (removal of normalization, for example), so the users will be in contact with the real values in a user-friendly interface. Also, new data can be continuously included as the whole work frame is already defined.

2.5.2 Machine Learning models

The machine learning methods are many, and each type is well suited to an application, as shown in Figure 11.

Figure 11 – Machine learning types



Source: Buckley Barlow (2019)

As shown in the previous figure, machine learning has 3 mainly types of learning algorithms: supervised, unsupervised, reinforcement, but also a fourth one that is a mix of supervised and unsupervised, called semi-supervised. According to Sarker (2021), each one of them corresponds to the following:

- *Supervised learning:* utilizes labeled datasets to train algorithms, allowing for accurate classification of data and prediction of outcomes. By using labeled inputs and outputs, the model can assess its own accuracy and improve over time, which is essential for applications that require pattern recognition and decision-making based on predefined data (SUBASI, 2020).
- *Unsupervised learning:* analyzes unlabeled datasets, facilitating the discovery of hidden patterns without human intervention. This approach is typically applied in three main tasks: clustering, which organizes similar data points; association, which identifies relationships between variables; and dimensionality reduction, which simplifies complex datasets while retaining essential information (ALIRAMEZANI; KOCH; SHAHBAKHTI, 2022).
- *Reinforcement learning:* enables software agents and machines to automatically evaluate the optimal behavior in a particular context or environment to improve its efficiency.

There are many Machine Learning methods whose complexity is great, and there is no single one-size-fits-all type of algorithm that is best to solve a problem. The kind of algorithm depends on the problem, number of variables, etc. Beyond that, there are two classes of machine learning models. Classification models, that predict class membership, and Regression models that predict a number (WAKEFIELD, 2021).

As this projects want to predict the savings of retrofit improvements, it focus on prediction models, which the most popular are :

- Artificial neural networks: attempts to simulate a human brain such that a computer can learn and make decisions in a human-like manner;
- Decision Tree: recursively evaluating different features and using the feature that best splits the data at each node;
- Linear and Logistic regression: extends linear regression for classification problems and models the probabilities with two possible outcomes. Assumption: linearity, no outliers and independence.

From the neural networks, it emerged a whole new field of research that is called deep learning and whose functioning is not going to be covered in this project. From decision tree, two other well known methods, that are:

- Random forest: multiple decision trees to each subset of data;
- Gradient boosting: combination of weak predictors to form a strong one.

Choosing an appropriate machine learning model is critical, as the model type directly impacts the system's ability to capture patterns, make accurate predictions, and generalize effectively to new data.

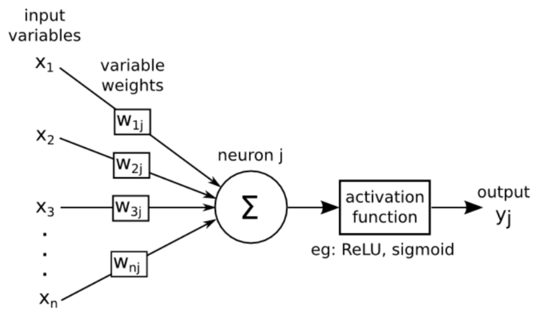
Each model has its strengths, limitations, and underlying assumptions about the data; for instance, linear regression models assume linear relationships, while decision trees can capture complex, non-linear patterns but may be prone to overfitting.

Selecting a model that aligns well with the data structure and the project objectives allows practitioners to maximize performance while avoiding biases or inaccuracies. A well-chosen model not only optimizes predictive accuracy but also enhances efficiency, interpretability, and scalability, making it an indispensable step in building reliable, robust machine learning applications.

2.5.3 Artificial Neural Networks (ANN)

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next network layer (IBM, 2017). The model created by Frank Rosenblatt in 1958 show simplest neural network that consists of n number of inputs, only one neuron, and one output, n being the number of features of the dataset (DASARADH, 2020).

Figure 12 – Preception of ANN



Source: Dasaradh (2020)

1. For each input, multiply the input value x_1 with weights w_1 and sum all the multiplied values. Weights — represent the strength of the connection between neurons and decide how much influence the given input will have on the neuron's output.
Therefore: $\sum x \cdot w$
2. Add bias b to the summation of multiplied values, called z: $z = x \cdot w + b$
3. Pass the value of z to a non-linear activation function, used to introduce non-linearity into the output.
It can be used Logarithmic, Sigmoid, ReLU functions $y(z)$.

To verify the accuracy of the solution, the methods are backpropagation and optimization. Backpropagation refers to the estimation of how far the answer is from the desired solution, with the help of a loss function. The *Loss function* is the MSE (mean squared error) calculated to each pair of calculation and prediction. The loss function is used to calculate the *Cost Function*, that is the mean of the MSE of all data.

To find the best weights and bias for the Perceptron, it is necessary to know how the cost function changes about weights and bias. This is done with the help of the gradient (rate of change) of the cost function concerning to the weights and bias.

The optimization is the selection of best weights and bias of the perceptron. The weights and bias are updated as follows, and the backpropagation and gradient descent is repeated until convergence. Learning rate (α) is a hyperparameter which is used to control how much the weights and bias are changed.

2.5.4 Gradient Boosting Machine (GBM)

GBM is part of the Boosting class, which is actually an ensemble of learning algorithms that improve robustness of a single estimator, by combining multiple weak or average predictors (RAY, 2023).

According to Temizel et al. (2020), the GBM method can be seen as a numerical optimization algorithm that aims at finding an additive model that minimizes the loss function. It is capable of reducing the model variance by averaging several decision trees, and it is also capable of reducing the bias through the sequential error modeling by adding, at each step; a new decision tree ("weak learner") that best reduces the loss function. This is shown in Figure below.

Figure 13 – GBM learning mechanism

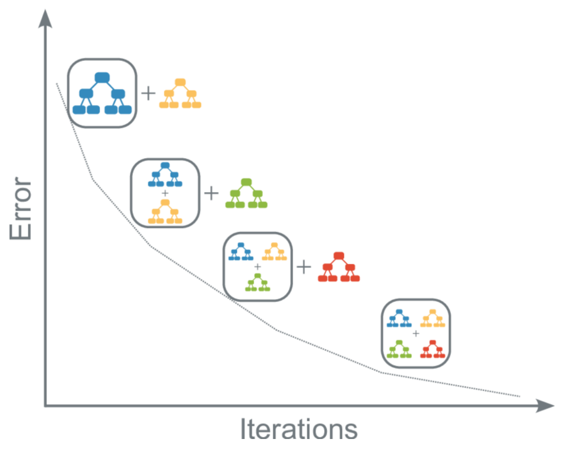
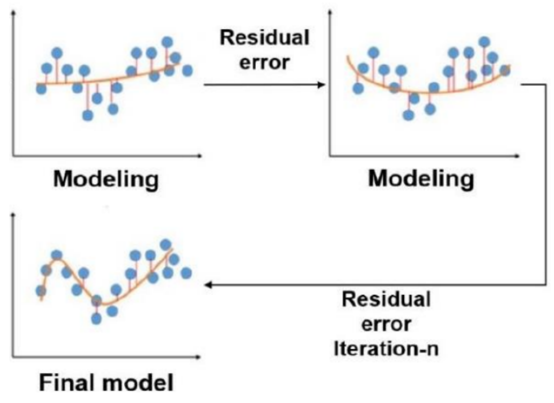


Figure 14 – GBM interations



Source: Temizel et al. (2020)

According to Temizel et al. (2020), a simplified calculation mechanism to GBM:

1. Initialization: set the residual $r_0 = y$ and $\hat{f} = 0$. y is an initial guess of $\hat{f}$.
2. For $k = 1, 2, ..., K$, do the following:
a Randomly choose a subsample $(y_i, x_i)^{N'}$ from the full training dataset, with N' is the number of data points corresponding to the fraction;
b Using $(y_i, x_i)^{N'}$ fit a decision tree $\hat{f}^k$ of depth d to the residual r_{k-1}.
c Update $\hat{f}$ by adding the decision tree to the model $\hat{f}(x) \leftarrow \hat{f}(x) + \alpha\hat{f}^k(x)$.
d Update the residual $r_k \leftarrow r_{k-1} - \hat{f}(x)$.
3. End For

Where:

d: the depth of decision trees or the maximum interaction order of the model;
K: the number of iterations, which also corresponds to the numbers of decision trees;
α: the learning rate, which is usually a small positive value between 0 and 1, where decreases lead to slower fitting, thus requiring the user to increase K;
η: the fraction of data that is used at each iterative step;

As shown, the algorithm starts by initializing the model by a first guess, which is usually a decision tree that maximally reduces the loss function (the mean squared error), then at each step a new decision tree is fitted to the current residual and added to the previous model to update the residual. The algorithm continues to iterate until a maximum number of iterations, provided by the user, is reached. This process is so-called stage wise, meaning that at each new step the decision trees added to the model at prior steps are not modified, improving in the regions where it does not perform well (TEMIZEL et al., 2020).

Gradient Boosting Machines (GBM) have become instrumental in the construction and building industries due to their predictive power and adaptability to complex, multidimensional data. For instance, GBMs have been used in predicting energy consumption in high-performance buildings, where they can handle diverse datasets including weather conditions, occupancy patterns, and equipment operation metrics.

In studies such as Touzani, Granderson, and Fernandes (2018) have made their study upon a large dataset of 410 commercial buildings. The model training periods were varied and several prediction accuracy metrics were used to evaluate the model's performance. The results show that using the gradient boosting machine model improved the R-squared prediction accuracy and the CV(RMSE) in more than 80 percent of the cases, when compared to an industry best practice model that is based on piecewise linear regression, and to a random forest algorithm.

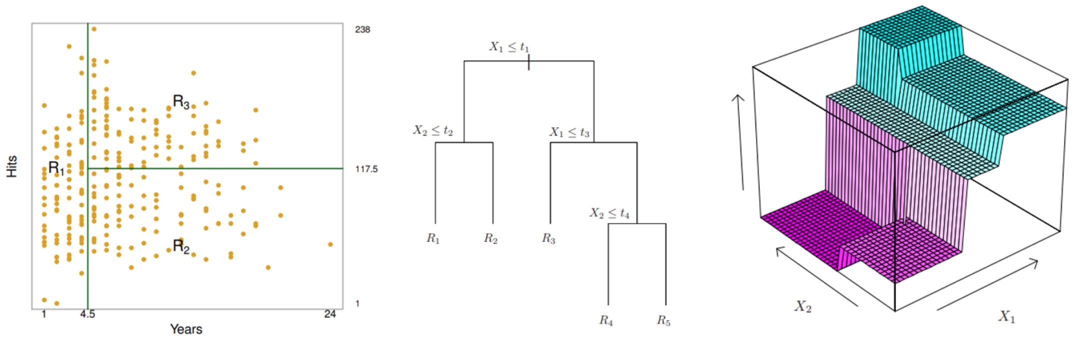
Another example is the work of Shchetinin (2019) that has studied 300 buildings in Russia and shown that the capacity and effectiveness of GBM in solving the problem of energy efficiency was tested on both model and real data of energy consumption from smart meters of building conglomerate. As a whole, the GBM model showed higher forecasting accuracy than the regression and random forest models for all tested training periods. The results of computer experiments showed that the use of the GBM model can improve the accuracy of energy efficiency assessment as a separate building and a complex of buildings as a whole.

The ability to extract practical insights from complex datasets has become crucial across various industries in this era of data abundance. Predictive modeling, a central aspect of this process, leverages machine learning to forecast future outcomes, trends, and patterns with unprecedented accuracy (GUPTA; SHARMA; ALAM, 2024).

2.5.5 Decision Tree

Decision Trees are a widely used class of supervised learning algorithms that facilitate both classification and regression tasks, the case of this project as described in the Figure 16 below. As shown, the space variable x prediction is subdivided into trees, which are actually multidimensional, considered the many variables used to make the model's prediction.

Figure 15 – Decision tree algorithm



Source: Hastie, Friedman, and Tibshirani (2001)

The tree regression algorithm is described through the steps provided by Hastie, Friedman, and Tibshirani (2001):

1. Divide the predictor space—that is, the set of possible values for $X_1, X_2, \ldots, X_p$—into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
2. For every observation that falls into the region R_j, make the same prediction, which is simply the mean of the response values for the training observations in R_j.
3. Divide the predictor space into high-dimensional rectangles $R_1, \ldots, R_J$ that minimize the RSS, given by:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

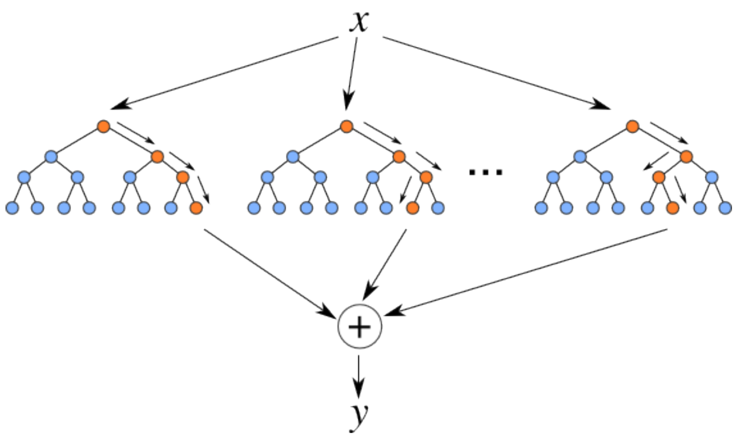
where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box. Which is to select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $X|X_j < s$ and $X|X_j \geq s$ leads to greatest possible reduction in RSS.
5. Repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions, looking for the smaller RSS within the simpler tree without overfitting.

This procedure generates a sequence of trees indexed by a nonnegative tuning parameter α. For each value of α, there is a a subtree $T \in T_0$ such that RSS + $\alpha|T|$ is as small as possible, with $|T|$ indicating the number of terminal nodes of the tree T. Therefore, according to Liu (2015), the tuning parameter α controls a trade-off between the subtree complexity and its fit to the training data.

2.5.6 Random Forest

Random Forest is an ensemble learning method primarily used for classification and regression tasks. It builds multiple decision trees during training and outputs the mode of their predictions for classification or the mean prediction for regression. This approach enhances model accuracy and controls overfitting.

Figure 16 – Random forest algorithm



Source: Hastie, Friedman, and Tibshirani (2001)

The random forest algorithm to regression is described through the steps provided by Hastie, Friedman, and Tibshirani (2001):

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x in regression:

$\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

As shown in the algorithm, what Random Forest does is to create aleatory decision trees which take into account different trunks of data, take the best one of each of them (through the process described in the last section) and combine the final prediction by calculating the mean of the prediction given by each decision tree of its ensemble. Therefore, random Forest provides a measure of feature importance, indicating the contribution of each feature to the model's predictions.

2.5.7 Bias and Variance

Controlling the flexibility of a machine learning algorithm is strongly linked to the balance between bias and variance. According to Wickramasinghe (2024), bias refers to error caused by oversimplification, while variance is due to oversensitiveness.

Figure 17 – AI applications

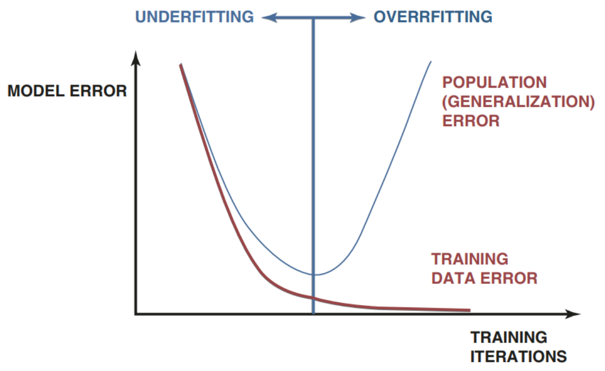
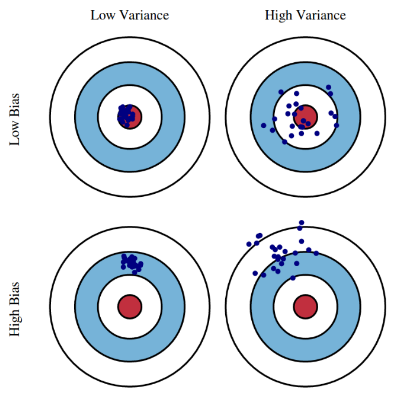


Figure 18 – ML Mathods



Source: Aliferis and Simon (2024)

Balancing bias and variance is a fundamental challenge in machine learning, as it directly influences a model's ability to generalize beyond the training data. Bias occurs when a model is overly simplistic, failing to capture the true underlying patterns in the data, while variance arises from excessive sensitivity to fluctuations in the training set, leading to overfitting. According to Aliferis and Simon (2024):

> Successful data analysis methods balance training data fit with complexity since too complex model [...] leads to overfitting [...] whereas too simplistic models [...] lead to under-fitting, which makes future predictive performance small (ALIFERIS; SIMON, 2024).

Rather than simply avoiding overfitting or underfitting, the goal is to achieve an optimal trade-off between these two sources of error. Strategies like cross-validation, regularization, and adjusting model complexity allow data scientists to fine-tune this balance. For example, using regularization techniques such as L1 or L2 penalties can prevent models from becoming too complex, while methods like cross-validation provide a more reliable estimate of how the model will perform on unseen data.

Ultimately, this balance is not static but needs continuous adjustment depending on the problem, data size, and algorithm choice, underscoring the importance of iterative experimentation in model development.

2.5.8 Filling the gaps

In data analysis and machine learning, dealing with missing data is a frequent and critical challenge. Information gaps within datasets can limit analytical accuracy and weaken the reliability of any insights drawn. Addressing these gaps requires careful planning and often involves a range of strategies to maintain data quality. From simple removal techniques to complex modeling solutions, each approach aims to mitigate the potential biases and inaccuracies that can arise when key information is absent.

The missing values are information gaps. At best, these information gaps may prevent us from reaching important insights. It can cast doubt on the robustness of insights derived from the data or limit our view of the whole, and in the worst scenario, they can become barriers that lead to biased hypothesis and, therefore, poor machine learning models: "garbage in, garbage out".

In order to minimize the gap effects, there are many strategies in data science. According to SHAW (2021), no matter how fancy the algorithm, data quality will always be a limiting factor. To help reduce the impact of missing data, there are three levels of strategy. The simple one ($\star$) is to drop the rows that present missing values. The intermediate one ($\star\star$) is to replace the missing values with a statistic calculated from the values which are not missing. And the advanced one ($\star\star\star$) is to iteratively model missing values using non-missing data.

- ($\star$) **Drop method:** Consists of dropping all rows with missing values, but as we are already working with a small amount of data, removing all the lines where there's a missing value is not the best option for this project;

- ($\star\star$) **Fill method:** Consists of imputing missing values using statistics based on non-missing values, this approach is well adapted to our case, as it needs almost no computational effort and little time.

- ($\star\star$) **Fill and indicate method:** Imputing as in (2), but adding columns indicating if a particular value was imputed, this is also a well adapted method to our situation. It needs only a little more storage and a small code adaptation when the metamodel will be used by the company's employees. To both methods, we use the median to the quantitative data and the mode to the qualitative data.

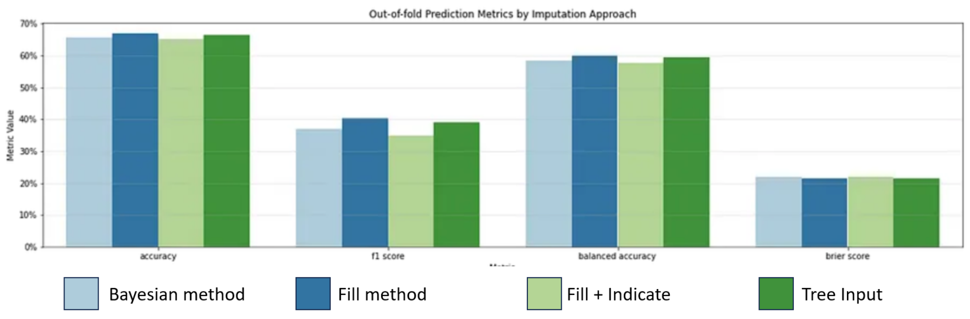
- ($\star\star\star$) **Bayesian method (MICE):** the Bayesian Ridge model, which is a Multivariate Imputation By Chained Equations algorithm (MICE) approach, that estimates a probabilistic model of the regression problem. Here the prior for the coefficient w is given by spherical Gaussian.

- (⋆ ⋆ ⋆) **Tree method:** the approach using an Extremely Random Forest model, also a MICE approach; that implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various subsamples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting (**SK-LEARN**).

- (⋆ ⋆ ⋆) **Completing the missing values with real data:** the data collect in the server was one of the first desires of this project, however due to the variability and complexity of the data and little time available, we prefer not to use this technique, with few exceptions.

SHAW (2021) made a test with the Fill methods and MICE approaches. The study used an example dataset to complete a dataset of 10 features, in which 3 of them had 15%, 24% and 5% of data missing, obtaining the results shown in the Figure 19 below :

Figure 19 – Cross validation results



Source: SHAW (2021)

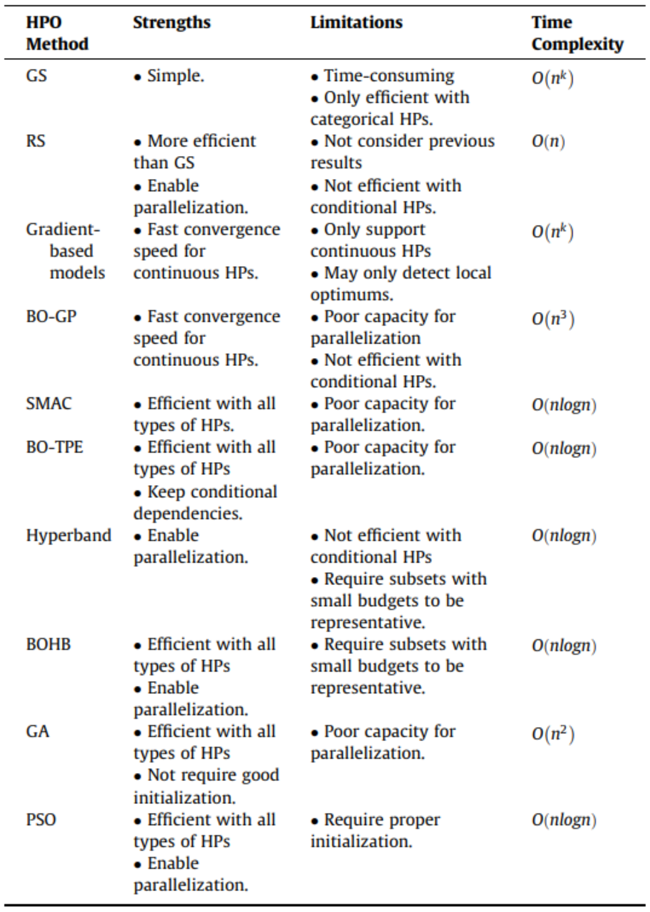
The results are real close and even if the MICE methods execution are a little bit more complex, separating between training and validation tests, they do not represent a significant energy saving in terms of any out of fold cross validation tests. Ideally, a test in our dataset should be conducted, but due to time constraints, we are nor going to test every method.

In conclusion, handling missing data is essential for maintaining data integrity and producing reliable machine learning models. Each method for addressing information gaps—ranging from simple deletion and statistical imputation to advanced Bayesian and tree-based imputation techniques—carries its strengths and trade-offs. While more sophisticated approaches like MICE can yield marginal improvements, their added complexity and resource requirements may not always justify their use, particularly in time-constrained projects. As shown by SHAW (2021) , simpler imputation methods can perform comparably well for cross-validation, making them practical for many scenarios where data is limited or computational efficiency is critical.

2.5.9 Hyperparameters optimization

Hyperparameter calibration plays a crucial role in definition of the most assertive model. Lavesson and Davidsson show that tuning hyperparameters is often more important than the choice of the machine learning algorithm (WEERTS; MUELLER; VANSCHOREN, 2020). Optimizing these hyperparameters helps to find a suitable balance, for its tuning, there are many methods to find the optimal values, as shown below.

Figure 20 – Hyperparameters optimization techniques



HPO Method	Strengths	Limitations	Time Complexity
GS	• Simple.	• Time-consuming • Only efficient with categorical HPs.	$O(n^k)$
RS	• More efficient than GS • Enable parallelization.	• Not consider previous results • Not efficient with conditional HPs.	$O(n)$
Gradient-based models	• Fast convergence speed for continuous HPs.	• Only support continuous HPs • May only detect local optimums.	$O(n^k)$
BO-GP	• Fast convergence speed for continuous HPs.	• Poor capacity for parallelization • Not efficient with conditional HPs.	$O(n^3)$
SMAC	• Efficient with all types of HPs.	• Poor capacity for parallelization.	$O(nlogn)$
BO-TPE	• Efficient with all types of HPs • Keep conditional dependencies.	• Poor capacity for parallelization.	$O(nlogn)$
Hyperband	• Enable parallelization.	• Not efficient with conditional HPs • Require subsets with small budgets to be representative.	$O(nlogn)$
BOHB	• Efficient with all types of HPs • Enable parallelization.	• Require subsets with small budgets to be representative.	$O(nlogn)$
GA	• Efficient with all types of HPs • Not require good initialization.	• Poor capacity for parallelization.	$O(n^2)$
PSO	• Efficient with all types of HPs • Enable parallelization.	• Require proper initialization.	$O(nlogn)$

Source: Grid Search (GS), Random Search (RS), Gradient-Based Models, Bayesian Optimization using Random Forest (SMAC), Bayesian Optimization-Gaussian Process (BO-GP), Bayesian Optimization HyperBand (BOHB), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO) (WEERTS; MUELLER; VANSCHOREN, 2020)

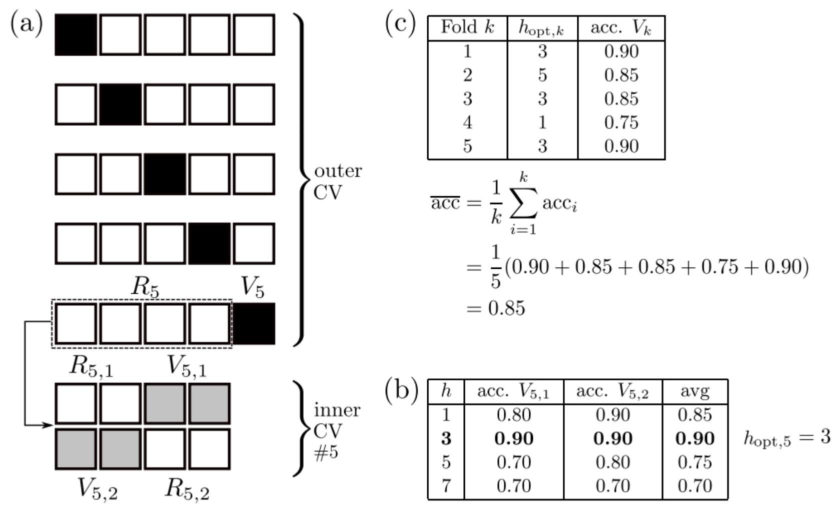
In conclusion, selecting the right hyperparameter optimization technique depends on the specific requirements and constraints of the model, dataset, and computational resources. While methods like grid search (GS) and random search (RS) are straightforward and widely applicable, they may not always be efficient for complex or high-dimensional problems. Advanced methods offer more targeted exploration of the hyperparameter space and can yield faster results, though they often require more setup or computational power. Ultimately, the strengths and limitations of each method highlight that there is no universally best approach—each comes with trade-offs between accuracy, efficiency, and resource allocation.

2.5.10 Cross Validation

Cross-validation is one of the most widely used data resampling methods for model selection and evaluation, being used to assess the generalization ability of predictive models and to prevent overfitting (BERRAR, 2019).

> In k-fold cross-validation, the available learning set is partitioned into k disjoint subsets of approximately equal size. Here, "fold" refers to the number of resulting subsets. This partitioning is performed by randomly sampling cases from the learning set without replacement. The model is trained on k-1 subsets, which, together, represent the training set. Then, the model is applied to the remaining subset, which is denoted as the validation set, and the performance is measured. This procedure is repeated until each of the k subsets has served as validation set. The average of the k performance measurements on the k validation sets is the cross-validated performance (BERRAR, 2019).

Figure 21 – Cross-validation: K-fold method



Fold k	$h_{\text{opt},k}$	acc. V_k
1	3	0.90
2	5	0.85
3	3	0.85
4	1	0.75
5	3	0.90

$$\begin{aligned}\overline{\text{acc}} &= \frac{1}{k}\sum_{i=1}^{k}\text{acc}_i \\ &= \frac{1}{5}(0.90+0.85+0.85+0.75+0.90) \\ &= 0.85\end{aligned}$$

h	acc. $V_{5,1}$	acc. $V_{5,2}$	avg
1	0.80	0.90	0.85
3	**0.90**	**0.90**	**0.90**
5	0.70	0.80	0.75
7	0.70	0.70	0.70

$h_{\text{opt},5} = 3$

Source: Berrar (2019)

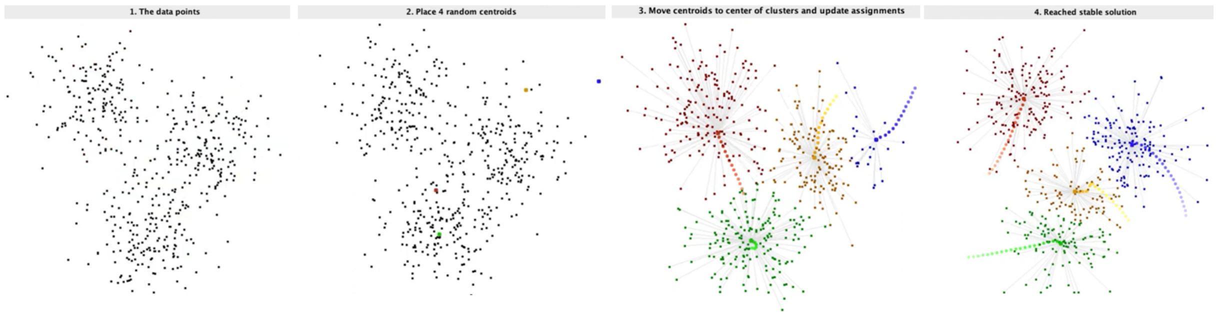
The choice of parameter k in validation crusade does not follow a precise rule, although divisions into 5 or 10 are common parties. As we increase the value of k, the difference in size between the original training set and the resampled subsets decreases, reducing thus the bias of the cross-validation technique. However, an increase in k also implies an increase in the time required to obtain the final validation result crusade (KUHN; JOHNSON, 2013).

Therefore, cross-validation is a crucial technique in machine learning and statistical modeling because it helps to evaluate the performance of a model by assessing its ability to generalize to unseen data. Instead of relying on a single train-test split, cross-validation divides the data into multiple subsets, trains the model on some subsets, and validates it on others.

2.5.11 Clusterisation

Clustering algorithms exploit the underlying structure of the data distribution and define rules for grouping the data with similar characteristics, resulting in the partition of a given dataset according to the clustering criteria without any prior knowledge about the dataset (AHMED; SERAJ; ISLAM, 2020). There are many clustering methods, such as partitioning methods, hierarchical, density and model based and fuzzy clustering. For its simplicity and low computation cost, this project presents specifically the K-means method

Figure 22 – K-means method



Source: Hagelbäck (2019)

As shown in the Figure, what k-means does is to define random centroids and iterate them in order to find the closest point to each centroid while keep them distant of the other clusters. To discover the ideal number of clusters in k-means, there are two main graphical methods to be applied: Elbow and Silhouette.

2.5.11.1 Elbow Method

The Elbow Method involves running the algorithm for various values of k (the number of clusters) and calculating the *Within-Cluster Sum of Squares (WCSS)* for each k (number of clusters). The WCSS measures the total squared distance between each point and the centroid of its cluster, where C_i represents the i-th cluster, x is a point, and μ_i is the centroid of cluster C_i.

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

As the number of clusters increases, WCSS decreases since each cluster becomes more focused. The Elbow Method plots k versus WCSS, revealing a point where the decrease rate in WCSS sharply reduces, forming an "elbow." This elbow represents the optimal cluster number, as further increasing k provides minimal reduction in WCSS, suggesting diminishing returns on intra-cluster homogeneity.

2.5.11.2 Silhouette Score

The **Silhouette Score** measures the degree to which a data point fits within its assigned cluster compared to others. For each data point, the Silhouette Score is defined as:

$$\text{Silhouette Score} = \frac{b - a}{\max(a, b)}$$

where:

- a is the average distance between the point and other points in the same cluster
- b is the average distance between the point and points in the nearest neighboring

The Silhouette Score ranges from -1 to +1, where higher scores indicate better-defined clusters. A score close to +1 suggests that the data point is well-clustered, while scores around 0 indicate points on cluster boundaries. Negative values suggest possible misclassification. Calculating the average Silhouette Score across all data points for varying k values helps identify the optimal number of clusters, with the peak score representing the ideal number of clusters.

Choosing the correct number of clusters is critical for effective clustering, as too few clusters may miss key patterns, while too many may overcomplicate the model. Several techniques are commonly used to determine the ideal number of clusters, including the Elbow Method and the Silhouette Score.

2.6 EVALUATION METRICS FOR MACHINE LEARNING

2.6.1 R² (Coefficient of Determination)

The coefficient of determination, often denoted as R^2, is a statistical measure used to assess the goodness of fit of a regression model. It quantifies the proportion of the variance in the dependent variable that is explained by the independent variables in the model. R^2 ranges from 0 to 1, with higher values indicating a better fit.

The formula for R^2 is given by:

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

Where:

y_i : Actual values

$\hat{y}_i$: Predicted values

$\bar{y}$: Mean of actual values

n : Number of data points

R^2 serves as a valuable performance indicator for regression models. A higher R^2 value indicates that a larger portion of the variance in the dependent variable can be explained by the model. This essentially means that the model is more effective at capturing and accounting for the fluctuations or changes in the data, making it a strong tool for understanding and predicting real-world phenomena.

For Craven and Islam (2011) *apud* Balczareki (2024), this coefficient expresses the amount of variation in the target variable which can be explained by the variation in the attributes of the set of data. However, the isolated analysis of this coefficient is not sufficient to determine to validate a model, as it is possible for a bad model to obtain a high R2 value. Its usefulness lies in comparing two valid models.

While R^2 is a useful metric for evaluating the explanatory power of a regression model, relying on it alone can be misleading. A high R^2 does not necessarily indicate a robust model; it may still be susceptible to overfitting, particularly if the model is complex and fits noise in the data rather than the true underlying trend. Therefore, R^2 should be considered alongside other metrics such as adjusted R^2, mean absolute error (MAE), or root mean square error (RMSE) to assess the model's predictive accuracy and generalizability. By comparing these metrics across models, a more comprehensive understanding of model performance can be achieved, ultimately aiding in the selection of the model that best balances explanatory power and predictive reliability.

2.6.2 RMSE (Root Mean Square Error)

The Root Mean Square Error, abbreviated as RMSE, is a widely used metric to measure the average error between predicted and actual values in a regression model. It provides a measure of the typical or root average magnitude of errors. RMSE is sensitive to outliers. The RMSE formula is:

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

Where:

$n =$ number of data points

$y_i =$ actual values

$\hat{y}_i =$ predicted values

In essence, a lower RMSE signifies that the model's predictions, on average, exhibit less divergence from the actual observed values, reflecting its enhanced precision in estimating the relationships between variables and generating forecasts. In simpler terms, a reduced RMSE implies that the model's typical prediction errors are smaller, underscoring its ability to provide more accurate estimations.

2.6.3 MAE (Mean Absolute Error)

The Mean Absolute Error, denoted as MAE, is another measure of the average error between predicted and actual values in a regression model. It represents the absolute value of the average magnitude of errors and is less sensitive to outliers compared to RMSE. The formula for MAE is given by:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where:

$$n = \text{number of data points}$$
$$y_i = \text{actual values}$$
$$\hat{y}_i = \text{predicted values}$$

Mean Absolute Error (MAE) serves as a method for quantifying the average magnitude of errors in a predictive model. It does so by taking the absolute value of the differences between the model's predictions and the actual observed values and then averaging these absolute differences. Unlike other metrics, such as RMSE, that square errors, MAE provides a straightforward measure of the typical size of errors in their original units, which can be more easily interpreted.

2.6.4 AE95 (Absolute Error at 95th Percentile)

AE95, or Absolute Error at the 95th Percentile, is a statistic that quantifies the magnitude of errors at a specific percentile of the error distribution. It's useful for understanding the upper tail of the error distribution, which can be crucial in some applications. The formula for AE95 is:

$$AE95 = \text{Percentile}\left(|y_i - \hat{y}_i|, 95\%\right)$$

In this formula, there is the 95th percentile of the absolute errors in the data. AE95 provides insights into the worst-case errors, which can be important in risk assessment and decision-making. In the context of AE95, or Absolute Error at the 95th Percentile, the 95th percentile represents a statistical point in the error distribution.

To obtain the 95th percentile of the absolute errors in the dataset is essentially to identify the error value below which only 5% of the data points fall. This is particularly significant in risk assessment and decision-making because it focuses on the upper tail of the error distribution, capturing extreme or worst-case scenarios. By examining the AE95, there is the estimation of a maximum potential error that the model might encounter.

2.7 SENSITIVITY ANALYSIS

The SHAP (Shapley Additive Explanations) framework offers a robust, game-theoretic approach for interpreting the outputs of machine learning models, rooted in the Shapley values developed in cooperative game theory. Originally proposed by Shapley (1953), these values are used to allocate payoffs fairly to players in a game based on their marginal contributions to the total payoff. SHAP adapts this concept to machine learning by treating each feature as a “player” and the model’s prediction as the “payoff,” allowing for an objective distribution of each feature’s contribution to the model’s output (ROZEMBERCZKI et al., 2022).

In practical applications, SHAP values enhance interpretability by quantifying the contribution of each feature to individual predictions. In mathematical terms, the Shapley value for a feature i is derived by evaluating its contribution to every possible subset of features, thus accounting for feature interactions and ensuring that each feature’s importance is assessed fairly. The Shapley value ϕ_i is given by the formula (ZENG, 2024):

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f(S \cup \{i\}) - f(S) \right]$$

Where:

F represents the set of all features,

$S \subseteq F \setminus \{i\}$ is every subset that excludes i,

$|S|$ is the size of subset S,

$f(S \cup \{i\})$ is the model predictions using subset $S \cup \{i\}$,

$f(S)$ is the model predictions using subset S

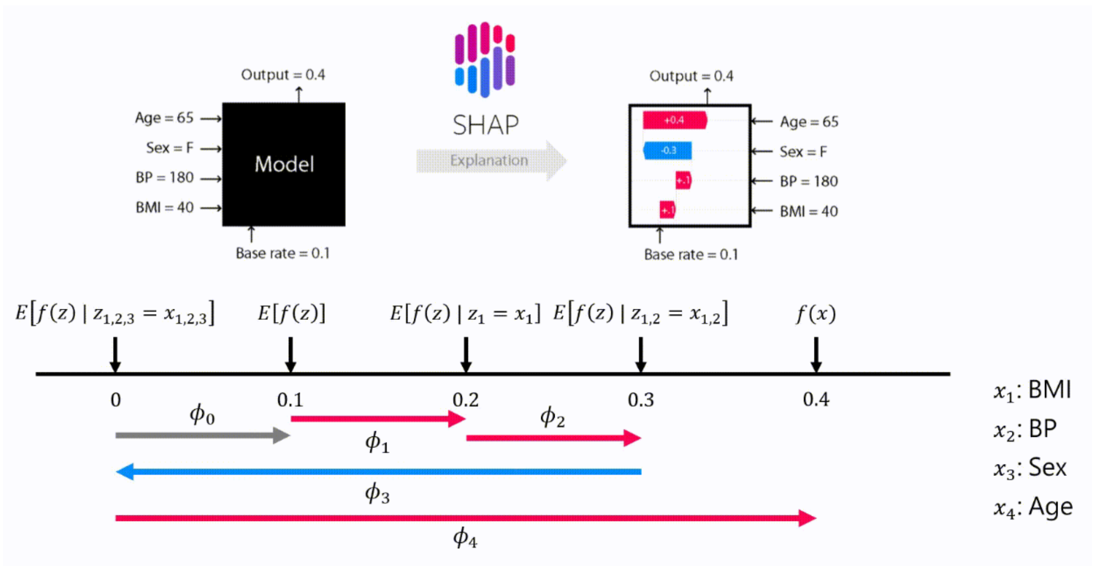
This equation captures the marginal impact of each feature by assessing how the model’s prediction changes with and without the feature across all feature subsets (ARROW et al., 1953). SHAP values, thus, provide both local explanations for individual predictions and global feature importance insights when aggregated across samples, making SHAP a powerful tool for interpreting complex models (LUNDBERG, 2017; SUNDARARAJAN; NAJMI, 2020).

Feature selection is another critical application of SHAP, especially in high dimensional datasets where irrelevant or redundant features can lead to decreased model performance. According to Guyon and Elisseeff (2003), feature selection not only improves computational efficiency but also enhances model interpretability by removing noise.

Feature selection is an essential step in the machine learning process that focuses on identifying the most important features in the data while discarding those that are unnecessary or repetitive. This step is critical for boosting model performance, minimizing overfitting, and simplifying the complexity of machine learning models, making them easier to interpret. In datasets with a large number of features, irrelevant or duplicate features can add noise and increase the likelihood of overfitting. Saeys (2007) apud Kraev et al. (2024) shows that this is particularly problematic in domains such as healthcare for example, where models are expected to make critical decisions. For instance, in the prediction of patient outcomes, irrelevant features might skew the model's predictions, leading to incorrect diagnoses or treatment plans.

The SHAP framework also offers visualization techniques to facilitate model interpretation. The summary plot, for instance, ranks features by their mean absolute SHAP value, while dependence plots display the relationship between specific features and their SHAP values across instances, highlighting how feature values affect model predictions (LUNDBERG, 2017; SUNDARARAJAN; NAJMI, 2020).

Figure 23 – SHAP Method



Source: Awan (2023)

Lastly, according to Awan (2023), SHAP values are useful for model interpretation due to their key properties: additivity, allowing the contribution of each feature to be summed up independently; local accuracy, enabling precise, localized interpretation of individual predictions; missingness, making them robust to irrelevant or missing data; and consistency, ensuring stability in interpretation as long as feature contributions remain unchanged. Overall, these properties make SHAP values an effective, consistent method for understanding feature importance in model predictions. These properties, combined with SHAP's rigorous game-theoretic foundation, makes it one of the most reliable tools for model interpretability in machine learning.

2.8 SIMILAR STUDIES

Recent studies highlight the effective application of machine learning techniques in predicting energy savings from retrofit actions in buildings. Xu (2020) wrote a thesis upon the application of machine learning to the retrofit effects on energy consumption. He applied a ML to assess the immediate and long term retrofit effect in energy reduction, energy end-use reduction, and LEED achievement, as a function of pre-retrofit energy use, pre-retrofit energy end-uses, short-term weather, long term climate, building characteristics, policy region, and past retrofit actions, for a commercial building portfolio.

Xu (2020) discovered that pre-retrofit energy use and short-term weather are key predictors of energy savings, while building characteristics strongly correlate with LEED certification. Combined capital and operational actions are more effective than capital actions alone, especially for reducing base load electricity and meeting LEED standards, with extreme weather yielding higher electricity savings.

Alanne and Sierla (2022) discussed the learning ability as a feature of buildings. They have concluded that the increasing autonomy of smart buildings, the evolving AI, and the increasing demand for interaction between humans and buildings challenge the future research. Further research is needed, for example, to find out to which extent the AI may enhance building performance and the buildings' adaptability to unpredicted changes when the entire system rather than single processes is concerned. The reviewed reinforcement learning applications involve adjustment both in real-time, hourly, and daily timescales. It is possible to conclude that the adjustments would perform better if they incorporated the outputs of asset management and prediction as state information to the reinforcement learning agent, being the major direction for future research to the application of AI to smart buildings.

Also, Zhou et al. (2023) wrote a review upon the impact of machine learning methods on the optimization and control of HVAC systems, as well as on building design and fault diagnosis and detection, coming to the conclusion that there are few practical examples of their adoption. The selection of appropriate machine-learning methods depends on several factors, such as the application, the data type, data quantity and quality, calculation cost and calculation complexity. At present, the machine learning methods used in HVAC system optimization are mainly supervised learning and reinforcement learning.

Finally, Bocaneala et al. (2024) reviewed AI applications and techniques that have been used in the context of retrofit projects. Their analysis revealed the potential advantages and difficulties associated with employing AI techniques in retrofit projects, and also identified the commonly utilized techniques, data sources, and processes involved, synthesizing the state-of-the-art of AI applications for Retrofit building actions.

According to the research of Bocaneala et al. (2024), machine learning accounts for 35% of the AI applications for retrofit in buildings, the most part of it dominated by supervised ML, as shown in the Figure below:

Figure 24 – AI applications

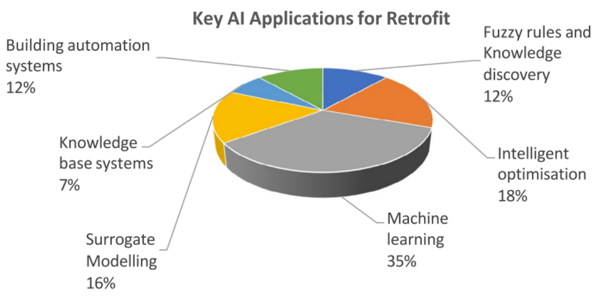
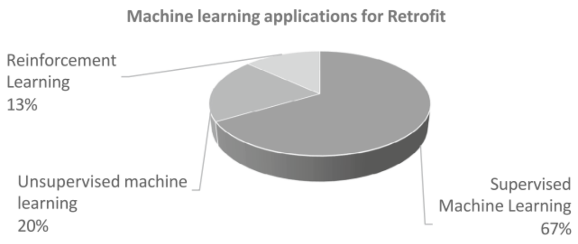


Figure 25 – ML Methods



Source: Bocaneala et al. (2024)

They have reviewed 56 articles, identifying that supervised ML methods such as deep learning methods were vastly implemented to discover the significant features that produce the cost-optimal retrofit strategy in an optimized way without having to undergo an exhaustive search process. The study's findings demonstrate that by integrating box plots and scatterplots, unsupervised machine learning techniques like scenario sampling (2-ary coverage), clustering (k-means), and dimensionality reduction (PCA) can be used to visually communicate high level information about KPI trade-offs across renovation scenarios. The authors have proposed a set of steps to create a open integrated system to access the general retrofit data:

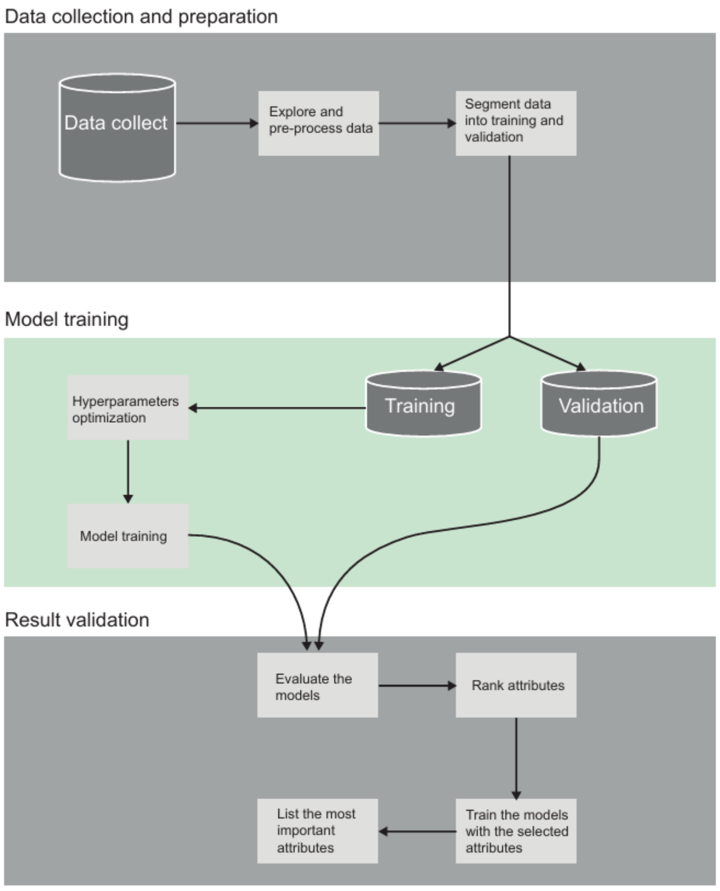
1. Establish an open data source framework for retrofit projects;
2. Embrace semantic web technologies;
3. Focus on building performance;
4. Include stakeholders in the retrofit process;
5. Increase the use of AI applications;
6. Conduct further research.

These recent studies show the shift of the industry towards data-driven and AI-based approaches in building energy efficiency and retrofit projects. As phenomenons in buildings are physical and therefore mathematically and statistically approachable, AI and specially machine learning can enable modeling the expected behaviors of buildings, allowing the designers to learn, adapt, and have fast answers, having more scientific support to create more sustainable buildings. Therefore, data-driven solutions can be seen as tools that, when combined with the human analysis capabilities and creative solutions, can maximize a building's energy performance.

3 METHODOLOGY

The methodology of this project is divided into two phases: data exploration and preprocessing of the databases and the creation of the prediction models and their analysis. The workflow is described in the Figure 26:

Figure 26 – Workflow methodology


Data collection and preparation
Data collect
Explore and pre-process data
Segment data into training and validation
Model training
Hyperparameters optimization
Training
Validation
Model training
Result validation
Evaluate the models
Rank attributes
List the most important attributes
Train the models with the selected attributes

Source: Balczareki (2024) and Uriona-Maldonado, Vaz, and Zaghi (2024)

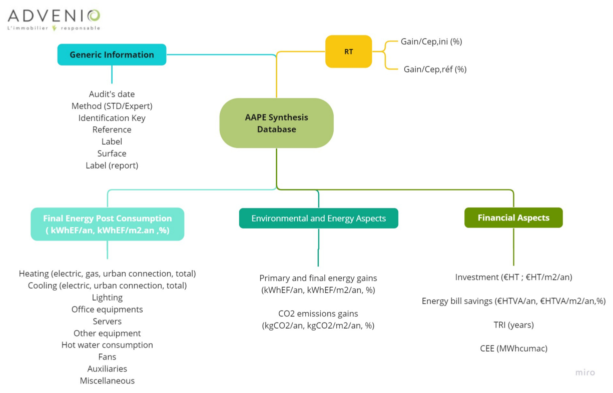
The data used in this study is acquired by the audits. The methodology begins gathering and preprocessing two different databases to prepare the data for machine learning. After the treatment of the data, which involves standardization, manual cleaning and many other procedures, the data is divided in training and test. Then, machine learning models, with optimized parameters, are trained and then evaluated for performance through a robust search of the best hyperparameters. When the performance achieve a satisfactory performance, SHAP analysis is conducted to identify and rank the most important features. This ranking is checked to verify if all the variables in it are coherent from a physical point of view and are in a reasonable quantity, if the features are relevant and not too numerous, the model is selected to receive some tuning to verify the possibility of its improval to finally get to the best model. This process is repeated for each one of the 12 models created in this project.

3.1 STEP 1: EXPLORATORY ANALYSIS AND PREPROCESSING DATABASES

3.1.1 First database exploration (Retrofit Synthesis database)

The Retrofit Synthesis database contains the results of the retrofit energy savings of the data, subdivided in five classes: Generic information, final energy by post of consumption, environmental, energy and financial aspects, shown in Figure 27.

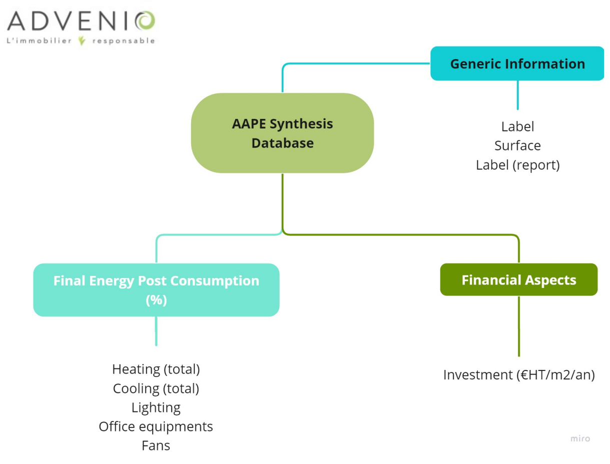
Figure 27 – Retrofit Synthesis database


ADVENIO
L'immobilier responsable
Generic Information
Audit's date
Method (STD/Expert)
Identification Key
Reference
Label
Surface
Label (report)
RT
Gain/Cep,ini (%)
Gain/Cep,réf (%)
AAPE Synthesis Database
Final Energy Post Consumption (kWhEF/an, kWhEF/m2.an ,%)
Heating (electric, gas, urban connection, total)
Cooling (electric, urban connection, total)
Lighting
Office equipments
Servers
Other equipment
Hot water consumption
Fans
Auxiliaries
Miscellaneous
Environmental and Energy Aspects
Primary and final energy gains (kWhEF/an, kWhEF/m2/an, %)
CO2 emissions gains (kgCO2/an, kgCO2/m2/an, %)
Financial Aspects
Investment (€HT ; €HT/m2/an)
Energy bill savings (€HTVA/an, €HTVA/m2/an,%)
TRI (years)
CEE (MWhcumac)
miro

Source: Author

However, as complete as this database may seem, its Identification key does not correspond to any other database. To use this database, a filter is applied:

Figure 28 – Retrofit Synthesis database filtered



Source: Author

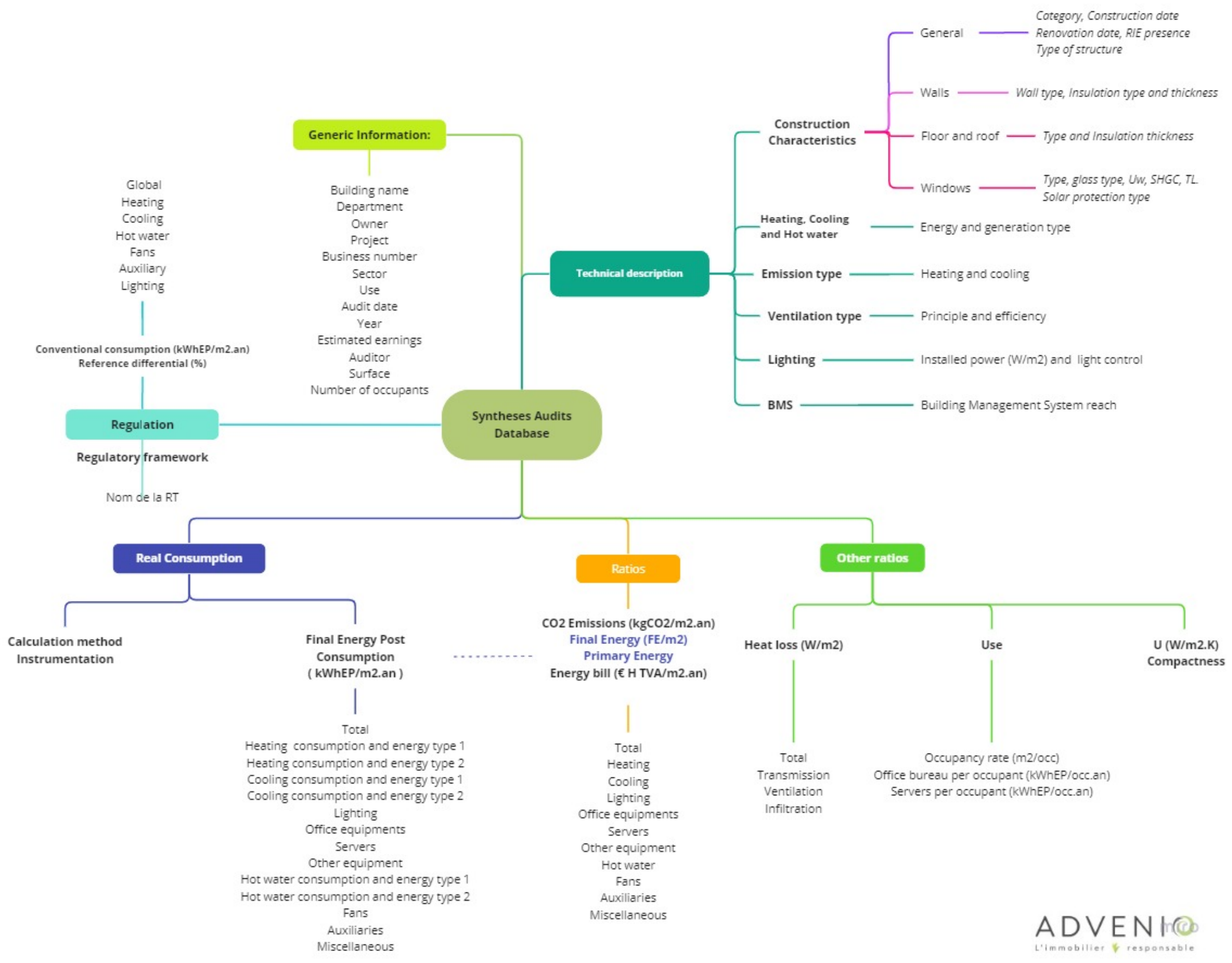
As shown in the Figure 28 above, there's only 9 columns retained, they are: surface, the retrofit description and energy savings to heating, cooling, lighting, bureau equipments and ventilation systems. RT (thermal regulation) information, Environmental e most of the economic aspects are not used either. As these would be mostly characteristics to be predicted, we keep them outside the model.

3.1.2 Second database exploration (Audit’s synthesis)

This database is one of the most important ones. It has 133 columns and 504 buildings. There are 13 columns that register the generic information about the buildings. Another 15 columns that relate the building with a reference. 22 columns that describe the building final energy consumption and its calculation method. We also have for final energy, primary energy, CO2 emissions and energy bill, 11 columns that represent the ratio kWh/m²/year to each one of them.

Beyond that, there are 4 columns that describe the building’s heat loss, 3 to describe occupation characteristics, 5 to describe general characteristics of the envelope, 13 to describe the walls, windows, floor and roof, 12 columns to describe the building’s heating, cooling, hot water, emission, ventilation and lighting systems and the last column shows the reach of the BMS (Building Management System) in the building. The Figure 29 below shows the general scheme of the database:

Figure 29 – Audit’s synthesis database



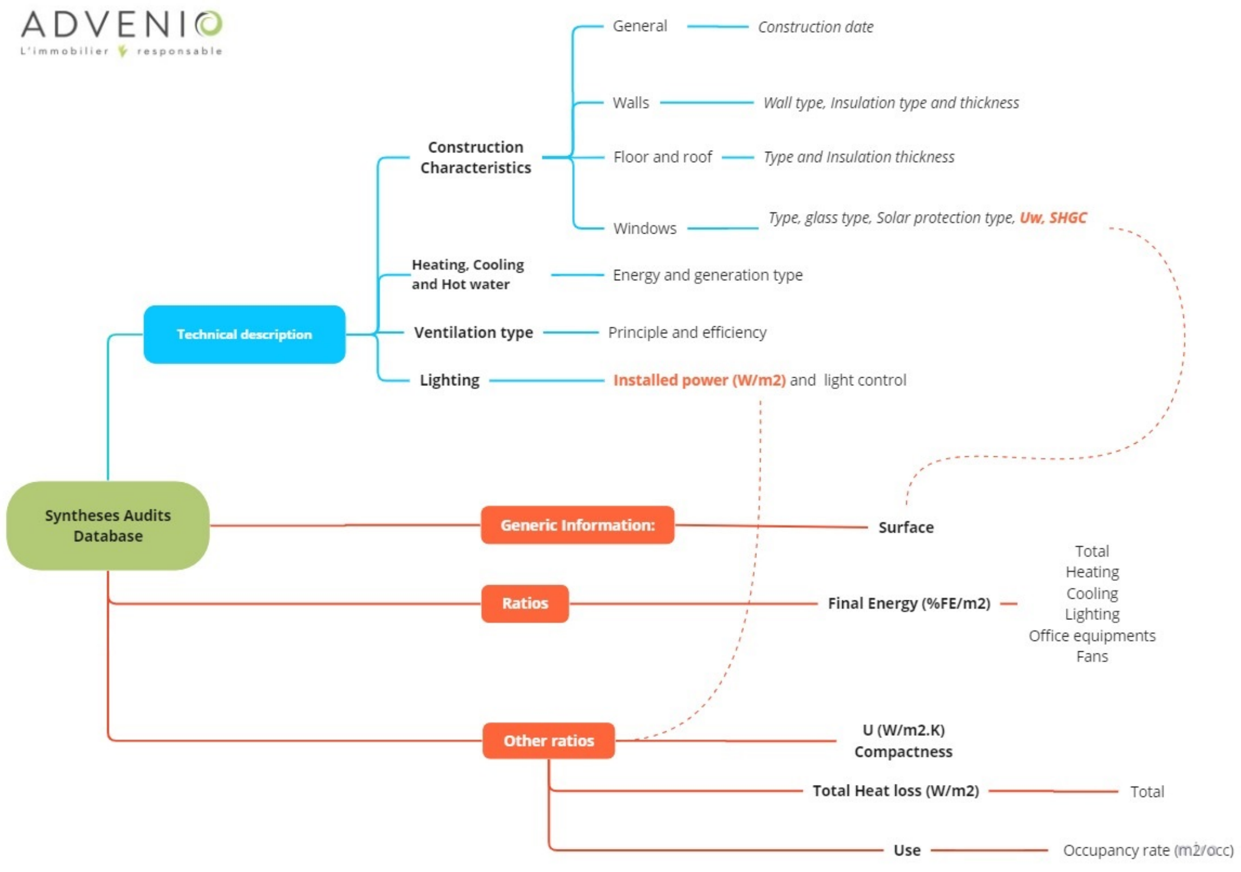
Source: Author

To use this database, a filtering is made to the sake of a greater homogeneity in the data, retraining the location of the buildings only to Paris and its suburbs. Also, the methods are restrained to simulation. These restrictions are particular important to the model development because it restrains the cases to those which are made by thermal simulation under the same or very similar weather data files (EPW).

A large and reliable database is very desirable to make a model, mainly a huge amount of cases. However, a large amount of characteristics to each case may turn the model utilization very difficult, because the characteristics we use to the training are those one required from the user. In practice, it is desired the larger amount of rows and the smaller number of columns as possible. Therefore, in order to simplify the model, a reduction of the number of columns is made.

Therefore, a selection of the most relevant or desired columns to be part of the model are made. The identification columns, the regulation column, all the columns that describe the consumption beyond the final energy in percentage terms, all the detailed heat losses and others that were not judged useful by the company's employees. The remaining columns are classified between quantitative and qualitative:

Figure 30 – Synthesis Audits database



Source: Author

The figure illustrates the structure of the "Synthesis Audits Database" for building performance analysis, highlighting connections between various data categories. It begins with a "Technical Description" section, detailing construction characteristics such as wall, floor, roof, and window properties, as well as heating, cooling, hot water, ventilation, and lighting specifications.

Information flows into the "Generic Information" category, where parameters like surface area and installed lighting power are collected. "Ratios" are calculated for metrics such as Final Energy (WFE/m²) across heating, cooling, lighting, and other systems. Additional performance metrics, termed "Other Ratios," include building compactness, total heat loss, and occupancy rates, providing insights into energy efficiency and usage patterns. These values are going to be connected through surface to the first database.

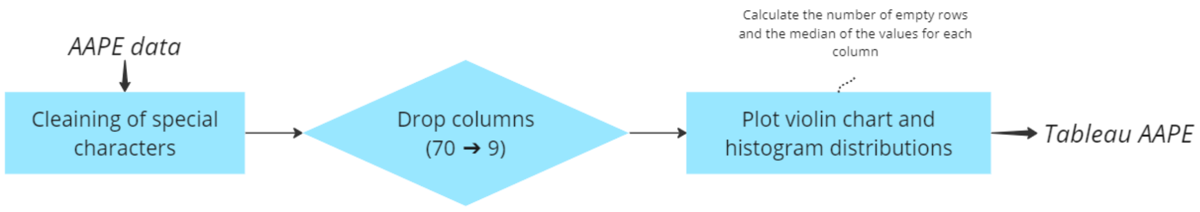
To fill up the empty spaces in this database, Fill method is used, which has a simple application and great indicators. Beyond that, it is important to pay attention if the data can really be replaced or if the feature demands an insertion of real data or its exclusion of the database.

First, both AAPE Analysis and Synthesis Analysis are made. These are the database treatments, so they will be cleaned and ready to be put together in the Assembling code. After that, an standardization of retrofit names is made, because between 1993 different retrofit actions, the majority of them have different names. After the standardization, filtering and correlation analysis are made, converting textual features into numeric ones, which can be understood by the model.

3.1.3 Preprocessing of first database (Retrofit Synthesis)

To the Retrofit Synthesis database, the data exploration is shown in Figure 31:

Figure 31 – Simplified Algorithm Code - Retrofit Actions Synthesis Analysis



AAPE data
Cleaining of special characters
Drop columns (70 ➔ 9)
Calculate the number of empty rows and the median of the values for each column
Plot violin chart and histogram distributions
Tableau AAPE

Source: Author

First, the data is imported and special characters that Python isn't capable of comprehending are substituted. After that, we choose the columns to be dropped and calculate the number of null rows, plotting the graphs with the medians to each column also. From 1,6 Mb and 70 columns, described in the Figure **??**, the selection of the most interesting columns generates a simpler file of only 0,2 Mb and 9 columns, described in the Figure 28.

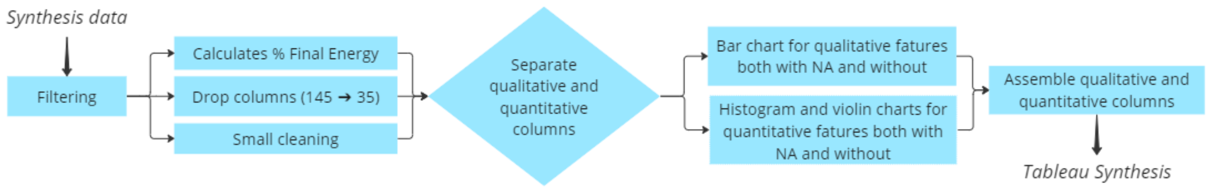
3.1.4 Preprocessing of second database (Audit's synthesis)

First, we extract the entire data directly from the original database. It has around 700 KB, which comprises 133 columns and 504 different buildings, that were described in the Figure 29 of the previous section.

From this, manually, 1 column was added to describe the audit's year and 11 columns added in order to regroup some of the features in each column [1]. The features that had some sort of regrouping were: wall, window, glass, shading, roof, floor, heating, cooling, ventilation principle, light control and the period of construction. Another column is added to indicate the audit's year.

After dealing with the data manually in order to know it carefully, we start the big manipulations, which requires the intensive use of coding. The entire database was treated with Python code, whose principles are described in the Figure 32 below.

Figure 32 – Simplified Algorithm Code - Synthesis Analysis


Synthesis data
Filtering
Calculates % Final Energy
Drop columns (145 → 35)
Small cleaning
Separate qualitative and quantitative columns
Bar chart for qualitative fatures both with NA and without
Histogram and violin charts for quantitative fatures both with NA and without
Assemble qualitative and quantitative columns
Tableau Synthesis

Source: Author

First, the data corresponding only to offices are selected. This choice is made because the database comprises more than 20 different types of usages, however the occupation may impact very much the consumption of the building, and it is better to keep the study restricted to similar occupations. Beyond that, offices represents around 58% of all cases having a good amount of cases for the model development.

Among these offices, we select those in the Paris and suburbs region, which corresponds to the 75,92,93 and 94 departments, representing 80% of all offices locations. That is extremely important because the model uses simulation results, which are based in weather data Energy Plus Weather file (.EPW). Therefore, the retrofit impact won't be influenced by the .EPW file.

The simulation files used are those comprised in three different methods, which are 'CW', 'VE' and 'P+C'. The first simulation method is CW, which is a pre-simulation method used by the company years ago. VE corresponds to Virtual Environment, which is a high quality software known internationally. finally, we use of P+C (Pleiades + Comfie), well known in the French market. After that, we calculate the percentage of final energy savings to all retrofit improvements.

[1] This regrouping was carefully made with the help of the company's experts and does not exclude any information, keeping the original data in the dataset to further verifications.

Also, it is necessary to drop many columns. This process describes the cleaning that was pointed out in the previous Section and described in the Figure 30. To deal with empty rows, the *Fill method* described in 3.1.2 is applied to fill the empty rows, being considered a statistic reliable method, even tough having the correct real data would be evidently better.

To the qualitative variables, the analysis will be presented in bar charts, indicating what values were inserted to every feature of the data with a bar plot comparison. For the quantitative variables, there are histograms to each and every feature, also comparing the distributions with and without the filling of the empty rows. The result analysis will be detailed in the section Results.

3.1.5 Preprocessing : join both databases

As shown, after the first two treatments we end up with two different databases; however, as we are going to create a model to predict energy savings from retrofit outputs we need these two databases to be merged as a single one, using the common characteristics between them. The scheme of this procedure is described in the Figure 33 below.

Figure 33 – Simplified Algorithm Code - Assembling

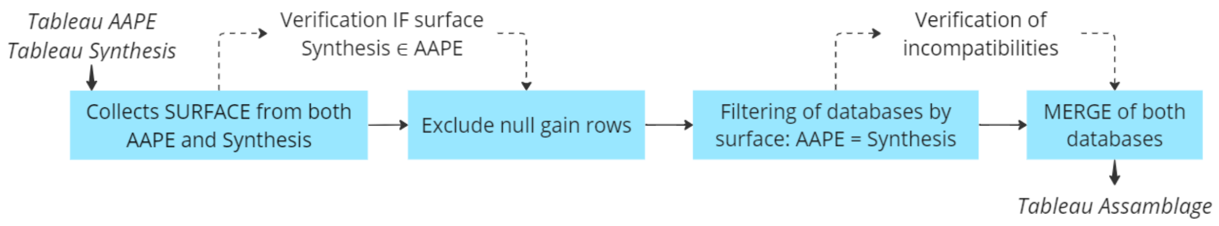


Tableau AAPE
Tableau Synthesis
Verification IF surface Synthesis ∈ AAPE
Verification of incompatibilities
Collects SURFACE from both AAPE and Synthesis
Exclude null gain rows
Filtering of databases by surface: AAPE = Synthesis
MERGE of both databases
Tableau Assamblage

Source: Author

First, both outputs from the two previous treatments are collected. From these two sources, we use the single element in common between them to merge the database. Luckily, as multiple verifications procedures showed, the building surfaces do not repeat themselves and are used as the common denominator to the assembling.

To effectively merge the data, we first verify which AAPE surfaces are contained in the Synthesis surfaces list. For those whose surfaces are verified, the merging is accomplished.

3.1.6 Preprocessing: Database standardization

After the Assembling Code, all the data is cointained in a single dataframe. However, the AAPE retrofit suggestions are many written in many forms, because they are inserted manually by each one of the auditors and specifically adapted to the case study upon which they worked. This multiplicity generates a need for standardization of the database, so these similar solutions will be grouped in categories, allowing the occurrence of sufficient cases to generate good statistics to make accurate predictions to each class. The procedure is described in the Figure 34.

Figure 34 – Simplified Algorithm Code - standardization Analysis

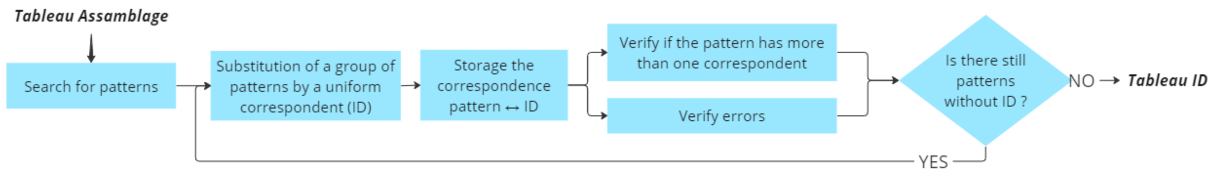


Tableau Assamblage
Search for patterns
Substitution of a group of patterns by a uniform correspondent (ID)
Storage the correspondence pattern ↔ ID
Verify if the pattern has more than one correspondent
Verify errors
Is there still patterns without ID ?
NO
Tableau ID
YES

Source: Author

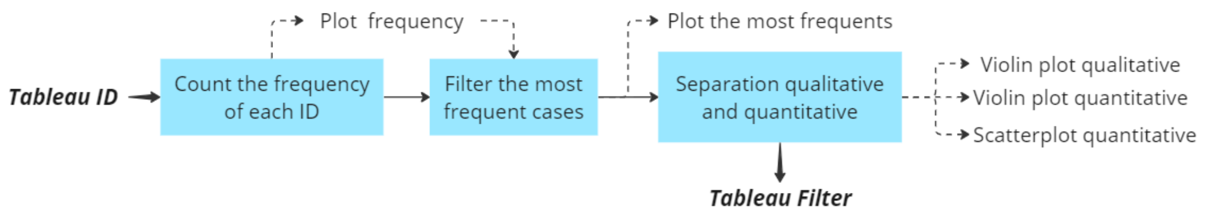
First, the data is manually searched for patterns, then these patterns are written down to pass through a massive grouping storing its respectively ID in a new column. After that, a search is made to verify if there are any retrofit suggestions that belong to existing ID categories that were not covered by the search and insert its pattern to its inclusion in the grouping.

This process is repeated many times until there is a good convergence of all or almost all patterns are attributed to an ID. The easiest way to do this would be using a classification model rather than by hand. However, due to the time limitations of this study and the small amount of data, that does not justify the application of sophisticated techniques as such, we are going to keep it simple.

3.1.7 Preprocessing: Database Filtering and Correlation

After all the procedures, including the standardization, we have sufficient mature data to make deeper analysis, mostly in the retrofit level rather than the previous building level. The general process of this step is described in the Figure 35.

Figure 35 – Simplified Algorithm - Filtering and Correlation Analysis



Source: Author

First, the frequency of each AAPE is verified in all the cases that were kept. Then, as there are very low frequent suggestions that wouldn't be statistically significant to the model's development, we make a filtering to keep only the retrofit suggestions that have more than 15 occurrences.

A substantial amount of data is a fundamental requirement to a good machine learning prediction. This need arises from the necessity to develop a model that can effectively generalize from the training data to make accurate predictions or decisions on new, unseen data. Larger frequency retrofit improvements provide a broader representation of the underlying patterns, enabling the model to handle the complexity and variability of actual data, extract relevant features, and reduce the risk of overfitting.

However, it's crucial to maintain a balance between data quantity and quality, ensuring that the data is representative and relevant to the problem at hand. While a substantial dataset is essential, it is not solely about the quantity of data; quality is equally crucial. A vast but noisy or biased dataset can lead to unreliable model performance, and this will be evaluated after the model is trained.

Overall, the number of retrofit suggestions are 739. Between these, 535 were classified. After filtering those with a frequency greater than 15, we have a total of 472 and eliminating those with zero saving, we have finally 460 retrofit actions in total to be integrated in the model. After selecting the most frequent cases, we step into the correlation analysis of all features against the overall final energy saving calculated for all the retrofit improvement. The formula of the overall saving is described below:

$$\text{Energy Saving (\%)} = \begin{pmatrix} \text{Heating Energy saving} & : pEFg_{\text{chauff}} \cdot pEF_{\text{chauff}} \\ \text{Cooling Energy saving} & : pEFg_{\text{refr}} \cdot pEF_{\text{refr}} \\ \text{Ventilation Energy saving} & : pEFg_{\text{vent}} \cdot pEF_{\text{vent}} \\ \text{Lighting Energy saving} & : pEFg_{\text{ecl}} \cdot pEF_{\text{ecl}} \\ \text{Bureau Equipment Energy saving} & : pEFg_{\text{bureautique}} \cdot pEF_{\text{bureautique}} \end{pmatrix}$$

The Total Energy saving is calculated in terms of final energy. This choice over primary energy is because we would like to have the savings without necessarily depending to which type of energy the building is submitted, which would be the case of primary energy.

This value is used to make an extensive correlation analysis with all features, the results are going to be explored in the Results section. For the quantitative variables, we make two different analysis, first a scatter plot and then a histogram and violin chart analysis. A complete analysis of the qualitative variables is also made through violin plots for each categorical variable of each column. However, these analysis are not presented for the sake of conciseness in this project.

3.1.8 Preprocessing: Feature Engineering

The pre-model phase is extremely important. It reflects basically the need of transforming textual sentences into normalized numeric correspondents. This is made through Python dictionaries.

Figure 36 – Simplified Algorithm Code - Pre modeling

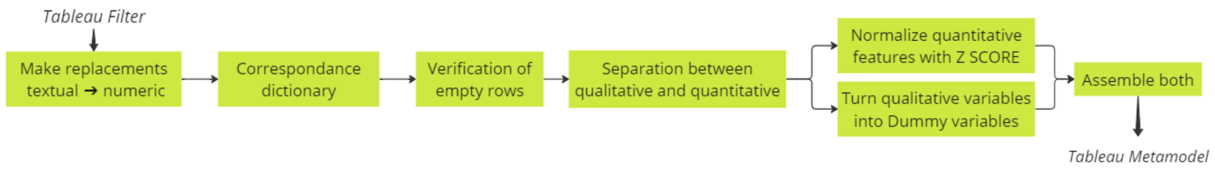


Tableau Filter
Make replacements textual → numeric
Correspondance dictionary
Verification of empty rows
Separation between qualitative and quantitative
Normalize quantitative features with Z SCORE
Turn qualitative variables into Dummy variables
Assemble both
Tableau Metamodel

Source: Author

First, we proceed to the storage of the variables in a dictionary. This is particularly important to understand and translate the results of the model after the training and make the sensibility analysis, but also to built an interface that is easily comprehensible by the users.

These dictionaries are storage and used in the interpretation of the data and the graphics that will be generated in the sensibility analysis after the model execution. The stored data is used to create functions that find the correspondent ID.

The second operation in the pre-model code is to verify if there are empty rows, for they prevent the right execution of the code. The second operation is to separate the features from the target. The features are the characteristics used to the prediction, and the target is the value the model is supposed to predict.

The features are the characteristics of the building and of the retrofit improvement data that were stored in the Tableau AAPE, described in the Results section. The target value is a single column that represents the overall saving of a retrofit improvement.

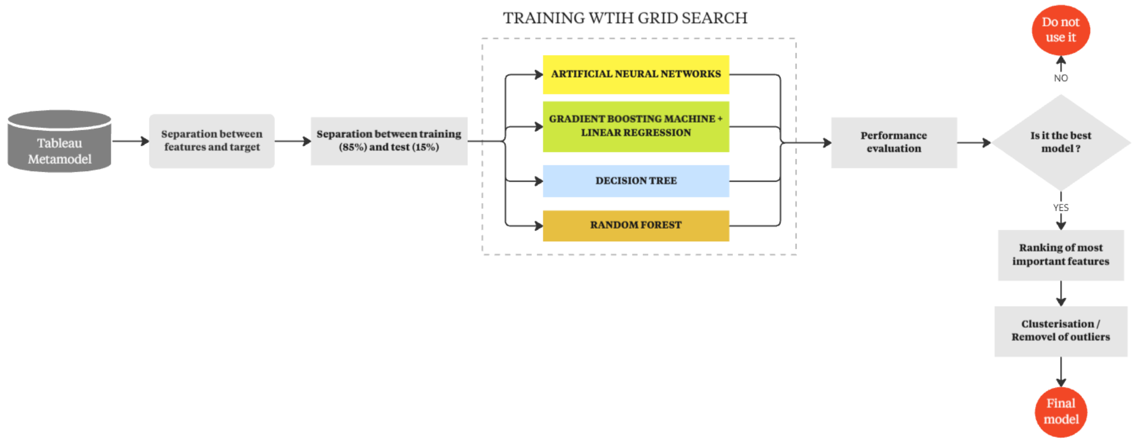
After that, both features and target are classified in qualitative and quantitative. This is necessary since artificial neural networks and gradient boosting machine do not make assumptions about the distribution of the data (INSTITUT MONTAIGNE, 2021). To the quantitative data, there is a normalization of the values and the qualitative variables are turned into binary columns.

In the case of the quantitative variables, we use the Min Max normalization function, one of the most common ways to normalize data. For every feature, the minimum value of that feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1. In the case of qualitative variables, a test to transform the values into binary columns (dummy variables).

3.2 STEP 2: MACHINE LEARNING MODEL

The model is the most important piece of code of this entire project. It contains all the training to the artificial network and the gradient boosting model. The workflow of the model is described in the Figure 37.

Figure 37 – Machine Learning Algorithm Code - model Workflow


TRAINING WTIH GRID SEARCH
Tableau Metamodel
Separation between features and target
Separation between training (85%) and test (15%)
ARTIFICIAL NEURAL NETWORKS
GRADIENT BOOSTING MACHINE + LINEAR REGRESSION
DECISION TREE
RANDOM FOREST
Performance evaluation
Is it the best model ?
NO
Do not use it
YES
Ranking of most important features
Clusterisation / Removel of outliers
Final model

Source: Author

The figure above shows the diagram that represents a machine learning pipeline for training and evaluating the 12 retrofit models, ultimately selecting the best-performing one as the final model. The process begins with a "Tableau Metamodel," which serves as the initial dataset. The first step in the pipeline involves separating the dataset into features (independent variables) and the target (dependent variable) to establish the variables used for model training. Once the features and target are defined, the dataset is split into training and testing sets, with 85% of the data used for training the models and 15% reserved for testing, ensuring that the model's performance can be evaluated on unseen data.

The training phase involves a grid search to optimize hyperparameters for four types of models: Artificial Neural Networks, Gradient Boosting Machine (GBM) combined with Linear Regression, Decision Tree, and Random Forest. Each model is trained and fine-tuned to find the best parameters for each algorithm, maximizing predictive accuracy and minimizing errors. This grid search approach allows systematic testing of various hyperparameter combinations, improving the likelihood of finding an optimal configuration for each model.

After training, the models undergo a performance evaluation, where metrics are calculated to determine how each model predicts the target variable. If a model is deemed unsuitable based on these evaluations (i.e., its performance does not meet a predefined threshold), it is discarded. For models that meet the performance criteria, the best-performing one is selected.

This model is then subjected to further analysis to identify the most influential features, which provides insights into the factors most relevant to the predictions and can inform decision-making. Then, the model's most important features are ranked to assess their impact on the outcome. Next, a clustering or outlier removal process is applied to clean the data, enhancing model robustness by ensuring that extreme values do not distort predictions. Once these refinements are complete, the final model is ready for deployment, optimized with essential feature information and free from the influence of outliers.

3.2.1 Training

Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples. A machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called empirical risk minimization (GOOGLE DEVELOPPERS, 2022). This process is made through hyperparameter tuning. After training and fine-tuning the model, data from the test set are used to evaluate the model for ability to generalize (MALEKI et al., 2020a).

This project uses the Scikit-learn library, one of the most popular machine learning libraries of machine learning. Scikit-learn is largely written in Python, and uses NumPy extensively for high-performance linear algebra and array operations.

The hyperparameters of Artificial Neural Networks:

- **Hidden layers size:** Defines the architecture of the neural network. It specifies the number of neurons (or units) in each hidden layer of the network. Deeper networks with more neurons can capture more intricate features in the data but might be prone to overfitting.
- **Activation function:** Has the goal of introducing non-linearity to the problem, it transforms the weighted sum of inputs at each neuron into an output. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh. The choice of activation function impacts the network's ability to approximate non-linear functions and its convergence during training.
- **Batch size:** Determines how many training examples are used in each iteration of training the neural network. During training, the dataset is divided into smaller batches. Training on batches, rather than the entire dataset, speeds up the learning process and can lead to more stable convergence.
- **Learning rate:** Determines the step size at each iteration when adjusting the model's weights during training. A high learning rate can cause the model to converge quickly but may overshoot the optimal solution. A low learning rate can help convergence but might require more training iterations. It's a critical hyperparameter to tune, as an inappropriate learning rate can lead to slow convergence or instability.

- **Maximum iterations:** Determines the number of iterations made in the code, specifying how many times the entire training dataset is passed through the network during training. It ensures that the model has sufficient opportunities to learn but setting it too high might lead to overfitting, so it should be chosen carefully.
- **Tolerance level:** It specifies the stopping criteria for training. Tolerance level is often used with early stopping to prevent overfitting. If the loss function improvement is less than the tolerance level for a certain number of iterations, training is terminated. This helps avoid training the model for too long and potentially overfitting to the training data.
- **Random state:** By setting the random state, you can ensure that the randomness is reproducible, crucial for obtaining consistent results and comparing different training runs.

The parameters of the GBM are described below:

- **Depth Maximum:** This parameter determines the maximum depth or the maximum number of layers a single decision tree in the gradient boosting ensemble can have. A smaller depth maximum value can lead to simpler trees, which are less prone to overfitting, while a larger value allows for more complex trees.
- **Number of Estimators:** The number of estimators, often represented as trees, in the gradient boosting ensemble. Each estimator is added sequentially to the ensemble, and they are trained to correct the errors made by the previous ones. A higher number of estimators typically leads to a more robust and accurate model.
- **Minimum Number of Samples:** This parameter sets the minimum number of samples required to split a node further in the decision tree. It controls the granularity of the splits in the tree. Smaller values may lead to more detailed splits, while larger values may result in coarser splits.
- **Learning Rate:** The learning rate determines the step size at which the model adapts to minimize errors. A smaller learning rate requires more iterations (i.e., more trees) to converge, but can lead to better generalization and avoid overfitting. A larger learning rate can speed up convergence, but may lead to overfitting.
- **Loss:** This parameter specifies the loss function to be minimized during training. Different loss functions are used to optimize different objectives. Common loss functions are Quantile and Huber. Quantile is used for quantile regression, modeling different quantiles of the target distribution. Huber is a robust loss function that combines the advantages of mean squared error (MSE) and mean absolute error (MAE) loss functions. It is less sensitive to outliers.
- **Random State:** This parameter is used to set the random seed, ensuring the reproducibility of results. When you set the random state to a specific value (e.g., 0), it ensures that the random initialization and shuffling of data are consistent across runs, making the results reproducible.

The parameters of decision tree:

- **Maximum Depth**: Limits the depth of the decision tree, where greater depth allows the model to capture more complex patterns. Excessive depth may lead to overfitting.
- **Minimum Samples for Split**: Defines the minimum number of data samples required to perform a split at any node. Higher values encourage splits only when substantial data is available, reducing the risk of overfitting.
- **Minimum Samples per Leaf**: Sets the minimum number of data samples that must be present in each leaf node. Requiring more samples in the leaves helps prevent the model from learning from noise, promoting generalization.
- **Maximum Features Considered**: Controls the number of features considered when determining the best split at each node. Options include using all features, the square root of the total features, or the logarithm of the total. Limiting features per split helps reduce model variance.
- **Maximum Number of Leaf Nodes**: Sets an upper limit on the number of leaf nodes, which restricts the tree's growth. Limiting leaf nodes can simplify the tree structure and reduce overfitting.
- **Random Seed**: Fixes the seed for random processes, ensuring consistent results across runs. This does not affect the model's structure but provides reproducibility in the results.

The hyperparameters of random forest:

- **Number of Estimators**: Specifies the number of decision trees in the forest. Higher numbers can improve performance by averaging out errors across multiple trees, but this also increases computational cost.
- **Maximum Depth**: Works the same way as in the decision tree model.
- **Minimum Samples for Split**: Functions the same way as in the decision tree, determining the minimum number of samples required to split an internal node.
- **Minimum Samples per Leaf**: Similar to the decision tree, this parameter sets the minimum samples needed in each leaf node to avoid learning from noise and improve generalization.
- **Random Seed**: Provides a fixed seed for the random processes within the model, ensuring consistent results. This parameter is also identical in purpose to the decision tree's random seed.

Also, the GBM model may be improved through a method of correction, which creates a Combined Model, submitted to posterior SHAP analysis.

The formula for the adjustment is based on slope m and intercept c :

$$m, c = \text{argmin}_{m,c} \sum_{i=1}^{n} (y_i - (m\hat{y}_i + c))^2$$

where y_i represents the actual observed values, and $\hat{y}_i$ denotes the model's predictions for the training data. Once m and c are estimated, we can adjust the predicted values $\hat{y}$ to obtain a corrected prediction $\tilde{y}$ using the following transformation:

$$\tilde{y} = m\hat{y} + c$$

This corrective adjustment reduces the systematic underprediction and improving the model's alignment with the ideal prediction line ($y = \tilde{y}$). This approach mitigates the bias introduced by the initial deviation in slope and enhances the model's accuracy.

Therefore, the `CombinedModel` incorporates the two key components: a `GradientBoostingRegressor` model (denoted as `gbm_model`) and a linear transformation applied to the `gbm_model` predictions. This transformation modifies predictions with an adjustment based on a slope and intercept derived from a linear fit between actual target values and initial predictions. Thus, the final prediction is an altered form of the raw `gbm_model` output, as implemented in the model's `predict` function.

3.3 STEP 3: HYPERPARAMETERS OPTIMIZATION

Hyper-parameter tuning is a critical process that significantly influences the accuracy and performance of machine learning models. To streamline this process, sensitivity analysis offers a valuable quantitative framework that enables the ranking of hyper-parameters based on their individual contributions to model accuracy (TAYLOR et al., 2021). This approach not only identifies which parameters are most impactful but also assists in prioritizing those that warrant closer attention during tuning.

In practice, the insights gained from sensitivity analysis are seamlessly integrated into the model training phase through the use of the *GridSearchCV* function, which is part of the *Scikit-Learn* Python package. This powerful function automates the process of evaluating various hyper-parameter combinations by employing rigorous cross-validation techniques. Specifically, the best regression model is chosen based on its cross-validation performance, with the mean score calculated on a validation dataset for all possible combinations of hyper-parameters explored exhaustively.

To summarize, *GridSearchCV* systematically trains a series of hyperparameter combinations that are carefully selected and then identifies the most effective combination among them. This ensures that we obtain the result of the best-performing model directly, without the need to manually sift through the results of all combinations.

3.4 STEP 4: VALIDATION

The testing data set is a separate portion of the same data set from which the training set is derived. The main purpose of using the testing data set is to test the generalization hability of a trained model (MALEKI et al., 2020b).

Statistical indicators are crucial for gauging the effectiveness of regression models, offering insights into their performance and predictive accuracy. R^2, known as the Coefficient of Determination, is a key metric that quantifies how well the model accounts for the variance in the dependent variable.

RMSE, or Root Mean Square Error, measures the typical size of the errors between predictions and actual data. It's highly sensitive to outliers, and lower RMSE values indicate more precise predictions. MAE, or Mean Absolute Error, takes the absolute value of errors and provides an average measure of their magnitude.

It's less influenced by extreme values and is useful when error size is a priority. AE95, which stands for Absolute Error at the 95th Percentile, is a statistical indicator designed to focus on the upper end of the error distribution. In other words, it specifically targets the scenarios where errors tend to be at their largest or most extreme.

3.5 STEP 5: RANDOM STATE AND CROSS VALIDATION TRUNKS

In order to make the results more robust, the final model combines both cross-validation and multiple random states to thoroughly improve model performance. Cross-validation is essential in this process, as it helps mitigate issues like overfitting and underfitting, particularly when working with small datasets. By systematically dividing the data into training and testing subsets across multiple folds, cross-validation ensures that every data point has a chance to be used for both training and testing, allowing for a comprehensive assessment of the model's stability and performance.

In this analysis, a 10-fold cross-validation is implemented through ‘GridSearchCV‘ with a grid of hyperparameters, scoring based on R^2, and parallel processing to speed up the search. This process identifies the best combination of hyperparameters by averaging model performance across the 10 folds, reducing the risk of overfitting to a particular subset of data and helping achieve an optimal bias-variance balance.

Additionally, a range of 10 predefined random states is used alongside cross-validation to introduce further variability in data splits. This technique helps simulate multiple "views" of the dataset, enhancing the reliability of the model evaluation by ensuring that results are not overly dependent on any single random split.

The selected random states allow each model configuration to be trained and tested across different splits, providing a more robust measure of the model's generalization ability. This combined approach of cross-validation and varying random states allows for a well-rounded analysis, helping to identify a model configuration that performs consistently well across different data arrangements.

3.6 STEP 6: SHAP SENSITIVITY ANALYSIS

In SHAP analysis, the goal is to examine the feature importance for each one of the 12 models, within the context of predicting subsets defined. For each AAPE value, the process reproduces the same conditions as the original training by loading a pre-trained combined model that specifically targets this value. This approach ensures consistency with the model's training environment, as the subsets of data used in SHAP analysis mirror the exact configuration used in model training.

The method follows a precise setup: first, it isolates the relevant subset of features based on the specific AAPE value's associated parameters. Next, it performs a train-test split on this subset, using a unique random state for each AAPE to avoid data leakage and ensure reproducibility. By applying the SHAP TreeExplainer to the Gradient Boosting Machine (GBM) component of the combined model, the analysis generates SHAP values, which quantify the impact of each feature on the model's predictions.

In utilizing it, SHAP captures the complete prediction output, which incorporates both the `gbm_model`'s base predictions and the applied linear transformation. This ensures that SHAP values reflect the contributions of features to the fully transformed predictions, not solely the raw predictions from `gbm_model`.

3.7 STEP 7: CLUSTERISATION

The clusters allow for a structured assessment of energy efficiency and overall performance. The following clusters have been established:

- **Envelope Cluster:** Variables related to the building's thermal envelope, including construction materials, wall structures, and insulation types. The variables in it are: year of construction, type of wall, type of wall insulation, thickness of wall insulation, type of joinery, type of glazing, type of upper floor, thickness of insulation, type of lower floor, thickness of lower floor.

- **Lighting Cluster:** Variables associated with lighting systems, such as power ratings and performance factors. The variables are: Lighting management, percentage of consumption for lighting.

- **Heating Cluster:** Variables defining the type and method of heating systems employed in the building. Combined by: Type of heating, and energy source of heating.

- **Cooling Cluster:** Variables related to cooling systems and their characteristics. Composed by: Type of cooling, and energy source of cooling.
- **Ventilation Cluster:** Variables pertaining to ventilation systems, focusing on their principles and efficiency metrics. Composed by: principle and efficiency of ventilation.

Overall, these clusters facilitate targeted analyses of the main aspects of building energy performance, as it regroups complementary characteristics, allowing an easier identification of these patterns and therefore, possibly enhancing a more precise model.

3.8 STEP 8: REMOVAL OF OUTLIERS

Removing outliers is a critical preprocessing step in enhancing the performance of statistical models. Outliers can distort model accuracy and lead to misleading results. A robust method for identifying outliers involves using the interquartile range (IQR).

The IQR is calculated as:

$$\text{IQR} = Q_3 - Q_1$$

Where Q_1 is the first quartile (25th percentile) and Q_3 is the third quartile (75th percentile). This measure helps to understand the spread of the central 50% of the data.

In this project, outliers can be defined using the following bounds:

$$\text{Lower Bound} = Q_1 - 1.5 \times \text{IQR}$$

$$\text{Upper Bound} = Q_3 + 1.5 \times \text{IQR}$$

Any data point x_i that lies outside these bounds is considered an outlier:

$$x_i \text{ is an outlier if } x_i < \text{Lower Bound or } x_i > \text{Upper Bound}$$

The filtered dataset, which excludes outliers, is represented as:

$$x_{\text{filtered}} = \{x_i \in D : \text{Lower Bound} \leq x_i \leq \text{Upper Bound}\}$$

Where D is the original dataset. This process helps in refining the dataset, leading to improved model performance and more reliable predictions.

4 RESULTS

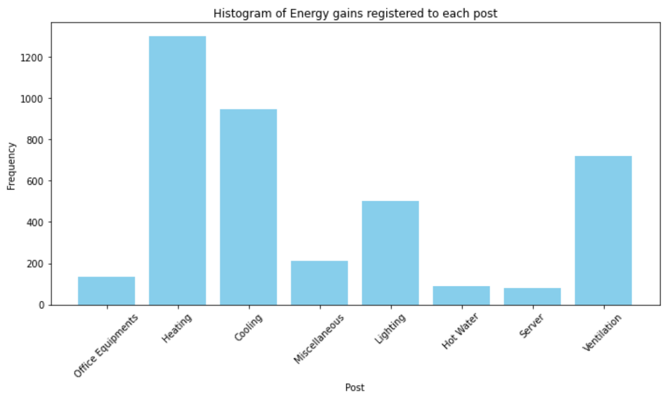
4.1 STEP 1: EXPLORATORY ANALYSIS AND DATA PREPROCESSING

4.1.1 Exploratory analysis and data preprocessing of first database

The first database to be analyzed is the Retrofit Synthesis, it is a relatively small database, but very important as it registers all the information regarding the retrofit suggestions, containing 1994 retrofit suggestion covering all typologies.

In a first visual and manual analysis of this database, it was noticeable that the column's percentage of final energy energy savings, a representation of energy efficiency, had some problems, showing values greater than 100% for office equipment savings that are not related to the retrofit suggestion described, such as ventilation. Therefore, all the percentage and ratios of end-use savings were recalculated. Beyond that, some other incoherences were found in around 1% of the data. Those which were easily identified were right away corrected, but carefully regarded as indications that a revision is needed. The result of the cleaning is shown in Figure 38 below.

Figure 38 – Frequency of retrofit suggestions by end-use



Source: Author

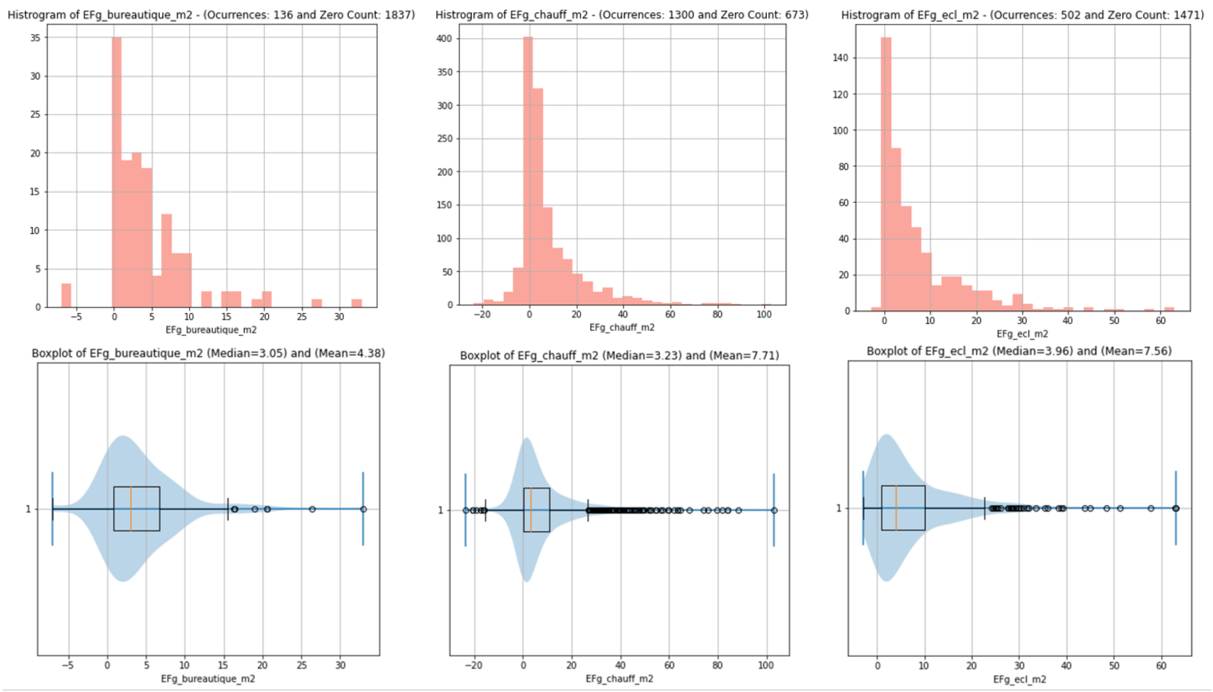
The graph represents which end-use is impacted by the retrofit. For example, by reducing the utilization of office equipment, turning them off when not used, won't only direct reduce the office equipment consumption but also reduce the cooling and increase the heating indirectly, due to less Joule effect dissipation. Therefore, a single retrofit suggestion can have multiple classes of impact.

As we can see in the graph, most of the classes impacted are heating and cooling, followed by ventilation. These classes are directly related to the french climate. Most of the cases are in Paris. [1].

[1] Paris climate is at the boundary between continental and oceanic solid influences, with lower precipitation compared to the mean of the country (between 500 and 800 mm against 900 mm; Canellaset al., 2014). Summers are relatively hot (18.8℃) and winters mild (4.4℃). Some studies focused on the urbanization and found different impacts such as a mean urban heat island of 3℃ over the period 1971–1980 with maxima exceeding 10℃ (LE ROY et al., 2020)

The primary energy saving by the square meter to each end-use is very similar between heating, lighting and office equipment and to note most of the occurrences happen between 0 and 20 kWh/m^2, even if the graphs are in different scales to preserve a good visualization.

Figure 39 – Frequency of retrofit suggestions by end-use



Source: Author

This may be explained by the fact heating and cooling are often affected by improvements planned to other end-uses retrofits, while lighting, which could be easily underestimated but represents a good retrofit in general, is often alone. Ventilation has a similar distribution, with median value of 1.92 kWh/m² and mean 4.76 kWh/m².

The cooling savings occurs in 50% of the cases, with a small mean of 2.03 kWh/m² and median of 0.65 kWh/m², smaller than the savings of hot water that have a mean of 2.81 kWh/m² and median of 1.09 kWh/m², showing that the consumption of hot water has a lot of potential of improvement compared to the cooling system. A hypothesis is that Paris does not show great need, neither have the culture of installing air conditioning even in offices.

The server savings have a near uniform distribution, with mean of 3.29 kWh/m² and median of 0.01 kWh/m². The miscellaneous consumption has a mean of 4.4 kWh/m² and median of 0.38 kWh/m². Finally, the investment per square meter has a mean of €34 and median of €11.50. But there are one third that do not cost anything. The most expensive investment rises up to €500/m².

4.1.2 Exploratory analysis and data preprocessing of second database

The Audit's synthesis database presents the information about the buildings and where they are located, allowing a more narrowed and profound analysis focused on the cases we want to address in this project. The first thing to do is to reduce the analysis to offices, excluding all the other typologies from the analysis. Another simplification is to choose only Paris and its suburbs [2]. After that, we select the desired columns described in Section 3.1.2 by the procedures described in 3.1.4 to generate the following analysis.

Figure 40 – Heating energy source

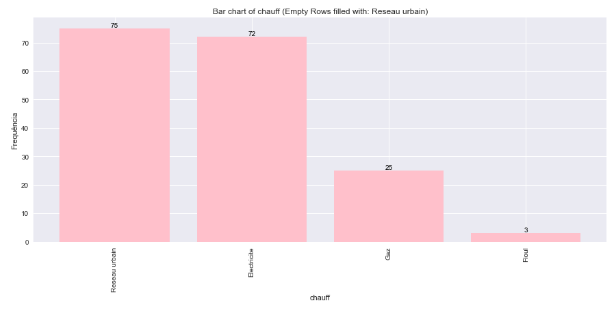
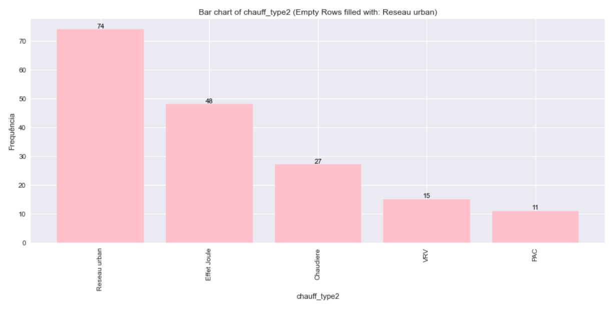


Figure 41 – Heating energy production



Source: Author

The urban heating network can be a household waste incineration plant (UIOM), a boiler room powered by fuel (oil, gas, wood, etc.), a deep geothermal power plant, etc. It is a centralized heat distribution system through pipes in which the heat is transported by a heat transfer fluid mixed up with water at ambient temperature in the buildings.

The electricity heat production is given by Joule Effect, VRF, electrical heat pumps or even boilers. The joule effect occurs during the passage of electric current through a conductor presenting resistance, such as electric heaters, radiating beams, etc. Regarding VRF, heat exchange occurs through the circulation of refrigerant, through which heat is extracted from the outdoor air and transferred to the indoor spaces using a compressor, which adjusts refrigerant flow and temperature as per the specific heating or cooling demands of each indoor zone.

A heat pump extracts heat from a lower-temperature source (such as outdoor air or the ground) and releases it into a higher-temperature space, using a refrigerant cycle to facilitate this heat transfer. A boiler works by using the correspondent energy source to heat water or another fluid to generate steam or hot water. The heated water or steam is then circulated through pipes to radiators or underfloor heating systems.

Beyond that, there is the gas energy production. This energy source has good energy efficiency and environmental advantages: its combustion does not emit dust, little sulfur dioxide (SO_2), little nitrogen oxide (NO_2) and less carbon dioxide (CO_2) compared to others fossil fuels (IFP, 2023). It is transported through pipelines, serving as a combustion source to power the boilers or gas heat pumps.

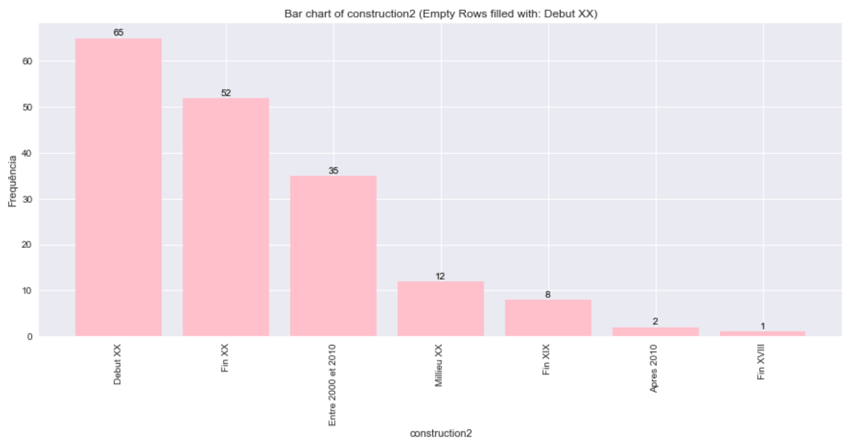
[2] The original database has 509 buildings, this filtering reduces the cases to 209.

The same analysis is made to the cooling, which shows that around two thirds of cooling in generated by electricity, followed by one third generated by urban cooling network and just one case by gas. Regarding the energy production system, the most frequent is the heat pump, then the cooling network, and finally VRV and aerorefrigerant tower. The heat pumps work extracts the heat from the hot source, the offices, and then expels it in the exterior of the building. The cold network captures heat from buildings using delivery stations. Then, buried pipes transport it to a cooling plant, which uses local energy sources (ENGIE, 2020). The aerorefrigerant tower (cooling tower), is a device used to remove heat from industrial processes or air conditioning systems. It operates by circulating hot water over fill material while drawing in air through fans. As the air passes over the water, some of it evaporates, cooling the water, which is then recirculated to maintain the desired temperature.

Regarding the ventilation systems, most of the ventilation is double-flux. This system allows, through an exchanger, to recover the heat from the extracted air to transfer it to the blown air through a heat exchanger, not mixing the extracted air and the supplied air. The simple flux ventilation does not recover the heat from the waste air. Regarding natural ventilation in Paris, people often don't open windows, increasing the amount of CO_2 upon the recommended limits recommended to human health.

Regarding the construction year of each building, shown in Figure 42 we have that most of them were made in the XX century, mainly the first half. There are also 20% of them that were built between the year 2000 and 2010.

Figure 42 – Frequency of retrofit s by end-use



Source: Author

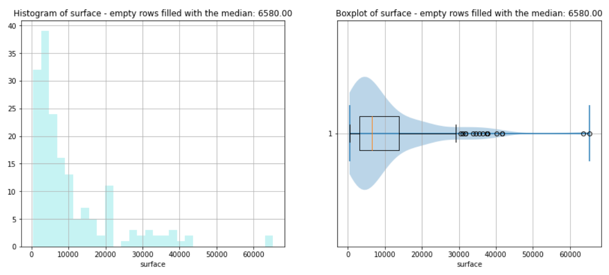
The year of construction is related to different materials and techniques, which impacts directly in the wall types. In this database, more than one third built in concrete, which is coherent with the buildings of the first and second half of the XX century. After that, we have curtain wall buildings, these buildings represent the significant part built in the second half of the XX century. Yet, there are around 30% of them built in stone (ancient buildings).

Regarding the insulation, more than 60% of the buildings have thickness between 5 and 10cm. This is coherent with the fact that Paris climate but also does not represent a huge useful area loss. Beyond these, one third are either non insulated or have an insulation smaller than 5cm. Only 10% have insulation greater than 10cm. Also, 70% of the buildings are insulated by the interior rather than the exterior.

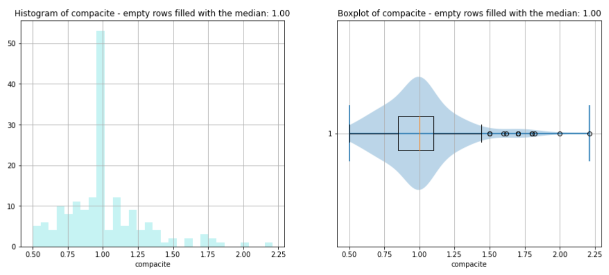
Regarding window characteristics, the majority if windows have shading protection, mainly in aluminum. And most important, the double glass is present in almost all windows, with 20% having some special treatment such as gas between the glass layers. There's still have the distributions of floor and ceiling insulation and lighting control that are not described for the sake of succinct. The buildings' surface in the database is very well distributed, with a median of 6580m^2. The largest surpassing 60000m^2, 50% of the buildings are between 4000 and 14000m^2, approaching a Pareto distribution.

Figure 43 – Surface and Compactness distribution

(a) Surface (m^2)



(b) Compactness (%)



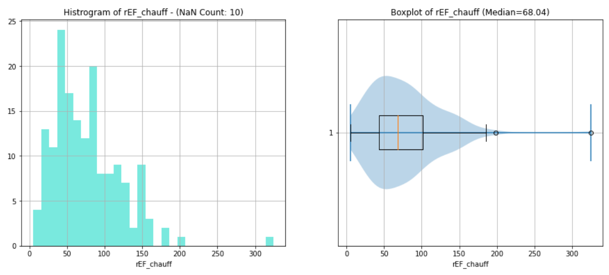
Source: Author

Regarding the windows transmittance and solar heat gain factor (SHGC), the empty rows are around 20%. Not considering these, the median for the SHGC is 0.58 and for the transmittance, 2.83 W/m^2K, which is extremely high since the window of a new construction must have a U_w less than 1.5 W/m^2K in the RE2020.

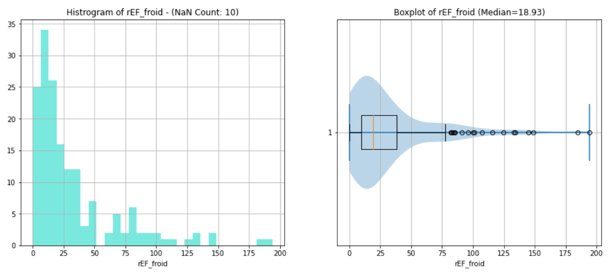
In this database, there is also the end-use for each building. These data had only ten empty rows each, a rapid manual analysis revealed that those belonged all to one single buildings being removed from the database. In Figure 44 below, we have the heating and cooling end-use consumptions:

Figure 44 – Final energy consumption by end-use (kWh_{EF}/m^2/year)

(a) Heating



(b) Cooling

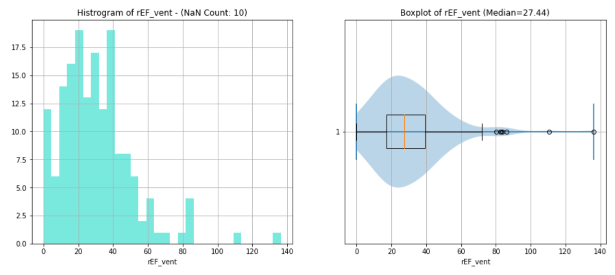


Source: Author

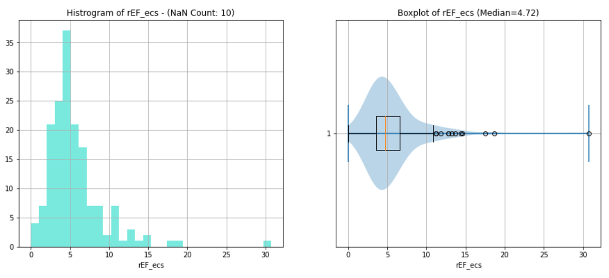
As expected, the most important consumption corresponds to the heating, with a normal distribution spreading until 200 kWh_{EF}/m^2/year, having 50% of them comprised between 50 and 100 kWh_{EF}/m^2/year. The cooling has a median of 19 kWh_{EF}/m^2/year, a narrower range, of 12 to 35 kWh_{EF}/m^2/year to 50% of the cases. In Figure 45 below, there is ventilation and hot water systems.

Figure 45 – Final energy consumption by end-use (kWh_{EF}/m^2/year)

(a) Ventilation



(b) Hot water

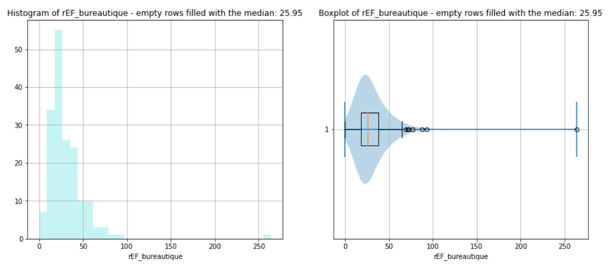


Source: Author

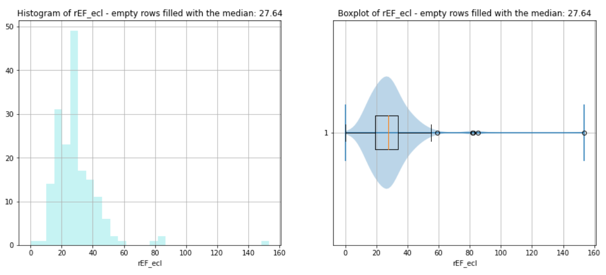
The median is 27 kWh_{EF}/m^2/year, but having a wide range, that gets 80 kWh_{EF}/m^2/year. The ventilation also may be used to help the heating of the building when it is double-flux, allowing the recovery of heat from the air that was expelled to the O_2 renovation. Hot water has a small median of 4.7 kWh_{EF}/m^2/year, because in offices it is generally used only to toilet and work kitchens. However, as we saw in the previous analysis in the Section 4.1.1, these have a relatively huge potential for improvement. Regarding the systems perceived and directly controlled by the users, such as lighting and office equipments, we have the Figure 45 below.

Figure 46 – Final energy consumption by end-use (kWh_{EF}/m^2)

(a) Office equipments



(b) Lighting



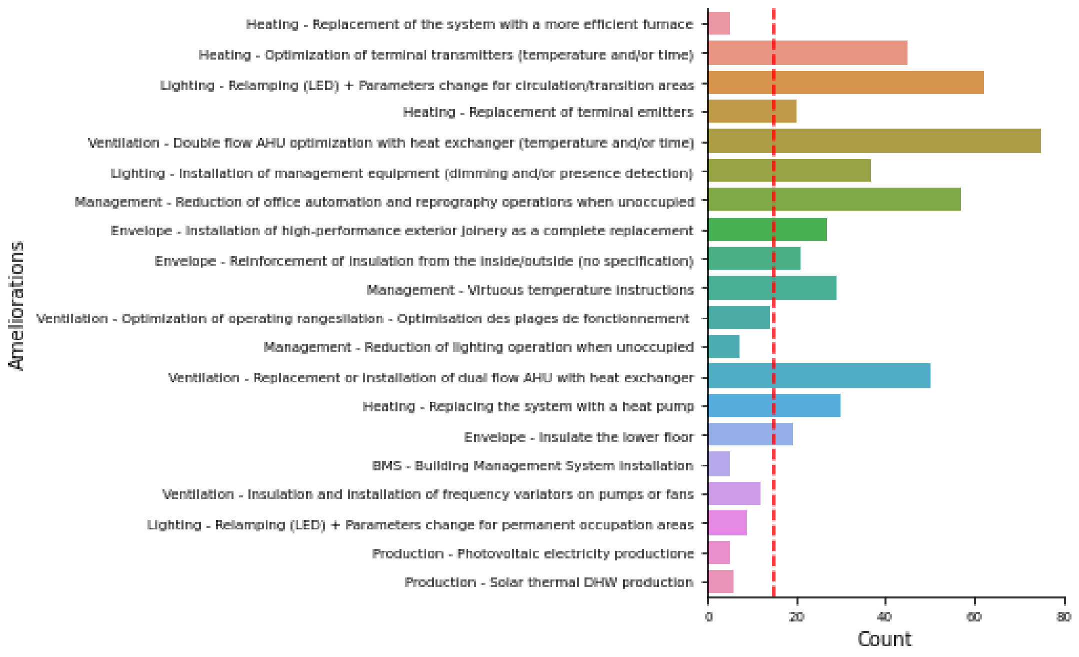
Source: Author

The office equipments, which do not include server consumption, may represent a huge energy consumption. The median is 26 kWh_{EF}/m^2, but it actually does not present a large range, achieving the maximum of 50 kWh_{EF}/m^2 in very small cases. Half of the cases are constrained in a range of 15 to 40 kWh_{EF}/m^2. The lighting consumption is very close to the office equipments consumption, with a median close to 28 kWh_{EF}/m^2, being important targets to retrofit.

4.1.3 Preprocessing: Joining, standardization, filtering and correlation

After this first analysis of both databases, we proceed to their assembling, which resulted in 65 compatible buildings, containing together 709 retrofit suggestions. Each of them represent a calculated retrofit suggestion, but in order to make the prediction model we need to assemble these in classes and standardize their titles. Therefore, from the 709 retrofit suggestions, were included 504 in the classes shown in Figure 47.

Figure 47 – Retrofit suggestions



Source: Author

As we can see, the classes of retrofit suggestions are very diversified. In the heating, we have the optimization and substitution of emitters such as radiators, or others. The changing of the heating production, furnaces, and the installation of heat pumps. To lighting solutions, we have the relamping, optimization of functioning hours through equipments and management.

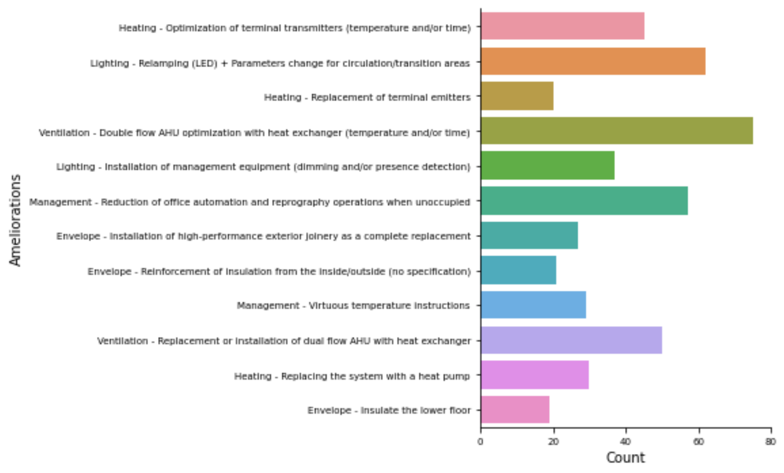
For the ventilation, we have two retrofits to change equipments, first the installation of double flux ventilation and the other the installation of the secondary equipment to the same strategy, also there is the optimization of the temperature or the hours of utilization. Also, there is the management of office equipments in the utilization front.

To the envelope, there is the installation of insulation in the walls and floor, and installation of highly efficient windows. There's also energy and hot water production. Finally, there are solar panels, both photovoltaic and thermal.

These retrofit suggestions are very diverse and interesting to investigate. However, in terms of prediction, many of them do not have a significant volume to create statistics, such is the case of solar photovoltaic and thermal panels.

Therefore, we select only those with a frequency of occurrence greater than 15, which are 448 out of 504, which are described in the Figure 48 below.

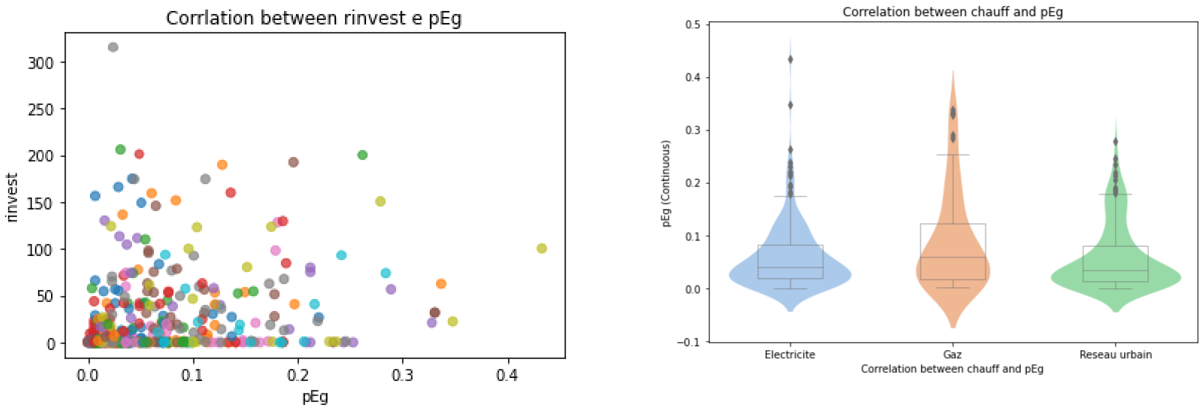
Figure 48 – Retrofit suggestions with more than 15 occurrences



Source: Author

From previous 21 retrofit suggestions, 12 were selected. The most frequent is the installation of double-flux ventilation, followed by relamping LED, good management of office equipments, optimization of emitters (temperature and activation time), installation of control equipments over lighting and finally thermal insulation of walls, floor, and window substitution. After calculating the percentage final energy saving of the retrofit suggestion in relation to the total energy consumption, shown in the Figure 49 below.

Figure 49 – Retrofit suggestion correlation with Final Energy savings (quantitative)



Source: Author

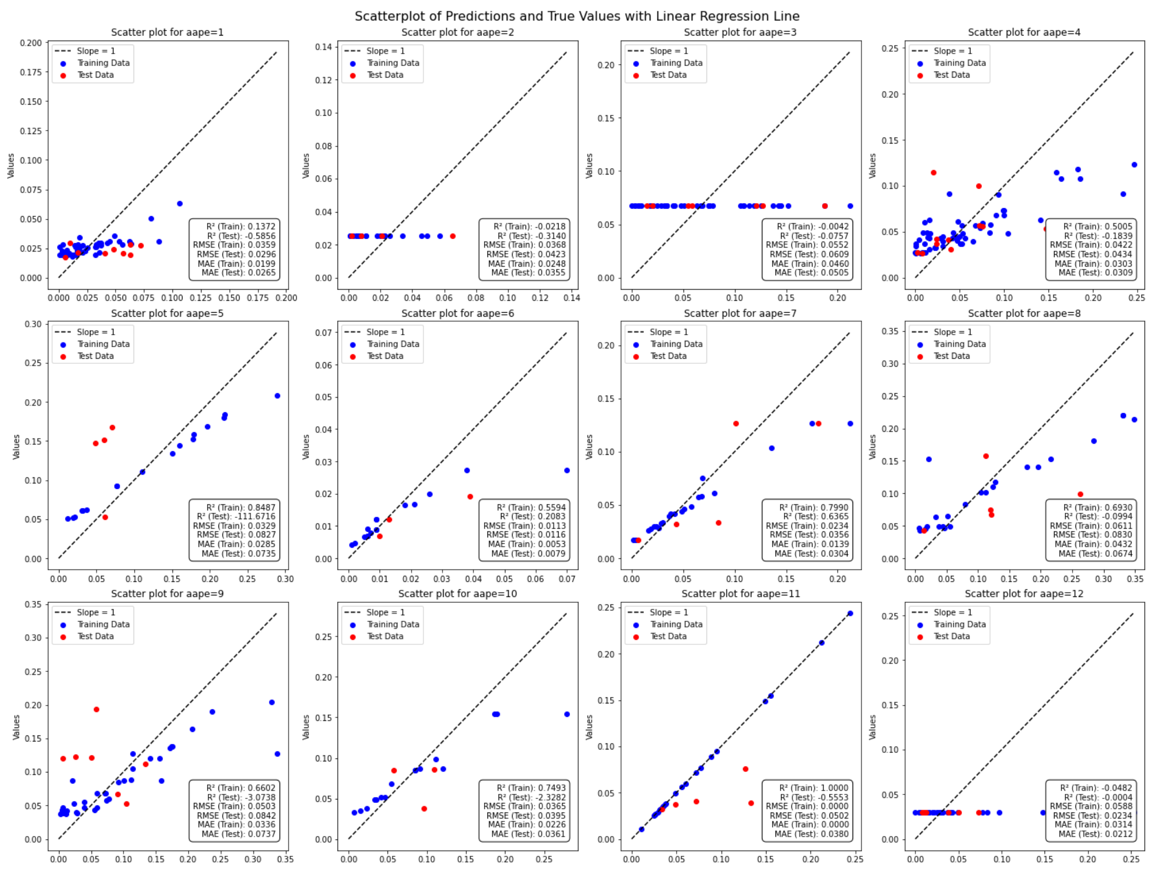
Each dot and violin represent a retrofit suggestion (*AAPE*) versus the energy savings. Clearly no correlation can be taken from these graphs, and it repeats to all other variables, showing how complex the relationship between the data is and how a prediction with all those features would be infeasible by hand. After this, feature engineering is applied to normalize data to the model. These results are not shown here as they are only a reflect of the physical values already presented.

4.2 ROUND 1 - MACHINE LEARNING RESULTS

4.2.1 R1 - STEPS 2,3,4 and 5: Training, Hyperparameters optimization, validation, Random state variation and cross-validation

For model analysis, the scatterplots below compare predicted and actual values, with blue representing training data and red representing test data. The results of the GBM, ANN, Decision Tree, and Random Forest methods will be presented. Additionally, for the best-performing method, further processing is applied, including variable reduction, outlier removal, and clustering techniques. The first one is made with GBM model using parameters described in the Methodology section.

Figure 50 – GBM with Cross-Validation



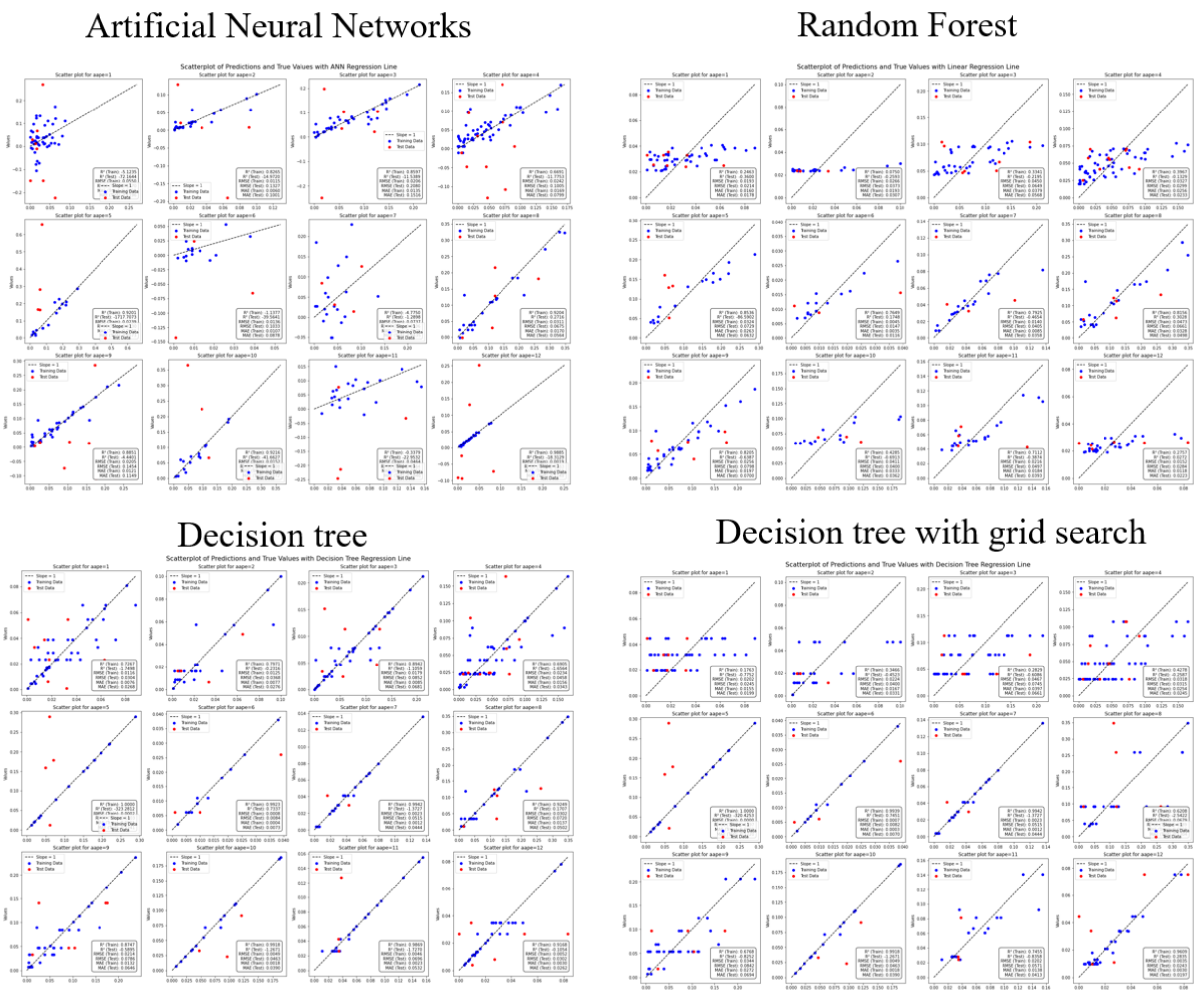
Source: Author

The analysis of the scatter plots reveals that the predictions made by the model systematically mispredict higher observed values, as indicated by slopes consistently less than 1 across various interventions. This inclination (slope $m < 1$) suggests that the model's predictions do not fully capture the variability in the observed data, particularly underestimating in the upper range.

In order to verify if these predictions could be improved by using other machine learning algorithms, such as Artificial Neural Networks, Random Forest and Decision Tree. As Decision Tree performed well, a cross-validation was also applied.

The results of these algorithms are presented below:

Figure 51 – Alternative algorithms with cross-validation / DT: hyperparameters tuning



Source: Author

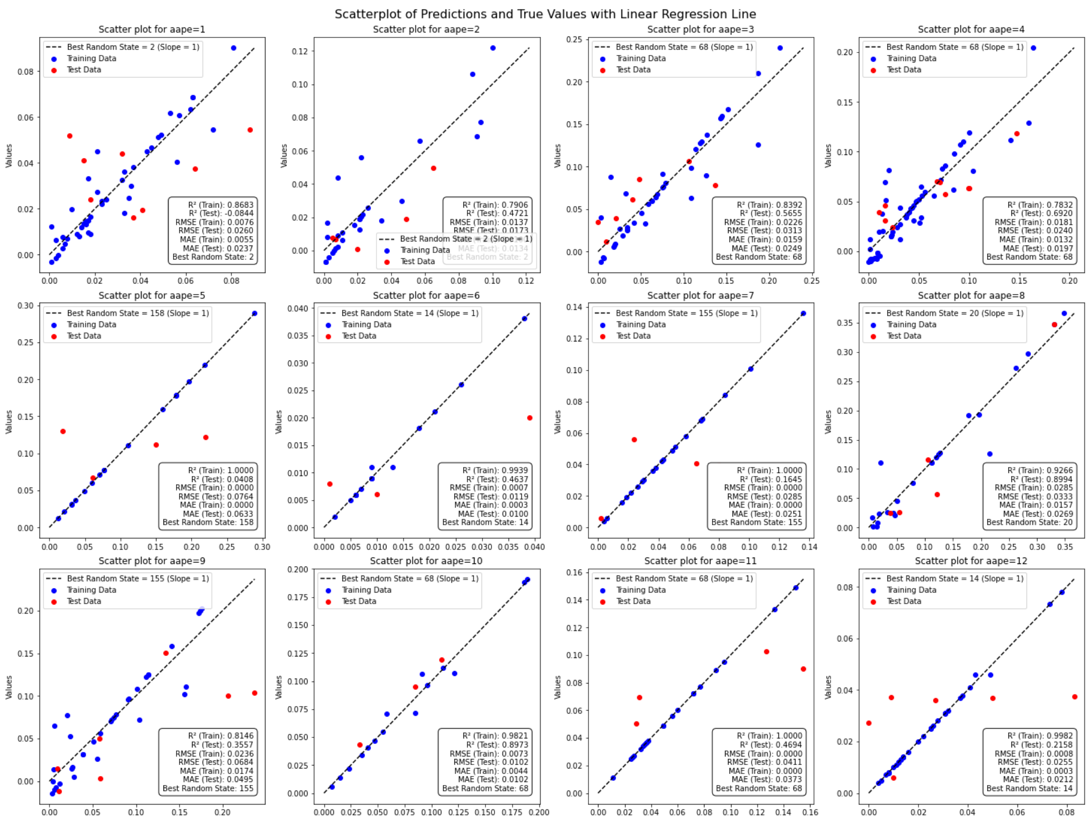
The artificial neural network proved to be the least effective model in this analysis, while the random forest initially exhibited performance comparable to that of the Gradient Boosting Machine (GBM). However, its higher dispersion ultimately rendered it less viable for creating a combined model.

In contrast, the decision tree emerged as a promising alternative (notably, only the 9th model demonstrated any significant improvement). Nevertheless, when cross-validation was applied, the resulting stratifications were highly pronounced, also as further treatments are applied to the GBM method, it consistently yielded better results, ultimately highlighting the unsuitability of the other algorithms for this particular analysis.

These characteristics and the smaller dispersion among points led the study to prioritize the enhancement of the GBM model over the other algorithms tested, seen the possibility of correct it from a linear regression as it will be seen in the next analysis. This focus on GBM underscores its potential as the most robust modeling approach in this context.

To address this bias, a linear adjustment can be applied to the model's predictions by fitting a linear regression line to the training data between the predictions and the true values through a combined function, shown below:

Figure 52 – GBM + Regression (Combined Model) - All parameters



Source: Author

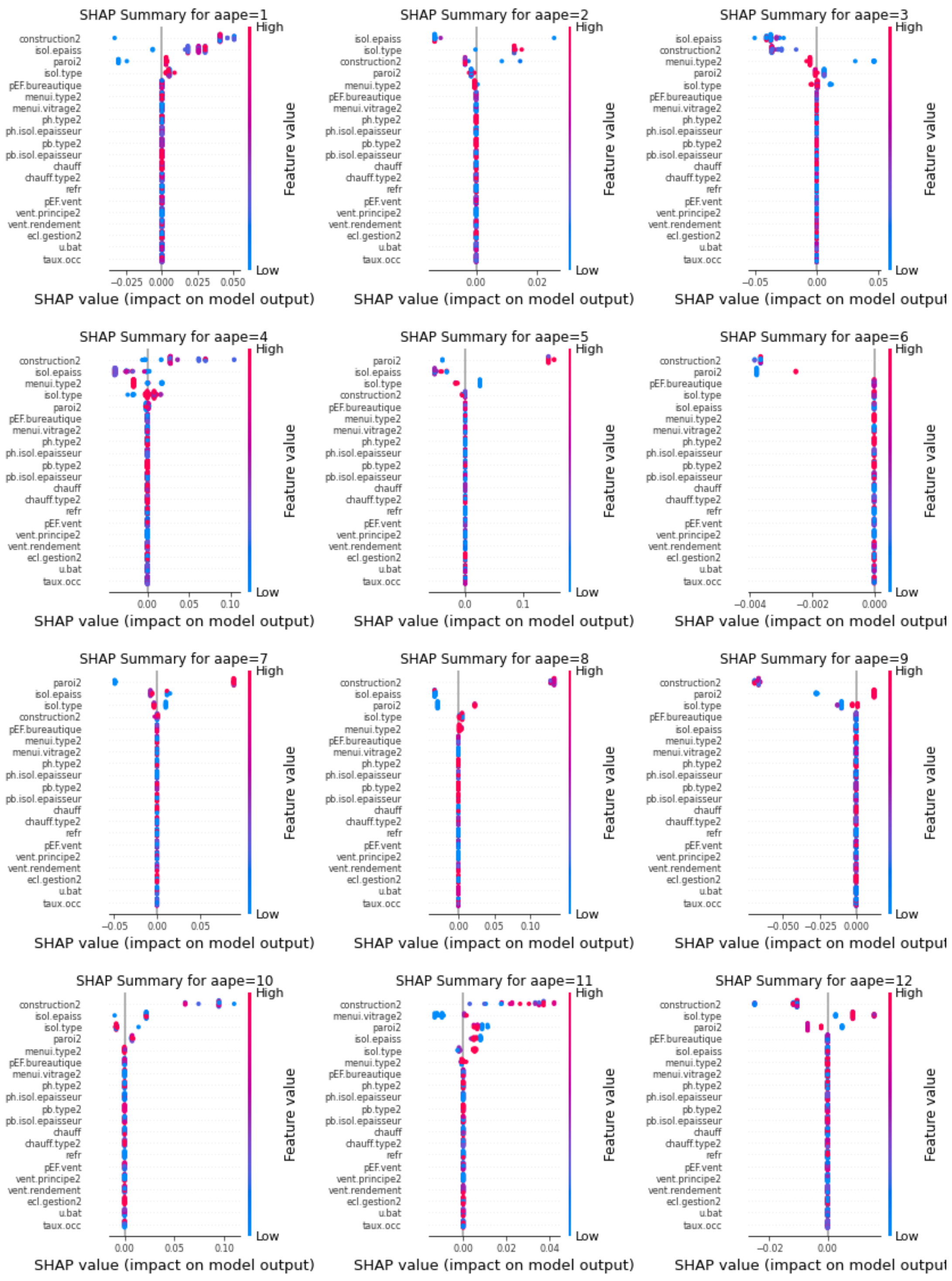
Interventions in Lighting generally perform well, with models for simple actions like relamping (Model 1) showing high generalization, while more complex tasks, like installing dimming or presence-detection systems (Model 2), exhibit signs of overfitting, suggesting sensitivity to training-specific patterns. Similarly, Ventilation interventions reveal mixed results: while the optimization model (Model 4) generalizes well, the model for complex installations (Model 3) struggles to capture broader variance in test set.

Insulation models (Models 5–7) consistently achieve high R^2 and low error values across datasets, indicating excellent generalizability. This pattern suggests that insulation improvements are well-captured by the model, likely due to their straightforward impact on energy efficiency. Heating interventions also show strong performance, particularly for direct upgrades like replacing systems with heat pumps (Model 8), while optimization models (e.g., Model 9) tend to overfit, likely due to complexity and feature sensitivity. Finally, Management interventions (Models 11 and 12) yield excellent results, reflecting predictable impacts from simple behavioral changes, such as setting virtuous temperature limits and reducing equipment usage during unoccupied periods, but perform not very well in the test set.

4.2.2 R1 - STEP 6: SHAP sensitivity analysis

This SHAP analysis provides insight into feature importance across twelve models, each representing one building retrofit action. As shown below, most parameters aren't influential, showing a possibility of simplification.

Figure 53 – SHAP Analysis Combined Model - All parameters



Source: Author

Lighting interventions, such as LED relamping and the installation of control equipment (dimming and presence detection), show varying impacts, particularly where energy efficiency are increased through responsive lighting management. Ventilation features, including the installation of double-flow air handling units (AHUs) with heat exchangers and optimized temperature schedules, display moderate to high influence, indicating substantial efficiency improvement through better airflow and heat recovery.

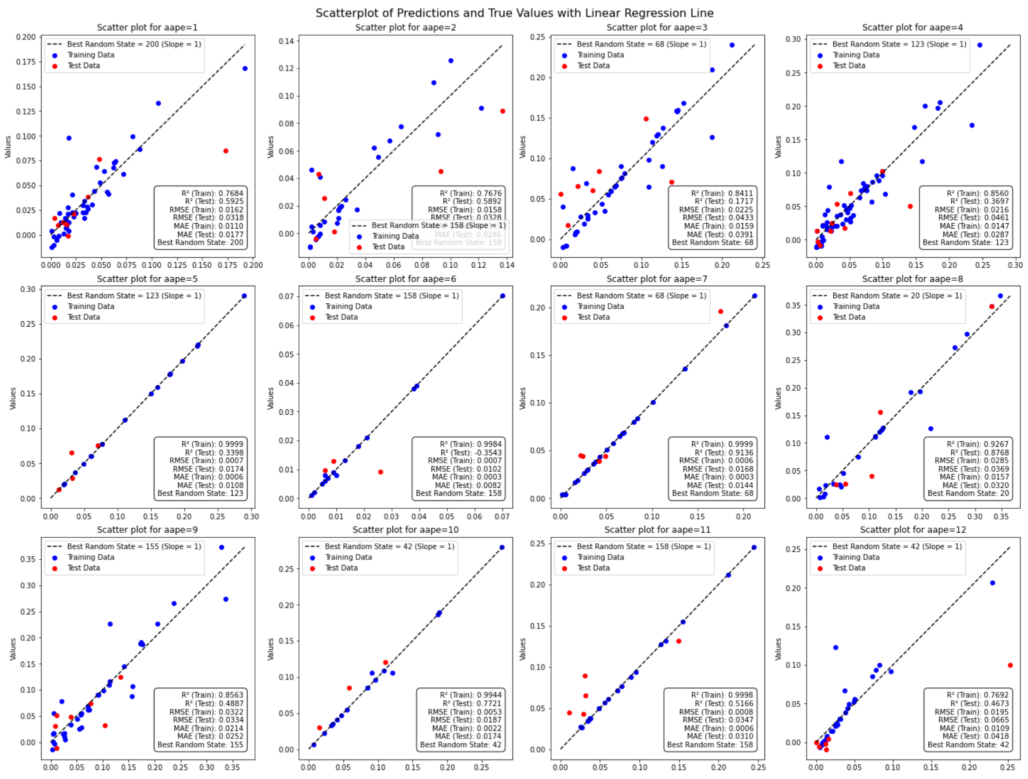
Building envelope improvements, such as enhanced insulation and high-performance windows, consistently demonstrate high SHAP values, reducing thermal losses and minimizing climate control demands. Heating interventions, like heat pumps and optimized emitters, also have strong impacts across models. Management measures, such as temperature setpoint strategies and reduced equipment usage during off-hours, add control over energy patterns, though with moderate influence.

4.3 ROUND 2 - MACHINE LEARNING RESULTS

A filtering process is applied to retain only the most impactful variables, making the model more user-friendly while maintaining effective retrofit recommendations.

4.3.1 R2 - STEPS 2,3,4 and 5: Training, Hyperparameters optimization, validation, Random state variation and cross-validation

Figure 54 – GBM + Regression (Combined Model) - Filtered parameters



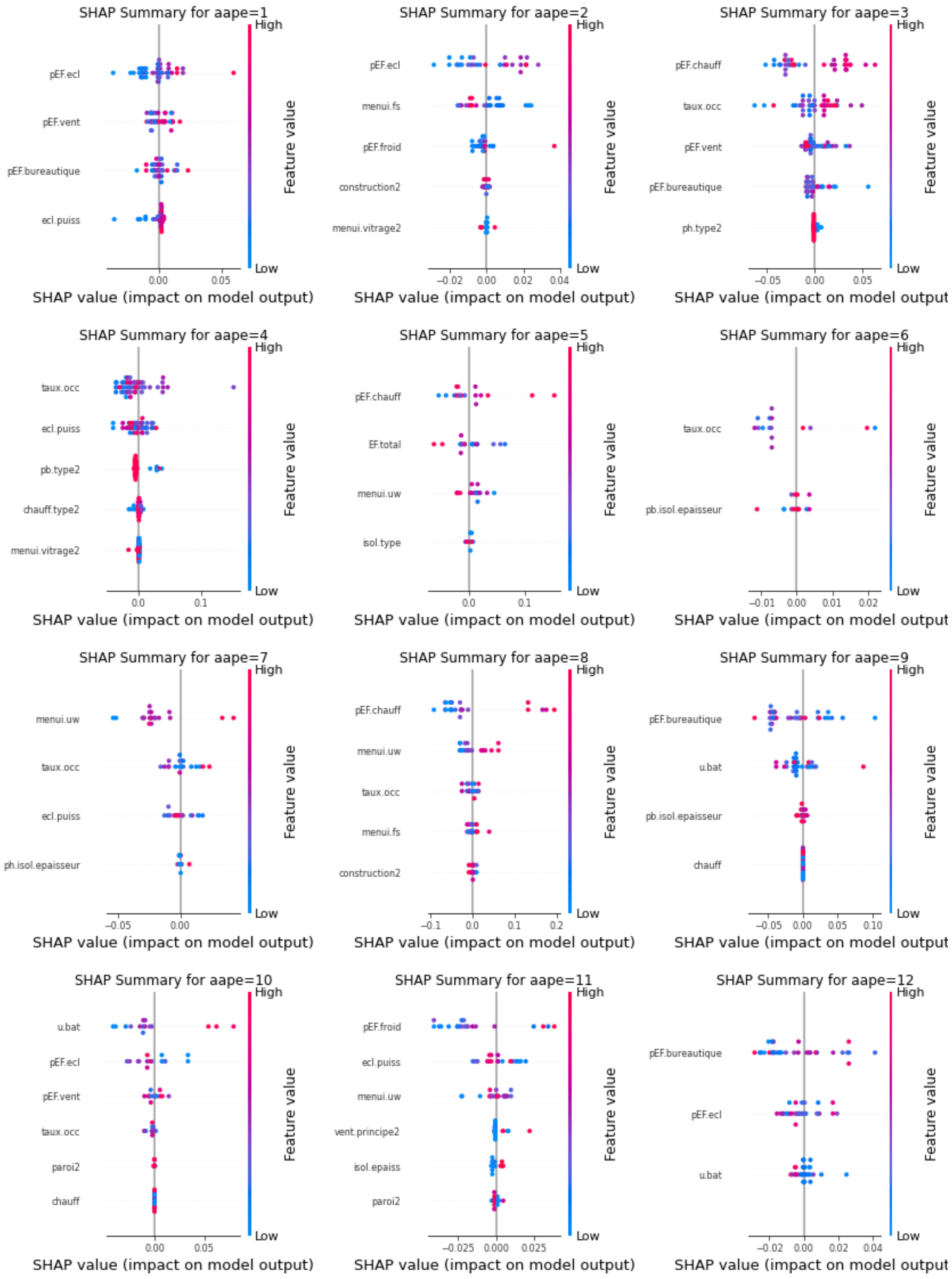
Source: Author

As the models present similar results, it’s possible to deduce the reduction of variables has generated a rather positive simplification of the model, allowing it to be focused to iterate in the variables that have been showing to be really important.

4.3.2 R2 - STEP 6: SHAP sensitivity analysis

To these models, the SHAP analysis is shown below:

Figure 55 – Round 2 - SHAP Analysis



Source: Author

In the lighting category, lighting end-use consumption ("pEF.ecl") and lighting power ("ecl.puiss") are consistently influential across models, suggesting that improvements in these areas can significantly reduce energy consumption. High SHAP values for these features indicate that both the choice of energy-efficient lighting systems and the control of lighting power usage contribute substantially to energy efficiency. This is particularly true in models where responsive lighting management systems, such as dimming and presence detection, are utilized to minimize waste during periods of low occupancy.

Ventilation variables, especially ventilation end-use consumption ("pEF.vent") and operational principles ("vent.principe2"), display moderate to high SHAP values, highlighting their importance in energy scenarios that prioritize airflow optimization. Efficient ventilation systems, especially those using energy recovery and adaptive control settings, positively affect energy consumption predictions by reducing the need for excess heating and cooling. Models that emphasize ventilation improvements underscore the potential for energy reductions through strategic management of airflow and temperature settings.

Building envelope variables, including window insulation ("menuiserie.uw"), insulation thickness ("pb.isol.epaisseur"), and wall insulation type ("paroi2"), show high SHAP values across several models, indicating their critical role in minimizing thermal losses. Enhancing insulation, both in terms of quality and thickness, is essential for reducing the building's heating and cooling requirements. Models that prioritize envelope improvements illustrate how effective insulation measures, particularly for windows and walls, can significantly enhance overall energy efficiency.

Heating variables, such as heating end-use consumption ("pEF.chauff") and heating system type ("chauff.type2"), are also consistently impactful across the models. High SHAP values for these features reveal the importance of efficient heating technologies in reducing energy demands, particularly in colder climates or buildings with high heating needs. Optimizing heating systems, such as using heat pumps and temperature controls, can align energy output more closely with actual occupancy needs, thereby reducing energy waste.

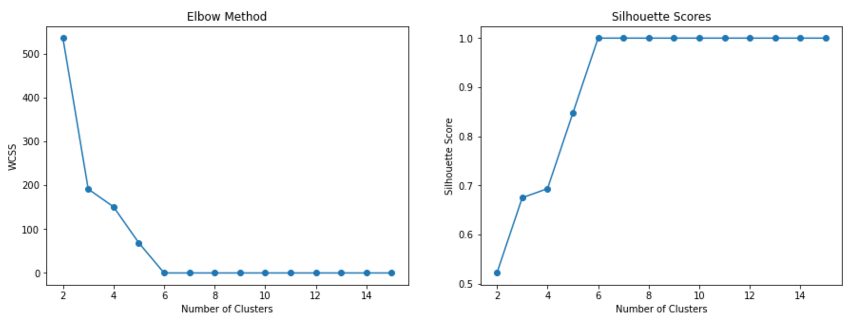
Management features, including the office equipment end-use consumption ("pEF.bureautique") and occupancy rates ("taux.occ"), show moderate influence, reflecting the potential of operational strategies to fine-tune energy usage. These features suggest that energy efficiency can be improved by adjusting equipment use patterns and aligning consumption with occupancy. To further streamline the models and make them more user-friendly, a filtering process will be applied to retain only the most impactful variables. This refinement will simplify the models while maintaining their effectiveness, making them a practical tool for optimizing building energy management by focusing on the highest-value interventions.

4.4 ROUND 3 - MACHINE LEARNING RESULTS

4.4.1 R3 - STEP 7: Clusterisation

Therefore, we proceed to data treatment, the first one applied is the clusterisation from two different methods: elbow and silhouette score. This clusterisation is made to group five classes of variables, that are: envelope, lighting, heating, cooling and ventilation. In the figure below, the ventilation cluster is shown, composed by ventilation principle and performance.

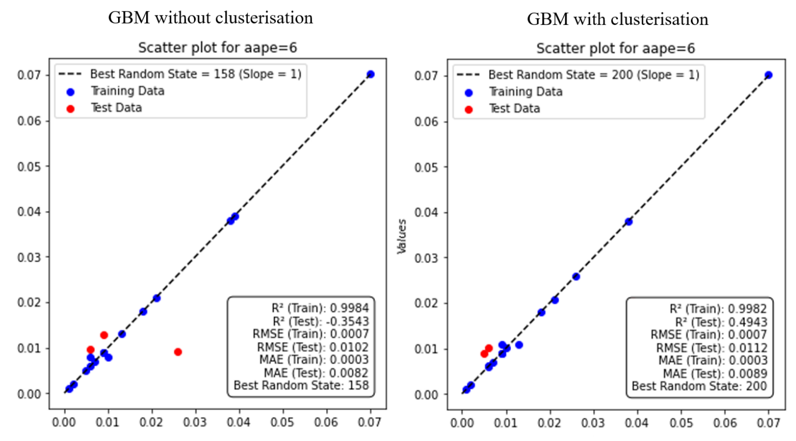
Figure 56 – Elbow and Silhouette Methods to ventilation



Source: Author

In the elbow method, as the number of clusters increases, this distance decreases; however, after a certain point (the "elbow"), the rate of decrease slows down. The optimal cluster number is typically chosen at this "elbow" point where adding more clusters yields diminishing improvements. The silhouette score, on the other hand, measures how similar points in one cluster are to points in the next closest cluster. The optimal number of clusters is where the silhouette score is highest, indicating well-defined and distinct clusters. Therefore, to this example we choose 6 as the appropriate number of clusters. The other clusters are shown in annex A. The effect these clusters had is shown in Figure below:

Figure 57 – Clusterisation improvement on models



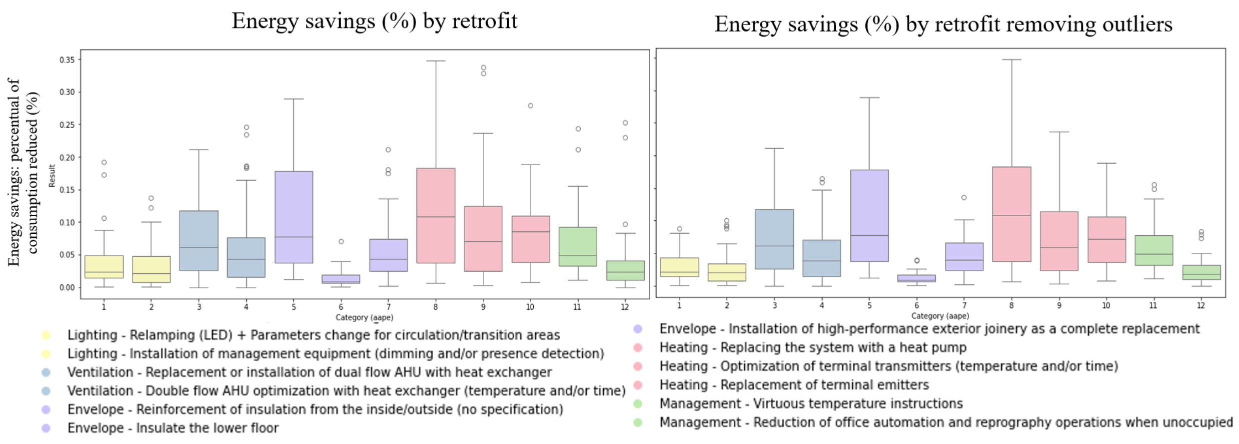
Source: Author

This clusterisation has little interfered in the performance of the models, however to the retrofit 6 ('Envelope - Insulation of Lower floor'), it has displaced a distant case in test set to a nearer place, showing the greater capacity of the model's generalization after the inclusion of the clusters.

4.4.2 R3 - STEP 8: Removal of outliers

Another treatment made is the removal of outliers, which improves clustering accuracy by eliminating extreme values that could distort group formation. This ensures that clusters represents the patterns in data without interference from irregular values. This removal of outliers has changed the distribution of the data as shown in the graphs below:

Figure 58 – Boxplot without (left) and with (right) the removal of outliers



Source: Author

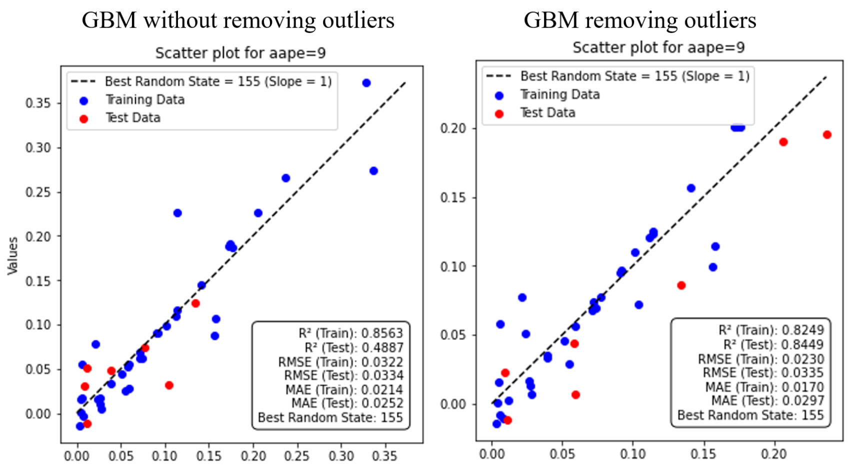
In the left plot, categories 5 (insulation reinforcement) and 8 (heating optimization) show high median savings but also substantial variability, indicating that these retrofits often yield larger efficiency improvements but with inconsistent results.

Many outliers in these categories suggest that some implementations lead to exceptionally high or low savings. In contrast, lighting improvements (categories 1 and 2) yield lower, more consistent savings due to their standardized application.

In the right plot, removing outliers narrows the distributions, allowing a clearer view of typical performance within each category. The insulation and heating categories still show high variability but are more comparable to other interventions, emphasizing their effectiveness without the influence of extreme cases. Overall, insulation and heating upgrades offer high, variable savings, while lighting improvements provide smaller, more predictable benefits.

This also does not interfere significantly in models, with the exception of two retrofit suggestions, which are show below:

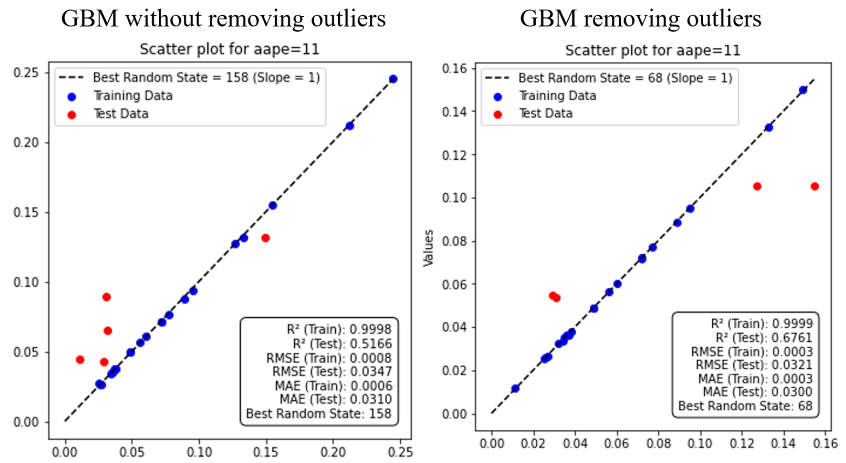
Figure 59 – Retrofit 9: Heating - Thermal emitters optimization



Source: Author

As shown, the improvement of R² to the test set is huge, going from 49% to 84%, however the other indicators such as RMSE and MAE don't change, showing the actual improvement are very slightly significative. The other retrofit that improves is Temperature Management shown below, having also a slightly improvement:

Figure 60 – Temperature Management



Source: Author

To conclude, as each model is independent of the others, we choose to apply the best one found among all others to each retrofit. Thus said, to the most part of the retrofit actions, the chosen model is the GBM with cross validation, coupled with linear regression in the Combined Model. To the insulation of lower floor, the clusterisation is applied with the improvement of performance due to the envelope cluster. To the replacement of terminal emitters and temperature management, the removal of outliers is applied in order to increase slightly the performance of the test set. Overall, the models are acceptably precise to be used in the company's cases.

5 CONCLUSION

This project is a step further in the simplification process of expert audits in the company where it was made as it allows fast decision making and reliable results considering the desired precision by the company. Therefore, achieving the main objective of improving the energy audit process by saving time to the company.

The databases analysis provided the base to the machine learning models. Using these data, it is possible to see that the average energy consumption across surveyed buildings is close to 120 kWh/m^2 per year, with heating accounting for 60% and cooling for 30% of total energy use. The output figures shows a wide margin of improvement in efficiency, particularly in older buildings. Buildings constructed after 2000 shows an average energy consumption close to 85 kWh/m^2, in comparison with those built before 1980 which consumes close to 150 kWh/m^2.

The outcome of theses analysis tend to demonstrate that retrofitting older structures with modern insulation and energy-efficient systems may reduce energy usage close to 40%. High-energy consumption areas are identified, with heating systems consuming an average of 72 kWh/m^2 during winter, meanwhile cooling systems peak at 40 kWh/m^2 during summer. Targeted energy audits are useful to spot both inefficiencies and recommending improvement such as high-efficiency HVAC systems.

After many trial and errors and multiple models tested, the Gradient Boosting Machine (GBM) consistently emerged as the most reliable approach, particularly for complex retrofit actions. This model demonstrated a good balance between bias and variance, making it adequate for capturing the slight effects of energy consumption improvement across different building systems.

Specifically, the GBM model's performance was enhanced by tuning some parameters, ten train the model and test it for validation using test data. Some other models like the Artificial Neural Networks (ANN), Decision Trees, and Random Forest offered an interesting comparison but presented some limitations in specific contexts which we will describe. The ANN model, while commonly effective in many predictive contexts, had hard time to maintain stability and consistency in this application, specifically in scenarios with high variance in energy use dataset.

The Decision Tree model seemed adequate but exhibited overfitting tendencies, especially under cross-validation, which compromised its ability to generalize across different retrofit scenarios. Similarly, the Random Forest model initially delivered satisfactory results comparable to GBM, but its higher dispersion in predictions makes it less reliable for use in a combined modeling approach, which drives us to use GBM as the best choice for this analysis.

To correct observed biases, mainly the systematic misprediction of higher observed values (evidenced by slopes consistently less than 1 in scatterplot analyses), a linear adjustment was applied. This adjustment is supposed to correct the slope error by transforming the model's predictions through linear regression. This tuned approach, incorporating both GBM and linear adjustments calculated from initial model error, allows a better final correlation between predicted and observed values.

The model evaluation also revealed some correlation across intervention categories. For example, lighting interventions were well-predicted, with simpler actions like changing the light system demonstrating high generalizability, while more complex controls (e.g., dimming and presence detection systems) showed slight overfitting. Ventilation models revealed mixed outcomes; while simpler optimizations generalized effectively, models predicting complex system installations faced challenges in capturing broader variance. Insulation models consistently delivered high R^2 and low error values, indicating a strong fit and robust predictions across datasets.

Many other data treatments were made to refine model performance, including clustering and outliers removal. Clustering was made using the elbow and silhouette methods, we ends up finding that six clusters was optimal. Cluster analysis grouped retrofit measures by category—such as envelope, lighting, heating, cooling, and ventilation—aiming to capture patterns among similar kind of interventions. Although clustering slightly enhanced the model's ability to generalize (for example by re-positioning outlier cases closer to the distribution's core), its impact on overall predictive accuracy was limited, with improvements only in the "Envelope- Insulation of Lower Floor" intervention.

By removing the outlier data, we see a minimal effect across most interventions, but it improved the prediction accuracy for heating and management models by reducing the influence of extreme values that disturb the regression. The SHAP analysis further provided a better value interpretation, indicating that several key variables could drive simplification in the model without compromising effectiveness. Notably, energy efficiency in lighting and power usage, ventilation efficiency, and building insulation thickness generally displayed high SHAP values. This pattern shows that focusing on these impactful variables allows to create a generic and reusable model. Its also gives the opportunity to create a more user-friendly tool for building energy management.

In conclusion, the GBM model with cross-validation, coupled with a linear regression adjustment in the Combined Model, demonstrated the highest accuracy and stability for most retrofit actions, outperforming other algorithms in this analysis. For complex interventions like envelope insulation and heating emitter replacement, additional treatments—such as clustering and outlier removal—further improved the model's performance by aligning predictions with actual energy consumption patterns.

Therefore, this project represents a great advancement in the company's path into machine learning. The method has already been applied in a previous company's study, but was vastly improved in order to have greater usability for employees, moving on to the development of independent models contrary to what was previously available in order to reduce the number of inputs for prediction of each of them, without compromising their respective performances. This was also made possible through SHAP analysis and the application of advanced machine learning techniques.

For future work, it is suggested:

- To develop a method that can be used to different climates to different cities in France: this will provide predictions to other cities in France, not only Paris making the models more robust to the company's application
- To compare new buildings in the expert opinion and thermal simulation with the machine learning model : this will allow to understand if the model is useful to the employees, if the unseen buildings have a behavior that is considered good enough to the company beyond the ones that were used in the test phase.
- To exploit the whole Server of the Company with cases that were not inserted in the database: Use data mining techniques to be able to explore the whole server and generate a larger database. It is know that only around 8% of the company's buildings were in the formal database, which gives a huge potential of exploitation of all the other buildings that are available but still not integrated in the model.
- To predict the energy efficiency acquired to each end-use consumption: as the energy audit requires a detailing of the end-use consumption and the discrimination of which end-use is impacted with the retrofit action, the improvement of the machine learning to give discriminated end-use values is crucial to simplify even more the audit process.
- To compare the SHAP analysis with other sensitivity tools to evaluate the impact of each feature on the model: compare the SHAP analysis with Sobol, or other techniques to verify if they present the same relationship between the results and the variables to validate the feature selection that was made.

REFERENCES

ABOELATA, Amir. Reducing outdoor air temperature, improving thermal comfort, and saving buildings' cooling energy demand in arid cities – Cool paving utilization. **Sustainable Cities and Society**, v. 68, p. 102762, 2021. ISSN 2210-6707. DOI: https://doi.org/10.1016/j.scs.2021.102762. Available in: https://www.sciencedirect.com/science/article/pii/S2210670721000561. Accessed on: 15 Nov. 2024.

ADEME. **Audit Énergétique dans les bâtiments**. 2020. Available in: https://librairie.ademe.fr/urbanisme-et-batiment/730-audit-energetique-dans-les-batiments.html. Accessed on: 15 Nov. 2024.

AHMED, Mohiuddin; SERAJ, Raihan; ISLAM, Syed Mohammed Shamsul. The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. **Electronics**, v. 9, n. 8, 2020. Available in: https://www.mdpi.com/2079-9292/9/8/1295.

ALANNE, Kari; SIERLA, Seppo. An overview of machine learning applications for smart buildings. **Sustainable Cities and Society**, v. 76, p. 103445, 2022. ISSN 2210-6707. DOI: https://doi.org/10.1016/j.scs.2021.103445. Available in: https://www.sciencedirect.com/science/article/pii/S2210670721007186. Accessed on: 16 Nov. 2024.

ALIFERIS, Constantin; SIMON, Gyorgy. Overfitting, Underfitting and General Model Overconfidence and Under-Performance Pitfalls and Best Practices in Machine Learning and AI. In: **Artificial Intelligence and Machine Learning in Health Care and Medical Sciences: Best Practices and Pitfalls**. Ed. by Gyorgy J. Simon and Constantin Aliferis. Cham: Springer International Publishing, 2024. P. 477–524. Available in: https://link.springer.com/chapter/10.1007/978-3-031-39355-6_10. Accessed on: 15 Nov. 2024.

ALIRAMEZANI, Masoud; KOCH, Charles Robert; SHAHBAKHTI, Mahdi. Modeling, diagnostics, optimization, and control of internal combustion engines via modern machine learning techniques: A review and future directions. **Progress in Energy and Combustion Science**, v. 88, p. 100967, 2022. ISSN 0360-1285. DOI: https://doi.org/10.1016/j.pecs.2021.100967. Available in: https://www.sciencedirect.com/science/article/pii/S0360128521000654. Accessed on: 15 Nov. 2024.

ARROW, K. J. et al. **Contributions to the Theory of Games (AM-28), Volume II**. [*S.l.*]: Princeton University Press, 1953. ISBN 9780691079356. Available in: http://www.jstor.org/stable/j.ctt1b9x1zv. Accessed on: 16 Nov. 2024.

AWAN, Abid Ali. **An Introduction to SHAP Values and Machine Learning Interpretability**. 2023. Available in:

https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability. Accessed on: 16 Nov. 2024.

AZOUZ, Mona; ELARIANE, Sarah. Towards energy efficiency: retrofitting existing office buildings using smart technologies. **Journal of Engineering and Applied Science**, v. 70, n. 1, p. 147, 2023. ISSN 2536-9512. DOI: 10.1186/s44147-023-00327-0. Accessed on: 15 Nov. 2024.

BALCZAREKI, Yuri Potrich. **Precificação de imóveis em Florianópolis utilizando técnicas de aprendizado de máquina**. 2024. Universidade Federal de Santa Catarina, Florianópolis, Brazil. Available in: https://repositorio.ufsc.br/xmlui/handle/123456789/255809. Accessed on: 15 Nov. 2024.

BERRAR, Daniel. Cross-Validation. In: RANGANATHAN, Shoba et al. (Eds.). **Encyclopedia of Bioinformatics and Computational Biology**. Oxford: Academic Press, 2019. P. 542–545. ISBN 978-0-12-811432-2. DOI: https://doi.org/10.1016/B978-0-12-809633-8.20349-X. Available in: https://www.sciencedirect.com/science/article/pii/B978012809633820349X. Accessed on: 15 Nov. 2024.

BOCANEALA, Nicoleta et al. Artificial Intelligence Based Methods for Retrofit Projects: A Review of Applications and Impacts. **Archives of Computational Methods in Engineering**, 2024. ISSN 1886-1784. DOI: 10.1007/s11831-024-10159-7. Available in: https://doi.org/10.1007/s11831-024-10159-7. Accessed on: 16 Nov. 2024.

BONTE, Mathieu; THELLIER, Francoise; LARTIGUE, Berangere. Impact of occupant's actions on energy building performance and thermal sensation. **Energy and Buildings**, v. 76, p. 219–227, June 2014. DOI: 10.1016/j.enbuild.2014.02.068. Accessed on: 15 Nov. 2024.

BROWN, Sara. **Machine learning, explained**. 2021. Available in: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained. Accessed on: 15 Nov. 2024.

BUCKLEY BARLOW, Cassidy Hilton. **The 4 Machine Learning Models Imperative for Business Transformation**. 2019. Available in: https://www.rocketsource.com/blog/machine-learning-models/. Accessed on: 15 Nov. 2024.

DASARADH. **A Gentle Introduction To Math Behind Neural Networks**. [*S.l.*], 2020. Available in: https://www.youtube.com/watch?v=CqOfi41LfDw&t=11s&ab_channel=StatQuestwithJoshStarmer. Accessed on: 15 Nov. 2024.

DAVE, Vachik S.; DUTTA, Kamlesh. Neural network based models for software effort estimation: a review. **Artificial Intelligence Review**, v. 42, n. 2, p. 295–307, 2012. DOI: `10.1007/s10462-012-9339-x`. Accessed on: 15 Nov. 2024.

DIXON, Tim. Commercial property retrofitting: What does "retrofit" mean, and how can we scale up action in the UK sector? **Journal of Property Investment & Finance**, Emerald Group Publishing Limited, v. 32, n. 4, p. 443–452, 2014. DOI: `https://doi.org/10.1108/JPIF-02-2014-0016`. Accessed on: 15 Nov. 2024.

EEA. **Buildings and Construction**. 2024. Available in: `https://www.eea.europa.eu/en/topics/in-depth/buildings-and-construction`. Accessed on: 15 Nov. 2024.

EIB. **Retrofitting for Energy Efficiency**. 2024. Available in: `https://www.eib.org/en/stories/retrofitting-energy-efficiency`. Accessed on: 15 Nov. 2024.

ENGIE. **Les réseaux de froid urbains**. 2020. Available in: `https://www.engie-solutions.com/fr/faire-economies-energies/reduire-consomation-chaleur-froid-electricite/reseaux-froid-urbains`. Accessed on: 16 Nov. 2024.

EUROPEAN COMMISSION. **Progress on Climate Action**. [*S.l.*: *s.n.*], 2023. `https://climate.ec.europa.eu/eu-action/climate-strategies-targets/progress-climate-action_en`. Accessed on: 16 Nov. 2024.

FOSSATI, Michele et al. Building energy efficiency: An overview of the Brazilian residential labeling scheme. **Renewable and Sustainable Energy Reviews**, v. 65, p. 1216–1231, 2016. ISSN 1364-0321. DOI: `10.1016/j.rser.2016.06.048`. Available in: `https://www.sciencedirect.com/science/article/pii/S1364032116302805`. Accessed on: 15 Nov. 2024.

GOOGLE DEVELOPPERS. **Descending into ML: Training and Loss**. 2022. Available in: `https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss`. Accessed on: 16 Nov. 2024.

GUPTA, Ruchi; SHARMA, Anupama; ALAM, Tanweer. Building Predictive Models with Machine Learning. In: [*S.l.*: *s.n.*], Mar. 2024. P. 39–59. ISBN 978-981-97-0447-7. DOI: `10.1007/978-981-97-0448-4_3`. Accessed on: 15 Nov. 2024.

GUYON, Isabelle; ELISSEEFF, Andre. An Introduction to Variable and Feature Selection. Ed. by Leslie Pack Kaelbling. **Journal of Machine Learning Research**, Journal of Machine Learning Research, Berkeley, CA, USA and Tübingen, Germany, v. 3, p. 1157–1182, Mar. 2003. Available in: `https://www.jmlr.org/papers/v3/guyon03a.html`. Accessed on: 16 Nov. 2024.

HAGELBÄCK, Johan. **Visualization of k-means clustering**. [*S.l.*], 2019. Available in: `https://www.youtube.com/watch?v=nXY6PxAaOk0&ab_channel=JohanHagelb%C3%A4ck`. Accessed on: 15 Nov. 2024.

HASTIE, Trevor; FRIEDMAN, Jerome; TIBSHIRANI, Robert. Additive Models, Trees, and Related Methods. In: THE Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York, NY: Springer New York, 2001. P. 257–298. Available in: `https://link.springer.com/book/10.1007/978-0-387-84858-7`. Accessed on: 15 Nov. 2024.

IBM. **What is a neural network?** [*S.l.*], 2017. Available in: `https://www.ibm.com/topics/neural-networks`. Accessed on: 15 Nov. 2024.

IEA. **Multiple benefits of energy efficiency: Energy savings**. 2024. Available in: `https://www.iea.org/energy-system/buildings`. Accessed on: 15 Nov. 2024.

IEA. **Multiple benefits of energy efficiency: Energy savings**. Disponível em: 2024. Available in: `https://www.iea.org/reports/multiple-benefits-of-energy-efficiency/energy-savings`. Accessed on: 15 Nov. 2024.

IEA. **Tracking buildings**. 2023. Available in: `https://www.iea.org/energy-system/buildings`. Accessed on: 15 Nov. 2024.

IFP. **Tout savoir sur le gaz naturel**. [*S.l.*], 2023. Available in: `https://www.ifpenergiesnouvelles.fr/enjeux-et-prospective/decryptages/energies-fossiles/tout-savoir-gaz-naturel`. Accessed on: 16 Nov. 2024.

INSTITUT MONTAIGNE. **Europe's Energy Transition: A Common Challenge**. [*S.l.*: *s.n.*], Sept. 2021. Report. Available in: `https://www.institutmontaigne.org/ressources/pdfs/publications/europes-energy-transition-common-challenge-report.pdf`. Accessed on: 16 Nov. 2024.

KRAEV, Egor et al. shap-select: Lightweight Feature Selection Using SHAP Values and Regression. **arXiv preprint**, 2024. arXiv: `2410.06815v1 [cs.LG]`. Available in: `https://arxiv.org/html/2410.06815v1`. Accessed on: 16 Nov. 2024.

KUHN, Max; JOHNSON, Kjell. **Applied Predictive Modeling**. New York, NY: Springer, 2013. ISBN 978-1-4614-6848-6. Available in: `https://link.springer.com/book/10.1007/978-1-4614-6849-3`. Accessed on: 15 Nov. 2024.

LE CAM, Mathieu; DAOUD, Ahmed; ZMEUREANU, R. Forecasting electric demand of supply fan using data mining techniques. **Energy**, v. 101, p. 541–557, Apr. 2016. DOI: `10.1016/j.energy.2016.02.061`. Accessed on: 15 Nov. 2024.

LE ROY, Benjamin et al. Long time series spatialized data for urban climatological studies: A case study of Paris, France. **International Journal of Climatology**, v. 40, n. 7, p. 3567–3584, 2020. DOI: `https://doi.org/10.1002/joc.6414`. Available in: `https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/joc.6414`. Accessed on: 16 Nov. 2024.

LIU. **Umass STAT 697F - Topics in Regression**. [*S.l.*], 2015. Available in: `https://people.math.umass.edu/~anna/stat697F/`. Accessed on: 15 Nov. 2024.

LUNDBERG, Scott. A unified approach to interpreting model predictions. **arXiv preprint arXiv:1705.07874**, 2017. Available in: `https://www.planchet.net/EXT/ISFA/1226.nsf/769998e0a65ea348c1257052003eb94f/02b26cfa6ecc8cd3c12583d9006de8c2/$FILE/7062-a-unified-approach-to-interpreting-model-predictions.pdf`. Accessed on: 15 Nov. 2024.

MALEKI, Farhad et al. Machine Learning Algorithm Validation. **Neuroimaging Clinics of North America**, v. 30, p. 433–445, Nov. 2020. DOI: `10.1016/j.nic.2020.08.004`. Accessed on: 16 Nov. 2024.

MALEKI, Farhad et al. Machine Learning Algorithm Validation. **Neuroimaging Clinics of North America**, v. 30, p. 433–445, Nov. 2020. DOI: `10.1016/j.nic.2020.08.004`. Accessed on: 16 Nov. 2024.

MONTAIGNE, Institut. **Europe’s Energy Transition: A Commun Challenge**. 2021. Available in: `https://www.institutmontaigne.org/ressources/pdfs/publications/europes-energy-transition-common-challenge-report.pdf`. Accessed on: 15 Nov. 2024.

MTE. **Éco Énergie Tertiaire (EET)**. [*S.l.*], 2023. Available in: `https://www.ecologie.gouv.fr/eco-energie-tertiaire-eet`. Accessed on: 15 Nov. 2024.

MTE. **Énergie dans les bâtiments**. 2021. Available in: `https://www.ecologie.gouv.fr/energie-dans-batiments`. Accessed on: 15 Nov. 2024.

MTE. **Loi de transition énergétique pour la croissance verte**. 2017. Available in: `https://www.ecologie.gouv.fr/loi-transition-energetique-croissance-verte`. Accessed on: 15 Nov. 2024.

OPERA. **Consommation d’énergie des bâtiments tertiaires : chiffres clés et objectifs**. 2023. Available in: `https://opera-energie.com/consommation-energie-batiments-tertiaires`. Accessed on: 15 Nov. 2024.

OPERA. **Audit énergétique dans le tertiaire : obligations et financement**. [*S.l.*], 2023. Available in: `https://opera-energie.com/audit-energetique-tertiaire/`. Accessed on: 15 Nov. 2024.

PAUDEL, Subodh. **Méthodologie pour estimer la consommation d'énergie dans les bâtiments en utilisant des techniques d'intelligence artificielle**. 2016. PhD thesis – Ecole des Mines de Nantes. Available in: `https://theses.hal.science/tel-01382882v1/file/Paudel_S_09_2016.pdf`. Accessed on: 15 Nov. 2024.

PBE EDIFICA. **Etiquetagem de edificações comerciais e de serviços**. [*S.l.*: *s.n.*], 2020. Available in: `https://pbeedifica.com.br/`. Accessed on: 16 Nov. 2024.

PUGLIESE, Raffaele; REGONDI, Stefano; MARINI, Riccardo. Machine learning-based approach: global trends, research directions, and regulatory standpoints. **Data Science and Management**, v. 4, p. 19–29, 2021. ISSN 2666-7649. DOI: `https://doi.org/10.1016/j.dsm.2021.12.002`. Available in: `https://www.sciencedirect.com/science/article/pii/S2666764921000485`. Accessed on: 15 Nov. 2024.

RASCHKA, Sebastian; LIU, Yuxi Hayden; MIRJALILI, Vahid. **Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python**. [*S.l.*]: Packt Publishing Ltd, 2022. Accessed on: 15 Nov. 2024.

RASHIDI, Hooman H. et al. Common statistical concepts in the supervised Machine Learning arena. **Frontiers in Oncology**, v. 13, 2023. Available in: `https://www.frontiersin.org/journals/oncology/articles/10.3389/fonc.2023.1130229`. Accessed on: 15 Nov. 2024.

RAY, Sunil. **Top 10 Machine Learning Algorithms (with Python and R Codes)**. [*S.l.*], 2023. Available in: `https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/`. Accessed on: 15 Nov. 2024.

RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. "Why Should I Trust You?": Explaining the Predictions of Any Classifier, 2016. arXiv: `1602.04938 [cs.LG]`. Available in: `https://arxiv.org/abs/1602.04938`. Accessed on: 15 Nov. 2024.

ROZEMBERCZKI, Benedek et al. The shapley value in machine learning. **arXiv preprint arXiv:2202.05594**, 2022. Available in: `https://arxiv.org/pdf/2202.05594`. Accessed on: 15 Nov. 2024.

SAH, Shagan. **Machine Learning: A Review of Learning Types**. [*S.l.*], 2020. DOI: `10.20944/preprints202007.0230.v1`. Accessed on: 15 Nov. 2024.

SARKER, Iqbal H. Machine Learning: Algorithms, Real-World Applications and Research Directions. **SN Computer Science**, v. 2, n. 3, p. 160, 2021. DOI:

10.1007/s42979-021-00592-x. Available in: https://doi.org/10.1007/s42979-021-00592-x. Accessed on: 15 Nov. 2024.

SCHUTZE, Amanda; HOLZ, Rhayana; ASSUNÇÃO, Juliano. **Eficiência Energética (EE) no Brasil e no Mundo: Mecanismos das Políticas de EE em Unidades Consumidoras Intensivas de Eletricidade**. Rio de Janeiro: Climate Policy Initiative, 2022. Available in: https://www.climatepolicyinitiative.org/wp-content/uploads/2023/01/230109-REL-GIZ-Mecanismos-de-EE.pdf. Accessed on: 15 Nov. 2024.

SERVICES EN BÂTIMENT, Société de. **L'isolation thermique de mon logement: Comment et pourquoi ?** 2024. Available in: https://www.ssb.fr/lisolation-thermique-de-mon-logement-comment-et-pourquoi/. Accessed on: 16 Nov. 2024.

SHARMA, Sunil Kumar et al. Retrofitting Existing Buildings to Improve Energy Performance. **Sustainability**, v. 14, n. 2, 2022. Available in: https://www.mdpi.com/2071-1050/14/2/666. Accessed on: 15 Nov. 2024.

SHAW, Bradley. **Filling in the Gaps: Imputation 3 Ways**. 2021. Available in: https://towardsdatascience.com/filling-in-the-gaps-imputation-3-ways-6056c09b6417. Accessed on: 15 Nov. 2024.

SHCHETININ, Eugene Yu. Modeling the energy consumption of smart buildings using artificial intelligence. In: K. E. SAMOUYLOV L. A. SEVASTIANOV, D. S. Kulyabov (Ed.). **Selected Papers of the IX Conference "Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems"**. Moscow, Russia: CEUR-WS, 2019. Leningradsky pr. 49, Moscow, 117198, Russia. Available in: https://ceur-ws.org/Vol-2407/paper-14-163.pdf. Accessed on: 15 Nov. 2024.

SOUZA, Renan et al. Workflow provenance in the lifecycle of scientific machine learning. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 34, n. 14, e6544, 2022. Available in: https://arxiv.org/abs/1602.04938. Accessed on: 15 Nov. 2024.

SUBASI, Abdulhamit. Chapter 1 - Introduction. In: SUBASI, Abdulhamit (Ed.). **Practical Machine Learning for Data Analysis Using Python**. [*S.l.*]: Academic Press, 2020. P. 1–26. ISBN 978-0-12-821379-7. DOI: https://doi.org/10.1016/B978-0-12-821379-7.00001-1. Available in: https://www.sciencedirect.com/science/article/pii/B9780128213797000011. Accessed on: 15 Nov. 2024.

SUNDARARAJAN, Mukund; NAJMI, Amir. **The many Shapley values for model explanation**. [*S.l.*: *s.n.*], 2020. arXiv: 1908.08474 [cs.AI]. Available in: https://arxiv.org/abs/1908.08474. Accessed on: 16 Nov. 2024.

TAYLOR, Rhian et al. Sensitivity Analysis for Deep Learning: Ranking Hyper-parameter Influence. In: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI). [*S.l.*: *s.n.*], 2021. P. 512–516. DOI: 10.1109/ICTAI52525.2021.00083. Accessed on: 16 Nov. 2024.

TEMIZEL, Cenk et al. Production Forecasting in Shale Reservoirs Using LSTM Method in Deep Learning. In. DOI: 10.15530/urtec-2020-2878.

TOUZANI, Samir; GRANDERSON, Jessica; FERNANDES, Samuel. Gradient boosting machine for modeling the energy consumption of commercial buildings. **Energy and Buildings**, v. 158, p. 1533–1543, 2018. ISSN 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2017.11.039. Available in: https://www.sciencedirect.com/science/article/pii/S0378778817320844. Accessed on: 15 Nov. 2024.

U.S.GOVERNMENT. **Energy Efficiency in Buildings and Industry**. 2024. Available in: https://www.energy.gov/eere/energy-efficiency-buildings-and-industry#:~:text=Energy%20efficiency%20is%20the%20use,less%20energy%20to%20produce%20goods.. Accessed on: 15 Nov. 2024.

UNDP. **Sustainable Energy Hub**. 2024. Available in: https://www.undp.org/energy/our-work-areas/energy-transition. Accessed on: 15 Nov. 2024.

URIONA-MALDONADO, Mauricio; VAZ, Caroline R.; ZAGHI, Lucca M. Real State Price Estimation in Brazil Using Machine Learning. In: **Knowledge Management and Artificial Intelligence for Growth: Cases from Emerging and Developed Economies**. Ed. by Isaias Bianchi and Guillermo Antonio Davila. Cham: Springer Nature Switzerland, 2024. P. 137–163. ISBN 978-3-031-65552-4. DOI: 10.1007/978-3-031-65552-4_8. Available in: https://doi.org/10.1007/978-3-031-65552-4_8. Accessed on: 16 Nov. 2024.

VERSAGE, Rogério. **METAMODELO PARA ESTIMAR A CARGA TÉRMICA DE EDIFICAÇÕES CONDICIONADAS ARTIFICIALMENTE**. 2015. PhD thesis – Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil. Available in: https://repositorio.ufsc.br/bitstream/handle/123456789/169362/337477.pdf. Accessed on: 15 Nov. 2024.

VERTIGO. **Décret tertiaire - Un enjeu économique, écologique et réglementaire**. 2023. Available in: https://vertigo-energy.com/accueil-b2b/entreprises-et-collectivites/entreprises-et-collectivites-offres-pro-decret-tertiaire/. Accessed on: 15 Nov. 2024.

WAKEFIELD, Katrina. **Predictive Modeling Analytics and Machine Learning**. 2021. Available in: https://www.sas.com/en_gb/insights/manuals/analytics/a-guide-to-predictive-analytics-and-machine-learning.html. Accessed on: 15 Nov. 2024.

WEERTS, Hilde J. P.; MUELLER, Andreas C.; VANSCHOREN, Joaquin. **Importance of Tuning Hyperparameters of Machine Learning Algorithms**. [*S.l.*: *s.n.*], 2020. arXiv: 2007.07588 [cs.LG]. Available in: https://arxiv.org/abs/2007.07588. Accessed on: 15 Nov. 2024.

WICKRAMASINGHE. **Bias–Variance Tradeoff in Machine Learning: Concepts Tutorials**. [*S.l.*], 2024. Available in: https://www.bmc.com/blogs/bias-variance-machine-learning/. Accessed on: 15 Nov. 2024.

XU, Yujie. **Using Machine Learning to Target Retrofits in Commercial Buildings under Alternative Climate Change Scenarios**. 2020. PhD thesis – Carnegie Mellon University. Available in: https://kilthub.cmu.edu/articles/thesis/Using_Machine_Learning_to_Target_Retrofits_in_Commercial_Buildings_under_Alternative_Climate_Change_Scenarios/14454648. Accessed on: 16 Nov. 2024.

ZENG, Xianlong. Enhancing the Interpretability of SHAP Values Using Large Language Models. **arXiv preprint arXiv:2409.00079**, 2024. Available in: https://arxiv.org/pdf/2409.00079. Accessed on: 15 Nov. 2024.

ZHOU, S.L. et al. A comprehensive review of the applications of machine learning for HVAC. **DeCarbon**, v. 2, p. 100023, 2023. ISSN 2949-8813. DOI: https://doi.org/10.1016/j.decarb.2023.100023. Available in: https://www.sciencedirect.com/science/article/pii/S2949881323000239. Accessed on: 16 Nov. 2024.

ZHOU, Zhi-Hua. **Machine learning**. [*S.l.*]: Springer Nature, 2021. Available in: https://www.statistiques.developpement-durable.gouv.fr/edition-numerique/chiffres-cles-energie-2021/6-bilan-energetique-de-la-france. Accessed on: 15 Nov. 2024.

ZUNE, May et al. A review of traditional multistage roofs design and performance in vernacular buildings in Myanmar. **Sustainable Cities and Society**, v. 60, p. 102240, 2020. ISSN 2210-6707. DOI: https://doi.org/10.1016/j.scs.2020.102240. Available in: https://www.sciencedirect.com/science/article/pii/S2210670720304613. Accessed on: 15 Nov. 2024.

ANNEX A – APPENDIX

A.1 CORRELATION MATRIX

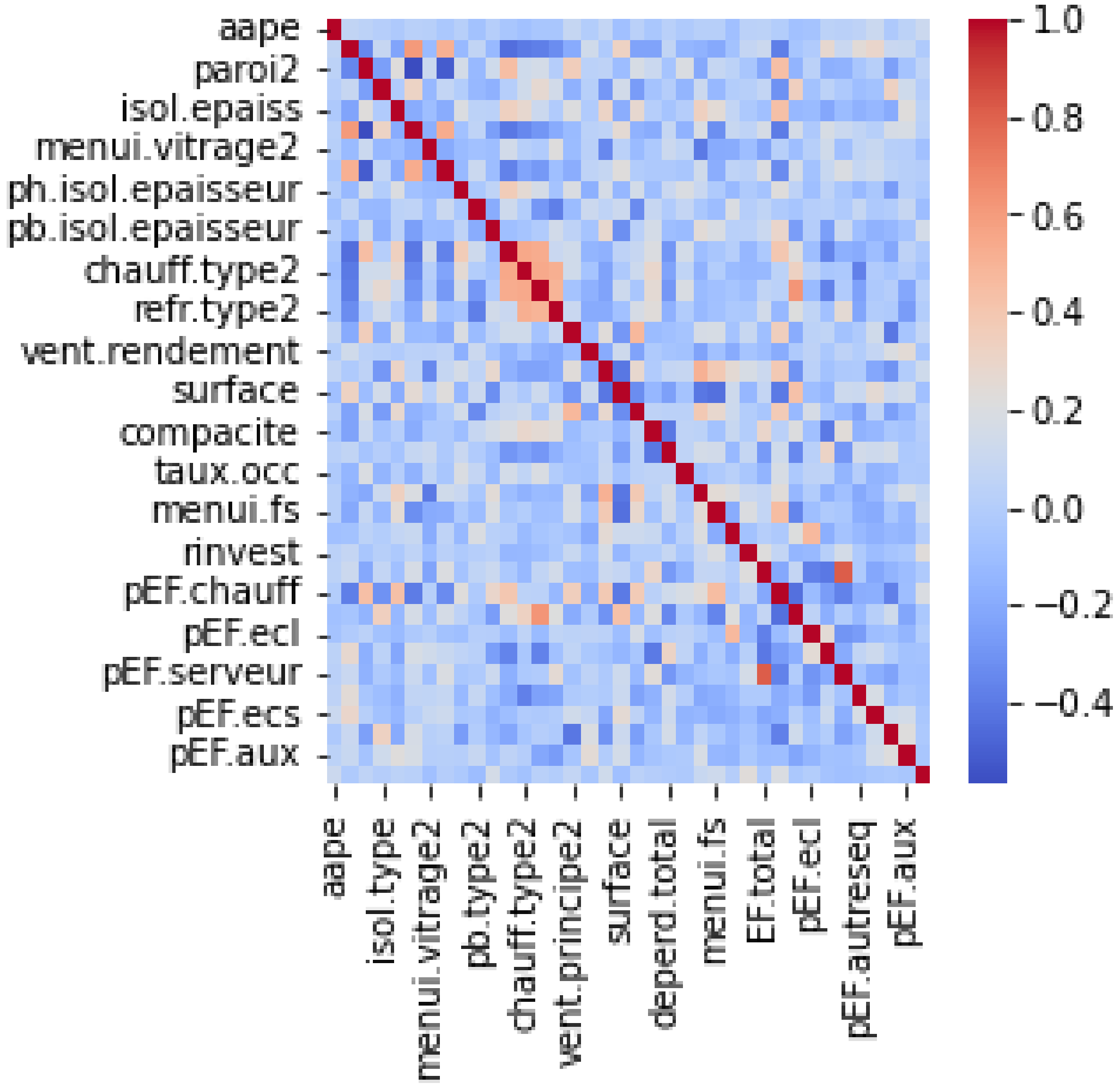


Figure 61 – Correlation matrix

A.2 ELBOW AND SILLHOUETTE METHODS FOR CLUSTERS

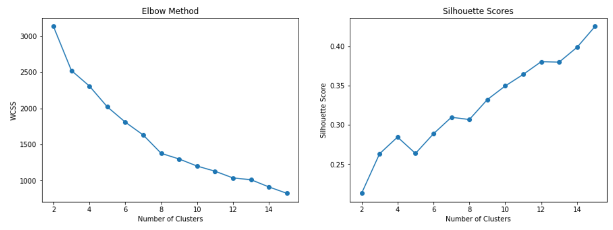


Figure 62 – Envelope Clusters

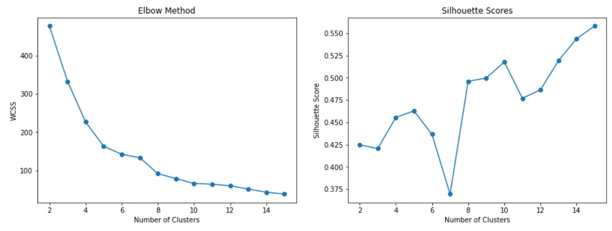


Figure 63 – Lighting Clusters

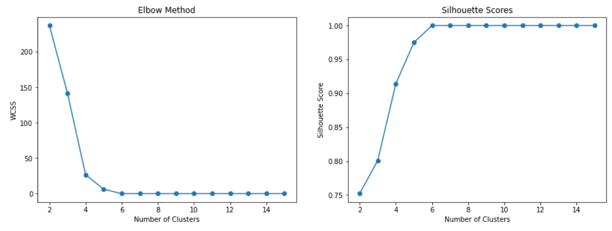


Figure 64 – Heating Clusters

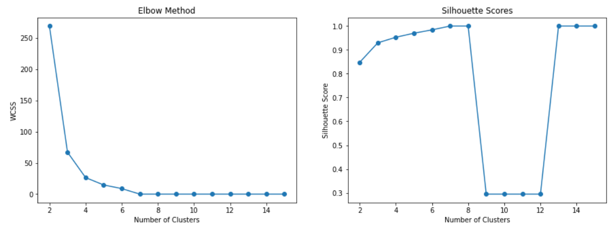


Figure 65 – Cooling Clusters

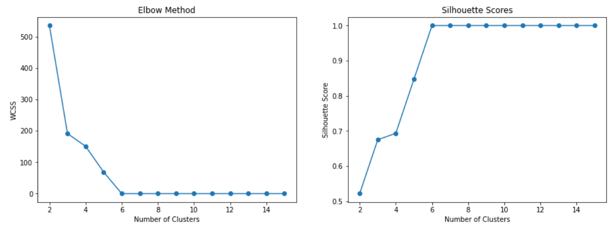


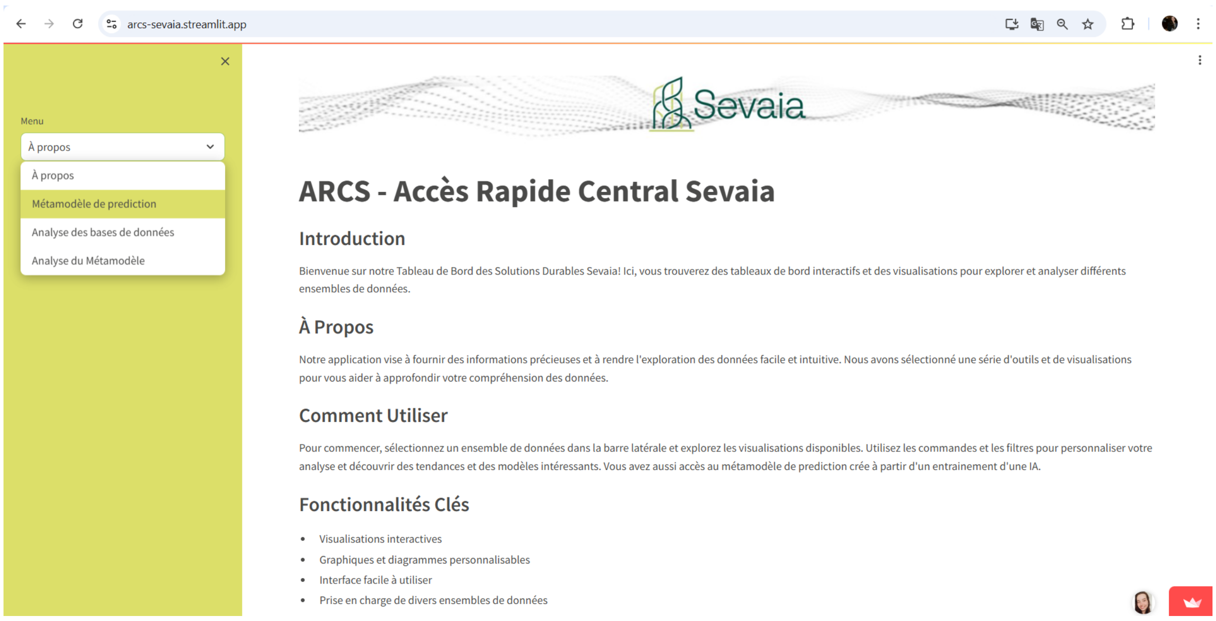
Figure 66 – Ventilation Clusters

ANNEX B – APPENDIX

The model described in this project is associated with an application which is already being used by the company. The access to it is through the following link:

https://arcs-sevaia.streamlit.app/

The functionning of this interface is described through the following images:


arcs-sevaia.streamlit.app

Menu

À propos

À propos
Métamodèle de prediction
Analyse des bases de données
Analyse du Métamodèle

Sevaia

ARCS - Accès Rapide Central Sevaia

Introduction

Bienvenue sur notre Tableau de Bord des Solutions Durables Sevaia! Ici, vous trouverez des tableaux de bord interactifs et des visualisations pour explorer et analyser différents ensembles de données.

À Propos

Notre application vise à fournir des informations précieuses et à rendre l'exploration des données facile et intuitive. Nous avons sélectionné une série d'outils et de visualisations pour vous aider à approfondir votre compréhension des données.

Comment Utiliser

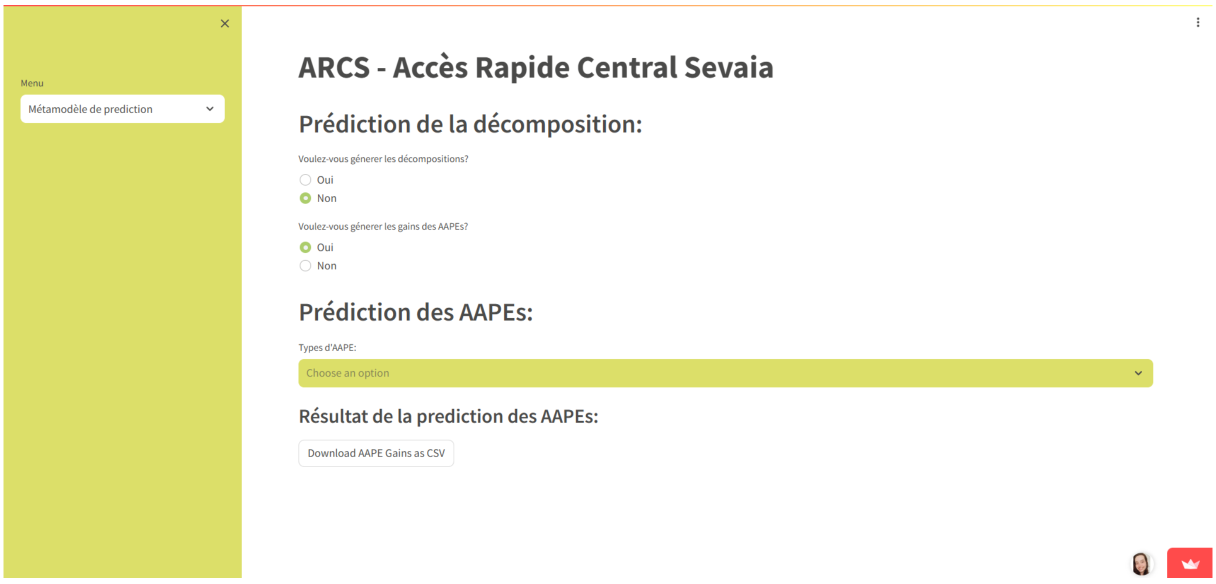
Pour commencer, sélectionnez un ensemble de données dans la barre latérale et explorez les visualisations disponibles. Utilisez les commandes et les filtres pour personnaliser votre analyse et découvrir des tendances et des modèles intéressants. Vous avez aussi accès au métamodèle de prediction crée à partir d'un entrainement d'une IA.

Fonctionnalités Clés

- Visualisations interactives
- Graphiques et diagrammes personnalisables
- Interface facile à utiliser
- Prise en charge de divers ensembles de données

Figure 67 – Home Page of Sevaia's data exploration center

As shown above, from the home page, a main menu is shown that allows the access to the prediction models. Then, the user must choose to generate the predictions to the AAPE (retrofit) by choosing 'Oui' in the Horizontal menu as shown below.


Menu

Métamodèle de prediction

ARCS - Accès Rapide Central Sevaia

Prédiction de la décomposition:

Voulez-vous génerer les décompositions?

Oui [✓ Non]

Voulez-vous génerer les gains des AAPEs?

[✓ Oui] Non

Prédiction des AAPEs:

Types d'AAPE:

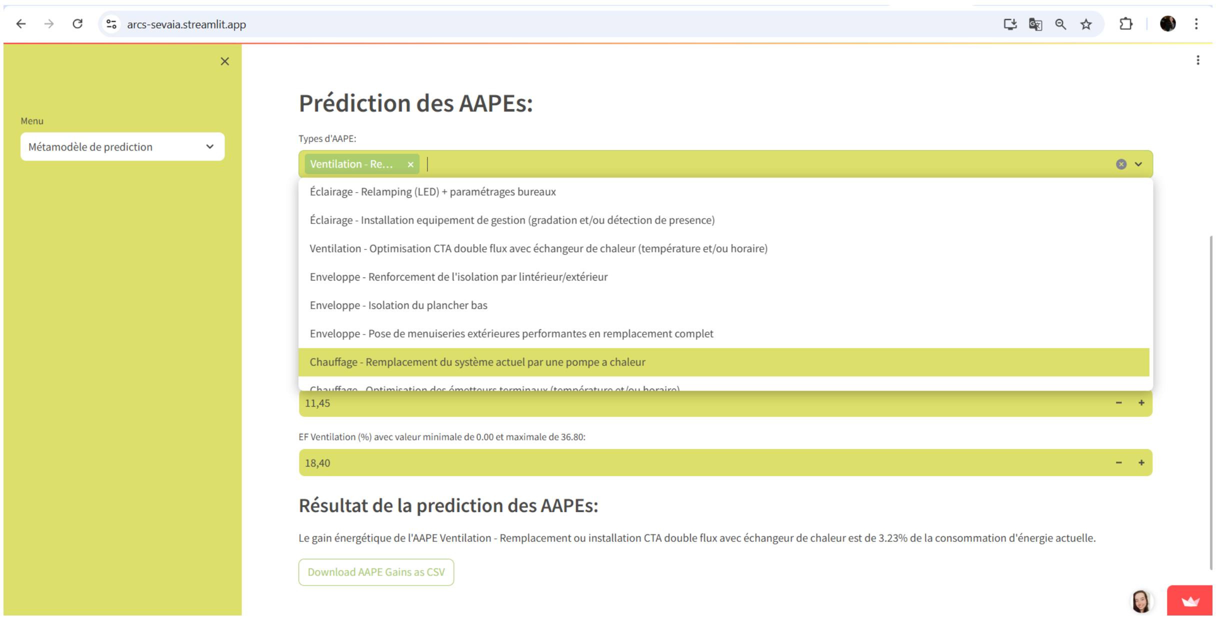
Choose an option

Résultat de la prediction des AAPEs:

Download AAPE Gains as CSV

Figure 68 – Lateral menu accessing Sevaia's webpage functionalities

Finally, the user can chose which retrofit is to be predicted, fill the parameters necessary and then, visualize the energy savings or export them to a CSV file. As many parameters are present in more than one retrofit, the interface allows the model to run simultaneously retrofit while filling the parameters only once, simplifying the user experience.


arcs-sevaia.streamlit.app

Menu

Métamodèle de prediction

Prédiction des AAPEs:

Types d'AAPE:

Ventilation - Re... ×

Éclairage - Relamping (LED) + paramétrages bureaux

Éclairage - Installation equipement de gestion (gradation et/ou détection de presence)

Ventilation - Optimisation CTA double flux avec échangeur de chaleur (température et/ou horaire)

Enveloppe - Renforcement de l'isolation par lintérieur/extérieur

Enveloppe - Isolation du plancher bas

Enveloppe - Pose de menuiseries extérieures performantes en remplacement complet

Chauffage - Remplacement du système actuel par une pompe a chaleur

11,45

EF Ventilation (%) avec valeur minimale de 0.00 et maximale de 36.80:

18,40

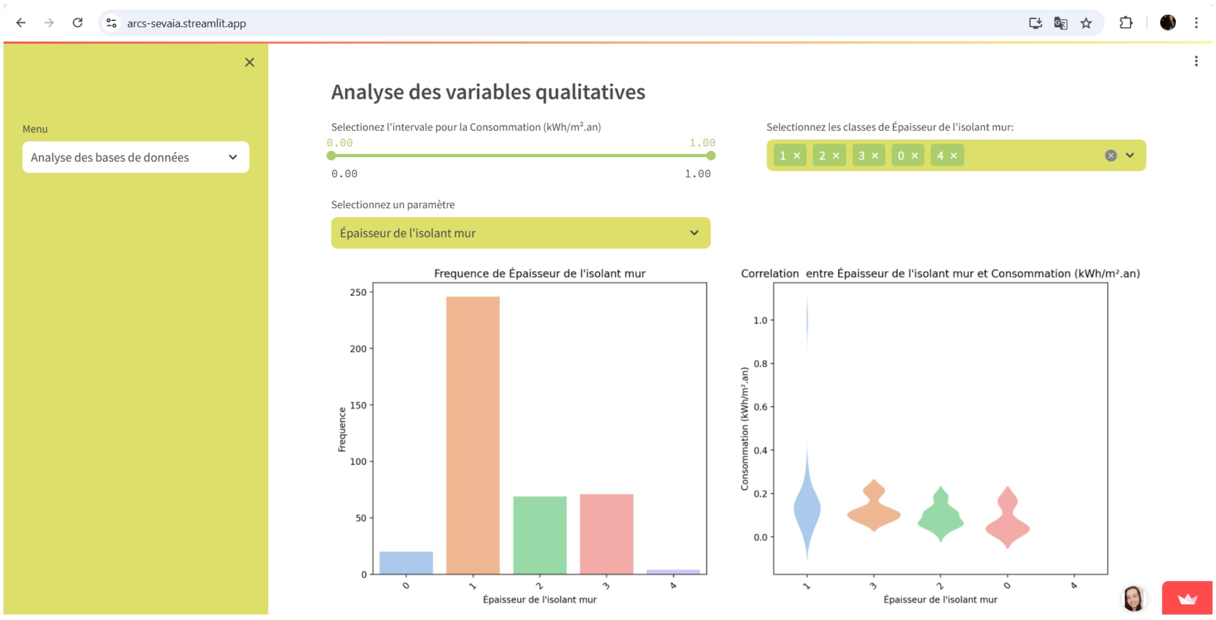
Résultat de la prediction des AAPEs:

Le gain énergétique de l'AAPE Ventilation - Remplacement ou installation CTA double flux avec échangeur de chaleur est de 3.23% de la consommation d'énergie actuelle.

Download AAPE Gains as CSV

Figure 69 – Sevaia's prediction models

Other functionality available is the possible exploration of the databases parameters correlating with the energy consumption in the building.


arcs-sevaia.streamlit.app

Menu

Analyse des bases de données

Analyse des variables qualitatives

Selectionez l'intervale pour la Consommation (kWh/m².an)

0.00 – 1.00

Selectionnez les classes de Épaisseur de l'isolant mur:

1 × 2 × 3 × 0 × 4 ×

Selectionez un paramètre

Épaisseur de l'isolant mur

Frequence de Épaisseur de l'isolant mur
250
200
150
100
50
0
Frequence
0
1
2
3
4
Épaisseur de l'isolant mur
Correlation entre Épaisseur de l'isolant mur et Consommation (kWh/m².an)
1.0
0.8
0.6
0.4
0.2
0.0
Consommation (kWh/m².an)
1
3
2
0
4
Épaisseur de l'isolant mur

Figure 70 – Sevaia's prediction models

ANNEX C – APPENDIX

C.1 PYTHON CODES USED IN THE RESEARCH

C.1.1 Code 1 - Synthesis Analysis

```

"""
Created on Thu Aug 17 09:57:14 2023

@author: LorranyDASILVA
"""
import pandas as pd
import matplotlib.pyplot as plt

df =
    pd.read_csv('C:/Users/LorranyDASILVA/Documents/PFE/Code/Synthese-Audits_enter.csv',sep=
#column_names = df.columns.tolist()

# Writing it correctly
replacements = {'  ':'a','  ':'a','  ':'e','  ':'e'
    ,'  ':'e','   ':'<','  ':'c','  ':'n','#DIV/0!':'0'}
df = df.replace(replacements, regex=True)

# Subset bureaux
df = df[df.usage == 'bureaux']
# dos 501 casos, 209 sao bureaux
df = df[(df.depto == '75') | (df.depto == '92') | (df.depto == '93') |
    (df.depto == '94')]
#subset Paris: 75, 92,93 e 94 - Para os bureaux havia apenas 7 casos 78 , 1
    caso 95 1 caso 91 - preferencia por adotar Paris et banlieu
df = df[(df.methode2 == 'VE') | (df.methode2 == 'CW') | (df.methode2 == 'P+C') ]



df['pEF_chauff'] = df['rEF_chauff']/df['rEF_total']
df['pEF_froid'] = df['rEF_froid']/df['rEF_total']
df['pEF_ecl'] = df['rEF_ecl']/df['rEF_total']
df['pEF_bureautique'] = df['rEF_bureautique']/df['rEF_total']
df['pEF_serveur'] = df['rEF_serveur']/df['rEF_total']
df['pEF_autreseq'] = df['rEF_autreseq']/df['rEF_total']
df['pEF_ecs'] = df['rEF_ecs']/df['rEF_total']
df['pEF_vent'] = df['rEF_vent']/df['rEF_total']
df['pEF_aux'] = df['rEF_aux']/df['rEF_total']


df = df.drop(columns = [

 # IDENTIFICATION COLUMNS --------->
'nom','depto','proprietaire','affaire','projet','secteur', #DROP THESE COLUMNS
    ONLY AFTER YOU MADE A SAFE ITERATION IN THE CODE_ASSAMBLAGE_BDD

# USELESS COLUMNS --------->
'usage','date','annee','methode1','auditeur','n_occ',


# GENERAL MAY BE USED TO ADVENIO'S INTERNAL CONTROL
'ref_reglementaire','ecart_global','ecart_chauff','ecart_froid',
'ecart_ecs','ecart_vent','ecart_ecl','ecart_aux',

# REGLEMENTATION
'cc_global','cc_chauff','cc_froid','cc_ecs','cc_vent','cc_ecl','cc_aux',
'cp_total','cp_chauff1.type','cp_chauff2.type','cp_froid1.type','cp_froid2.type',
'cp_ecs1.type','cp_ecs2.type','cp_chauff1','cp_chauff2','cp_froid1','cp_froid2','cp_ecl',
'cp_bureautique','cp_serveurs','cp_autreseq',
'cp_ecs1','cp_ecs2','cp_vent','cp_aux','cp_divers',
'methode2','instrumentation',

# ON VA UTILISER JUSTE L'ENERGIE EN PERCENTAGE
'rEF_total','rEF_chauff','rEF_froid','rEF_ecl','rEF_bureautique',
'rEF_serveur','rEF_autreseq','rEF_ecs','rEF_vent','rEF_aux','rEF_divers',
'rEP_total','rEP_chauff','rEP_froid','rEP_ecl','rEP_bureautique',
'rEP_serveurs','rEP_autreseq','rEP_ecs','rEP_vent','rEP_aux','rEP_divers',
'rENV_total','rENV_chauff','rENV_froid','rENV_ecl','rENV_bureautique',
'rENV_serveurs','rENV_autreseq','rENV_ecs','rENV_vent','rENV_aux','rENV_divers',
'rFACT_total','rFACT_chauff','rFACT_froid','rFACT_ecl','rFACT_bureautique','rFACT_serveurs
'rFACT_autreseq','rFACT_ecs','rFACT_vent','rFACT_aux','rFACT_divers',


#DEPERDITIONS EN PLUS
'deperd_trans','deperd_vent','deperd_infiltration',

#COLUMNS THAT ARE REDOUNDANT OR SUBSTITUTED BY OTHERS
'bureau_occ','serveur_occ','categorie','construction','renovation','rie','structure',

#COLUMNS THAT WERE SIMPLIFIED
'paroi','menui_type', 'menui_vitrage', 'menui_tl',  'menui_protect', 'ph_type',
    'pb_type',
'chauff_type','refr_type','vent_principe','ecs','ecs_type','emission_chauff','emission_refr
'ecl_gestion','gtb' ])


df.to_csv(f'C:/Users/LorranyDASILVA/Documents/PFE/Code/Synthese-Audits_enter_cdropped.csv',
    index=False)

# Compacite 0 et 5000 enleve , Tire - dans UW enleve et Dperditions 0 enlevees
    et un Facteur Solaire de 45 qui devient 0,45

df =
    pd.read_csv('C:/Users/LorranyDASILVA/Documents/PFE/Code/Synthese-Audits_enter_cdropped.

selec_quant
    =['surface','u_bat','compacite','deperd_total','taux_occ','menui_uw','menui_fs','ecl_pu
'pEF_chauff','pEF_froid','pEF_ecl','pEF_bureautique','pEF_serveur',
'pEF_autreseq','pEF_ecs','pEF_vent','pEF_aux']

selec_quali = [
'construction2','paroi2','isol_type','isol_epaiss','menui_type2',
'menui_vitrage2','menui_protect2','ph_type2','ph_isol_epaisseur','pb_type2',
'pb_isol_epaisseur','chauff','chauff_type2','refr','refr_type2',
'vent_principe2', 'vent_rendement', 'ecl_gestion2']


```

```
df_qualitative = df[selec_quali]
df_quantitative = df[selec_quant]


# Create a new DataFrame to store the most frequent values
most_frequent_values = pd.Series(dtype=object)

# Create a new DataFrame to store df_qualitative2
df_qualitative2 = df_qualitative.copy()

for col in df_qualitative:
    # Calculate the number of empty rows
    empty_rows = df_qualitative[col].isna().sum()

    # Calculate the most frequent value
    most_frequent_value = df_qualitative[col].mode().values[0]

    # Store the most frequent value
    most_frequent_values[col] = most_frequent_value

    # Plot bar chart
    value_counts = df_qualitative[col].value_counts()
    plt.figure(figsize=(15, 6))
    plt.bar(value_counts.index, value_counts.values, alpha=0.7,
        color='palevioletred')
    plt.xlabel(col)
    plt.ylabel('Frequ ncia')
    plt.title(f'Bar chart of {col} (Empty Rows: {empty_rows}, Most Frequent:
        {most_frequent_value})')
    plt.xticks(rotation=90)
    for i, v in enumerate(value_counts.values):
        plt.text(i, v + 0.5, str(v), color='black', ha='center')
    plt.grid(True)
    plt.savefig(f'00Quali_{col}_AS.png', bbox_inches='tight')
    plt.show()

    # Fill NA values with the most frequent value in df_qualitative2
    df_qualitative2[col].fillna(most_frequent_value, inplace=True)






for col in df_qualitative2:

    # Plot bar chart
    most_frequent_value = df_qualitative2[col].mode().values[0]
    # Store the most frequent value

    most_frequent_values[col] = most_frequent_value
    value_counts = df_qualitative2[col].value_counts()
    plt.figure(figsize=(15, 6))
    plt.bar(value_counts.index, value_counts.values, alpha=1, color='pink')
    plt.xlabel(col)
    plt.ylabel('Frequ ncia')
```

```python
    plt.title(f'Bar chart of {col} (Empty Rows filled with: {most_frequent_value})')
    plt.xticks(rotation=90)
    for i, v in enumerate(value_counts.values):
        plt.text(i, v + 0.5, str(v), color='black', ha='center')
    plt.grid(True)
    plt.savefig(f'00Quali_{col}_AS_fill.png', bbox_inches='tight')
    plt.show()

# Create a new DataFrame to store medians
medians = pd.Series(dtype=float)

for col in df_quantitative:
  # Calculate the count of NaN values in the column
    nan_count = df_quantitative[col].isna().sum()

    # Filter the non-NaN values in the column
    non_nan_values = df_quantitative[col].dropna()
    non_nan_values = non_nan_values.astype(float)

    # Calculate the median of non-NaN values
    median_value = non_nan_values.median()

    # Store the median value in the medians DataFrame
    medians[col] = median_value

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
    # Plot histogram on the first subplot
    ax1.hist(non_nan_values, bins=30, color='deepskyblue', alpha=0.7)
    ax1.set_xlabel(col)
    ax1.set_title(f'Histrogram of {col} - (NaN Count: {nan_count})')  # Display median in the title
    ax1.grid(True)

    # Plot violinplot on the second subplot
    ax2.violinplot(non_nan_values, vert=False)
    ax2.boxplot(non_nan_values, vert=False)
    ax2.set_xlabel(col)
    ax2.set_title(f'Boxplot of {col} (Median={median_value:.2f})')  # Display median in the title
    ax2.grid(True)

    plt.savefig(f'00Quanti_{col}_AS.png', bbox_inches='tight')
    plt.show()

# Create a new DataFrame df_quantitative2 with NaN values filled with medians
df_quantitative2 = df_quantitative.fillna(medians)

for col in df_quantitative2:

    preenchido = df_quantitative2[col]
    preenchido = preenchido.astype(float)
    # Calculate the median of non-NaN values
    median_value = preenchido.median()
    # Store the median value in the medians DataFrame
    medians[col] = median_value


    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
    # Plot histogram on the first subplot
    ax1.hist(preenchido, bins=30, color='lightskyblue', alpha=0.7)
    ax1.set_xlabel(col)
    ax1.set_title(f'Histogram of {col} - empty rows filled with the median: {median_value:.2f}')  # Display median in the title
    ax1.grid(True)

    # Plot violinplot on the second subplot
    ax2.violinplot(preenchido, vert=False)
    ax2.boxplot(preenchido, vert=False)
    ax2.set_xlabel(col)
    ax2.set_title(f'Boxplot of {col} - empty rows filled with the median: {median_value:.2f}')  # Display median in the title
    ax2.grid(True)

    plt.savefig(f'00Quanti_{col}_AS_fill.png', bbox_inches='tight')
    plt.show()

#Finally, the assembly of both of them
tableau_synthese = pd.concat([df_quantitative2, df_qualitative2], axis=1)
tableau_synthese.to_csv('Tableau_Synthese.csv', index=False)
```

Listing C.1 – Code 1 - Synthesis Analysis

C.1.2 Code 2 - AAPE Analysis

```
# -*- coding: utf-8 -*-
"""
Created on Wed Aug 23 11:45:02 2023
"""
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('C:/Users/LorranyDASILVA/Documents/PFE/Code/Synthese-AAPE_enter.csv',sep=";
column_names = df.columns.tolist()

all_col = [
 'date', 'methode1', 'cle', 'reference', 'libelle', 'surface',
 'invest', 'rinvest', 'FACT_gain', 'FACT_gainR', 'FACT_gainP', 'tri', 'cee',
 'EFg', 'rEFg', 'pEFg', 'Epg', 'rEPg', 'pEPg', 'ENVg', 'rENVg', 'pENVg',
 'RTg_INI', 'RTg_REF', 'EFg_chauff.elec', 'rEFg_chauff.elec', 'pEFg_chauff.elec',
 'EFg_chauff.gaz', 'rEFg_chauff.gaz', 'pEFg_chauff.gaz',
 'EFg_chauff.ru', 'rEFg_chauff.ru', 'pEFg_chauff.ru', 'EFg_chauff', 'rEFg_chauff', 'pEFg_chauff',
 'EFg_froid.elec', 'rEFg_froid.elec', 'pEFg_froid.elec', 'EFg_froid.ru', 'rEFg_froid.ru',
 'pEFg_froid.ru',
 'EFg_froid', 'rEFg_froid', 'pEFg_froid', 'EFg_ecl', 'rEFg_ecl', 'pEFg_ecl',
 'EFg_bureautique', 'rEFg_bureautique', 'pEFg_bureautique',
 'EFg_serveurs', 'rEFg_serveurs', 'pEFg_serveurs',
 'EFg_autreseq', 'rEFg_autreseq', 'pEFg_autreseq', 'EFg_ecs', 'rEFg_ecs',
     'pEFg_ecs',
 'EFg_vent', 'rEFg_vent', 'pEFg_vent', 'EFg_aux', 'rEFg_aux', 'pEFg_aux',
 'EFg_divers', 'rEFg_divers', 'pEFg_divers', 'libelle2']

# Writing it correctly
replacements = {'  ':'a','  ':'a','  ':'e','  ':'e'
    ,'  ':'e','  ':'<','  ':'c','  ':'n','  ':'o','  ':'o',
'  ':'2','  ':'E','  ':'i','  ':'<','  ':'  '}
df = df.replace(replacements, regex=True)

df = df.drop(columns = [
# identification columns --------->
'date','methode1','cle','reference',
# Financial aspects that may be a little deceiving - you shold test it anyways
'FACT_gain','FACT_gainR','FACT_gainP','tri','cee',
# columns that will maybe be useful in the futur
'invest','EFg','rEFg','pEFg',
# useless columns --------->
'Epg','rEPg','pEPg','ENVg','rENVg','pENVg','RTg_INI','RTg_REF',
'pEFg_chauff.elec','pEFg_chauff.gaz','pEFg_chauff.ru','pEFg_froid.elec','pEFg_froid.ru',
'EFg_chauff.elec','rEFg_chauff.elec','EFg_chauff.gaz','rEFg_chauff.gaz',
'EFg_chauff.ru','rEFg_chauff.ru','EFg_chauff','rEFg_chauff',
'EFg_froid.elec','rEFg_froid.elec','EFg_froid.ru','rEFg_froid.ru','EFg_froid','rEFg_froid'
'EFg_ecl','rEFg_ecl','EFg_bureautique','rEFg_bureautique','EFg_serveurs','rEFg_serveurs',
'EFg_autreseq','rEFg_autreseq','EFg_ecs','rEFg_ecs','EFg_vent','rEFg_vent','EFg_aux',
'rEFg_aux','EFg_divers','rEFg_divers',
'pEFg_serveurs', 'pEFg_autreseq', 'pEFg_ecs', 'pEFg_aux','pEFg_divers' ])

df.to_csv('Tableau_AAPE.csv', index=False)

# make some manual treatments that are basically to elimitate this : -  (des
    petits tires)
df =
    pd.read_csv('C:/Users/LorranyDASILVA/Documents/PFE/Code/Tableau_AAPE.csv',sep=";")


columns_to_drop = ['libelle', 'libelle2']
df_drop = df.drop(columns=columns_to_drop, axis=1)


# Create a new DataFrame to store medians
medians = pd.Series(dtype=float)

for col in df_drop:

    # Calculate the count of NaN values in the column
    nan_count = df_drop[col].isna().sum()

    # Filter the non-NaN values in the column
    non_nan_values = df_drop[col].dropna()
    non_nan_values = non_nan_values.astype(float)

    # Calculate the median of non-NaN values
    median_value = non_nan_values.median()

    # Store the median value in the medians DataFrame
    medians[col] = median_value

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
    # Plot histogram on the first subplot
    ax1.hist(non_nan_values, bins=30, color='salmon', alpha=0.7)
    ax1.set_xlabel(col)
    ax1.set_title(f'Histrogram of {col} - (NaN Count: {nan_count})')  # Display
        median in the title
    ax1.grid(True)

    # Plot violinplot on the second subplot
    ax2.violinplot(non_nan_values, vert=False,showextrema=True)
    ax2.boxplot(non_nan_values, vert=False)
    ax2.set_xlabel(col)
    ax2.set_title(f'Boxplot of {col} (Median={median_value:.2f})')  # Display
        median in the title
    ax2.grid(True)

    plt.savefig(f'01Quanti_{col}_AS.png', bbox_inches='tight')
    plt.show()

# Create a new DataFrame df_drop2 with NaN values filled with medians - but we
    don't use it
df_drop2 = df_drop.fillna(medians)
```

Listing C.2 – Code 2 - AAPE Analysis

C.1.3 Code 3 - Assamblage

```

# -*- coding: utf-8 -*-
"""
Created on Tue Aug 22 11:23:10 2023

@author: LorranyDASILVA
"""

import pandas as pd
#import matplotlib.pyplot as plt

a_data =
    pd.read_csv('C:/Users/LorranyDASILVA/Documents/PFE/Code/Tableau_AAPE.csv',sep=";")
s_data =
    pd.read_csv('C:/Users/LorranyDASILVA/Documents/PFE/Code/Tableau_Synthese.csv',sep=",")

a = a_data['surface'].unique()
a_df = pd.DataFrame(a,columns=['aape'])

s = s_data['surface']
s_df = s.to_frame()
s_df.columns = ['synthese']

data2 = pd.concat([s_df, a_df], axis=1)
data2['result'] = data2['synthese'].isin(data2['aape'].dropna())
count_true = data2['result'].sum()
```

```


subset_data2 = data2[data2['result'] == True]
subset_s = subset_data2[['synthese']]
subset_list = subset_s.values.tolist()
integer_list = [int(x[0]) for x in subset_list]



# Verificacao
    -------------------------------------------------------------------------

# Initialize a count variable
count_true = 0

# Iterate through elements in list A
for element in data2['aape']:
    # Check if element is in column B
    if element in integer_list:
        # Print the value in column B for the matching element
        print(f'Surface {element} found in Synthese with value: {element} in
            AAPE')
        # Increment the count
        count_true += 1

# Print the count of matches
print("Number of buildings in both data bases:", count_true)


# Filtragem das bases de dados
    -----------------------------------------------------


#  ATTENTION : THE ERROR SURFACE IS EASILY CORRECTED IN EXCEL, YOU NEED
    BASICALLY TO SELECT THE COLUMN, MAKE IT NUMERIC AND THEN NOT SHOW ANY ZEROS
#  ALSO, THERE IS A LINE WITH ZEROS IN THE a TABLE - YOU NEED TO EXCLUDE IT
    BEFORE MAKING THE TRANSFORMATION BELOW

s_data['surface'] = s_data['surface'].astype(int)
a_data['surface'] = a_data['surface'].astype(int)

s_data = s_data[s_data['surface'].isin(integer_list)]
a_data = a_data[a_data['surface'].isin(integer_list)]

s_data = s_data.sort_values(by='surface')
a_data = a_data.sort_values(by='surface')


# Verification of incompatibilities between the two databases
    ---------------------------------


qqty = a_data['surface'].unique()
qqty = pd.DataFrame(qqty,columns=['surface'])
qqty = qqty.values.tolist()
qqty = [int(x[0]) for x in qqty]

```

```
qqty2 = s_data['surface']
qqty2 = pd.DataFrame(qqty2,columns=['surface'])
qqty2 = qqty2.values.tolist()
qqty2 = [int(x[0]) for x in qqty2]

boolean_expression = [x in qqty2 for x in qqty]
#print("The error is in the building which area is :", boolean_expression)
# Count the number of False statements

false_count = sum(1 for statement in boolean_expression if not statement)

print("Number of uncompatibilities between the AAPE and the Synthese bases:",
    false_count)


# Integracao das duas bases de dados
    -------------------------------------------------------


import os
os.chdir('C:/Users/LorranyDASILVA/Documents/PFE/Code')

final_data = pd.merge(s_data, a_data, on='surface')
final_data.to_csv('Tableau_Assamblage.csv', index=False)
```

Listing C.3 – Code 3 - Assamblage

C.1.4 Code 4 - Uniformisation

```

# -*- coding: utf-8 -*-
"""
Created on Wed Aug 23 16:36:12 2023

@author: LorranyDASILVA
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data =
    pd.read_csv('C:/Users/LorranyDASILVA/Documents/PFE/Code/Tableau_Assamblage.csv',sep=","

# FIRST STEP: UNIFORMISATION

ECL_ZPP = 'Eclairage - Relamping (LED) + paramatrages ZPP'
ECL_ZPT = 'Eclairage - Relamping (LED) + paramatrages ZPT'
ECL_GDP = 'Eclairage - Installation equipement de gestion (gradation et/ou
    detection de presence)'
CTA_SUB = 'Ventilation - Remplacement ou installation CTA double flux avec
    echangeur de chaleur'
CTA_OPT = 'Ventilation - Optimisation CTA double flux avec echangeur de chaleur
    (temperature et/ou horaire)'
```

```
VEN_PDV = 'Ventilation - Calorifugeage et installation de variateurs de
    frequence sur pompes ou ventilateurs'
VEN_OPT = 'Ventilation - Optimisation des plages de fonctionnement '
ENV_IMI = 'Enveloppe - Renforcement de lisolation par linterieur'
ENV_IME = 'Enveloppe - Renforcement de lisolation par lexterieur'
ENV_IMIE = 'Enveloppe - Renforcement de lisolation par linterieur/exterieur'
ENV_IPB = 'Isolation - Isoler le plancher bas'
ENV_IPH = 'Isolation - Isoler la toiture terrasse/les greniers/combles perdus'
ENV_MEN = 'Enveloppe - Pose de menuiseries exterieures performantes en
    remplacement complet'
CHA_CCC = 'Chauffage - Remplacement le systeme par une chaudiere plus
    performante'
CHA_OCC = 'Chauffage - Optimisation des consignes de la chaudiere'
CHA_PAC = 'Chauffage - Remplacement le systeme par une pompe a chaleur'
CHA_OET = 'Chauffage - Optimisation des emetteurs terminaux (temperature et/ou
    horaire)'
CHA_RET = 'Chauffage - Remplacement des emetteurs terminaux'
GES_CTV = 'Gestion - Consignes de temperature vertueuses'
GES_ECL = 'Gestion - Reduction du fonctionnement de leclairage en inoccupation'
GES_BUR = 'Gestion - Reduction du fonctionnement de bureautique et reprographie
    en inoccupation'
GTB_SUB = 'GTB - Installation dune GTB'
PRO_PPV = 'Production - Production electricite photovoltaique'
PRO_ECS = 'Production - Production ECS solaire thermique'


# Define search conditions and replacement texts
search_and_replace = [
    # ECLAIRAGE
    (['LED', 'bureaux'], ECL_ZPP),
    (['Eclairage'], ECL_ZPP),
    (['ECL'], ECL_ZPP),
    (['Renovation','eclairage'], ECL_ZPP),
    (['Relampage','LED'], ECL_ZPP),
    (['Remplacement','eclairage'], ECL_ZPP),
    (['Remplacement','luminaires'], ECL_ZPP),
    (['LED','Remplacement','halogenes'], ECL_ZPP),
    (['luminaires','performants'], ECL_ZPP),
    (['eclairage','puissance'], ECL_GDP),
    (['LED','sanitaires'], ECL_ZPT ),
    (['LED','parkings'], ECL_ZPT ),
    (['LED','intermittante'], ECL_ZPT ),
    (['LED','circulation'], ECL_ZPT ),
    (['Optimisation','eclairage'], ECL_GDP),
    (['minuterie','eclairage'], ECL_GDP),
    (['programmation','eclairage'], ECL_GDP),
    (['horloge','eclairage'], ECL_GDP),
    (['eclairage','performant'], ECL_GDP),
    (['Detection'], ECL_GDP),
    (['Detecteurs'], ECL_GDP),
    (['Gradation'], ECL_GDP),
    (['gradation'], ECL_GDP),



    # CTA
    (['CTA','Remplacement'], CTA_SUB),
    (['CTA','Recuperateur'], CTA_SUB),
    (['CTA','recuperation'], CTA_SUB),
    (['CTA','Recuperation'], CTA_SUB),
    (['CTA','changement'], CTA_SUB),
    (['CTA','double','flux'], CTA_SUB),
    (['CTA','Retrofit'], CTA_SUB),
    (['CTA','Optimisation'], CTA_OPT),
    (['CTA','Regulation'], CTA_OPT),
    (['CTA','horloge'], CTA_OPT),
    (['CTA','horaire'], CTA_OPT),
    (['CTA','temperature'], CTA_OPT),
    (['CTA','fonctionnement'], CTA_OPT),
    (['CTA','programmation'], CTA_OPT),
    (['CTA','OPTIMISATION'], CTA_OPT),
    (['CTA','consignes'], CTA_OPT),
    (['CTA','vitesse'], CTA_OPT),
    (['CTA','Arret'], CTA_OPT),
    (['CTA','melange'], CTA_OPT),
    (['CTA','Horloge'], CTA_OPT),
    (['CTA','Sonde'], CTA_OPT),
    (['Asservissement'], CTA_OPT), #### NOT SURE

    # Pompes debit variable
    (['Distribution','variable'], VEN_PDV),
    (['distribution','variable'], VEN_PDV),
    (['Optimisation','ventilation'], VEN_OPT),
    (['Arret','pompes'], VEN_OPT),
    (['Arret','pompe'], VEN_OPT),
    (['Arret','ventilateurs'], VEN_OPT),        ##### ARRET DES POMPES

    # Isolation
    (['Isolation','murs','interieur'], ENV_IMIE),
    (['Isolation','interieur'], ENV_IMIE),
    (['Isolation','Murs'], ENV_IMIE),
    (['Isolation','des','murs'], ENV_IMIE),              #############
       ATTENTION HYPOTHESES QUE C'EST PAR L'INTERIEUR MAIS ON NE SAIT PAS
       REALLY
    (['Isolation','murs','exterieur'], ENV_IMIE),
    (['Isolation','plancher','haut'], ENV_IPB),
    (['Isolation','plancher','bas'], ENV_IPB),
    (['menuiseries','remplacement'], ENV_MEN),
    (['menuiseries','Pose'], ENV_MEN),

    #CHAUFFAGE

    (['Chaudiere','condensation'], CHA_CCC),
    (['regulation','chaudiere'], CHA_CCC),
    (['Chaudiere','Reduit'], CHA_OCC),
    (['Chaudiere','Pilotage'], CHA_OCC),
    (['optimisation','chaudiere'], CHA_OCC),
    (['PAC'], CHA_PAC),
    (['pompe','chaleur'], CHA_PAC),
    (['pompes','chaleur'], CHA_PAC),

    # Ventilo-convecteurs
    (['VCV','Horloges'], CHA_OET),
    (['VCV','Horaires'], CHA_OET),
    (['VCV','regulations'], CHA_OET),
    (['VCV','regualtion'], CHA_OET),
    (['VCV','thermostat'], CHA_OET),
    (['VCV','programmation'], CHA_OET),
    (['Gestion','emetteurs'], CHA_OET),
    (['horaire','emetteurs'], CHA_OET),
    (['Optimisation','emetteurs'], CHA_OET),
    (['Optimisation','ventilo-convecteurs'], CHA_OET),
    (['commutation'], CHA_OET),
    (['VCV','Optimisation'], CHA_OET),
    (['Robinets','thermostatiques'], CHA_OET),
    (['robinets','thermostatiques'], CHA_OET),
    (['VCV','HEE'], CHA_RET),
    (['VCV','Remplacement'], CHA_RET),
    (['boitiers','terminaux'], CHA_RET),
    (['emetteurs','Remplacement'], CHA_RET),


    # Production SOLAIRE
    (['photovoltaique'], PRO_PPV),
    (['ECS','solaire'], PRO_ECS),
    (['thermique','Solaire'], PRO_ECS),

    #GESTION
    (['Consignes','vertueuses'], GES_CTV),
    (['Consignes','moderees'], GES_CTV),
    (['Consignes','raisonnables'], GES_CTV),
    (['GTB'], GTB_SUB),
    (['Reduction','bureautique'], GES_BUR),
    (['veille','bureautique'], GES_BUR),
    (['Optimisation','equipements'], GES_BUR),
    (['ecleirage','innocupation'], GES_ECL),
    (['Suppression','veilles'], GES_BUR),
    (['Suppression','lampes'], GES_ECL),
    (['Gestion','eclairage'], GES_ECL),
    (['eclairage','gestion'], GES_ECL),
    (['interrupteurs','vertueuse'], GES_ECL),
    (['Reduction','fonctionnement','ecleirage'],GES_ECL),
    (['Arret','ecleirage'], GES_ECL),
    (['Usage','ecleirage'], GES_ECL)
]

# Create a dictionary to store conditions and their corresponding replacement
    texts
conditions_dict = {}

for i, (search_condition, replacement_text) in enumerate(search_and_replace):
    condition_check = data['libelle'].str.contains(search_condition[0])
    for term in search_condition[1:]:
        condition_check = condition_check & data['libelle'].str.contains(term)

    if replacement_text in conditions_dict:
        conditions_dict[replacement_text] = conditions_dict[replacement_text] |
            condition_check
    else:
        conditions_dict[replacement_text] = condition_check

```

```
# Combine columns with the same replacement text into a single column
for replacement_text, condition_check in conditions_dict.items():
    data[replacement_text] = condition_check

# Create a new column with the name of the column where there's a True statement
data['result'] = data.iloc[:, -len(conditions_dict):].idxmax(axis=1)

# Filter out columns with False values
data['result'] = data.apply(lambda row: row['result'] if row[row['result']]
    else None, axis=1)

# Drop the original 'action' columns
data = data.drop(columns=[col for col in data.columns if
    col.startswith('action_')])

def check_multiple_true(row):
    true_columns = [col for col in conditions_dict if row[col]]
    if len(true_columns) > 1:
        libelle_value = row['libelle']
        return f"The line {row.name+2}: '{libelle_value}' has more than one
            action"
    else:
        return None

error_messages = data.apply(check_multiple_true, axis=1)
for error_message in error_messages.dropna():
    print(error_message)

count_none = data['result'].isna().sum()
num_rows = len(data)
print("You have completed ",num_rows - count_none," actions! You need yet to
    fill",count_none, ".")

import os
os.chdir('C:/Users/LorranyDASILVA/Documents/PFE/Code')
data.to_csv('Tableau_ID.csv', index=False)
```

Listing C.4 – Code 4 - Uniformisation

C.1.5 Code 5 - Filtering and Correlation

```

# -*- coding: utf-8 -*-
"""
Created on Tue Oct  3 17:58:25 2023

@author: LorranyDASILVA
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv('C:/Users/LorranyDASILVA/Documents/PFE/Code/Tableau_ID.csv')
```

```

# SECOND PART: FILTERING THE MOST FREQUENT CASES
counts = data['result'].value_counts()
counts = counts.reset_index()
counts.columns = ['result', 'frequence']
#subset_counts = counts[counts['frequence'] > 0] #### FIRST ITERATION
subset_counts = counts[counts['frequence'] > 15]
qtty = subset_counts['frequence'].sum()
print('Actual quantity of cases after choosing the most frequent, above 15
    occurences:', qtty)
subset_data = data[data['result'].isin(subset_counts['result'])]


# Determine the columns to drop
subset_counts = subset_counts['result'].tolist()
columns_to_drop = subset_data.columns[subset_data.columns.get_loc('libelle2') +
    1:subset_data.columns.get_loc('result')]
columns_to_drop = [col for col in columns_to_drop if col not in subset_counts]
subset_data.drop(columns=columns_to_drop, inplace=True)


plt.figure(dpi=2000)
g0 = sns.catplot(y='result', kind='count', data=subset_data, height=5, aspect=2)
g0.set_yticklabels(rotation=0, fontsize=7)
g0.set_xticklabels(fontsize=7)
plt.xlabel('Count', fontsize=10)
plt.ylabel('Ameliorations', fontsize=10)
#plt.axvline(x=15, color='red', linestyle='--', label='Count 15')   #### FIRST
    ITERATION

plt.savefig('filtrage2.png', bbox_inches='tight')
# Show the plot
plt.show()

subset_data.fillna(0, inplace=True)
subset_data['result2'] = subset_data['pEFg_chauff']*subset_data['pEF_chauff'] +
    subset_data['pEFg_froid']*subset_data['pEF_froid'] +
    subset_data['pEFg_ecl']*subset_data['pEF_ecl'] +
    subset_data['pEFg_bureautique']*subset_data['pEF_bureautique'] +
    subset_data['pEFg_vent']*subset_data['pEF_vent']
subset_data = subset_data[subset_data['result2'] != 0]

# Tinha 472 melhorias depois da filtragem dos casos principais, excluindo-se 12
    que tem ganho igual a zero, tem-se 460.

subset_data.to_csv('Tableau_Filter.csv', index=False)

df = subset_data
ultima_coluna = df.columns[-1]

actions = df.columns[df.columns.get_loc('libelle2') +
    1:df.columns.get_loc('result')]
actions = actions.tolist()

qualitative =
    ['construction2','paroi2','isol_type','isol_epaiss','menui_type2','menui_vitrage2',
'menui_protect2','ph_type2','ph_isol_epaisseur','pb_type2','pb_isol_epaisseur','chauff',
'chauff_type2','refr','refr_type2','vent_principe2','vent_rendement','ecl_gestion2']

quantitative =
    ['surface','u_bat','compacite','deperd_total','taux_occ','menui_uw','menui_fs','ecl_pui
'pEF_chauff','pEF_froid','pEF_ecl','pEF_bureautique','pEF_vent','rinvest','pEFg_chauff',
'pEFg_froid','pEFg_ecl','pEFg_bureautique','pEFg_vent','result2']

for col in quantitative:  # Iterando sobre todas as colunas, exceto a  ltima
 plt.figure(figsize=(6, 4))
 scatter =
     plt.scatter(df[ultima_coluna],df[col],alpha=0.7,c=pd.factorize(df['result'])[0],cmap='
 classes = actions
 plt.xlabel(ultima_coluna)
 plt.ylabel(col)
 plt.title(f'Correla  o entre {col} e {ultima_coluna}')
 plt.legend(handles=scatter.legend_elements()[0], labels=classes, title='Type',
     loc='upper center', bbox_to_anchor=(0.6, -0.1),fontsize='small')
 plt.savefig(f'Scatterplot_quanti_{col}.png', bbox_inches='tight')
 plt.grid(True)
 plt.show()

### TESTES
data_quali = df[qualitative]
data_qtty = df[quantitative]
data_quali['result2'] = df['result2']

#sns.set_style('white')
#sns.set_palette("pastel")  # You can replace "pastel" with your desired palette
for col in data_quali :
    # Create a boxplot or violin plot
    plt.figure(figsize=(8, 6))  # Adjust the figure size as needed
    # Boxplot
    ax = sns.violinplot(x=data_quali[col], y=data_quali['result2'],
        data=data_quali, linewidth=0)
    sns.boxplot(x=data_quali[col], y=data_quali['result2'], data=data_quali,
        linewidth=0.5, width=0.4, boxprops={'fill': None,'zorder': 2},ax=ax)

    # Or, use a violin plot for a different view
    # sns.violinplot(x='Column_A', y='Column_B', data=data)
    plt.xlabel(f' Rapport entre {col} e {ultima_coluna} (Categorical)')
    plt.ylabel(f'{ultima_coluna}(Continuous)')
    plt.title(f'Rapport entre {col} e {ultima_coluna} (Categorical)')
    plt.savefig(f'Boxplot_quali_{col}.png', bbox_inches='tight')
    plt.show()


for col in data_qtty:
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
    # Plot histogram on the first subplot
    ax1.hist(data_qtty[col], color='pink', alpha=0.7)
    ax1.set_xlabel(col)
    ax1.set_title(f'Boxplot of {col}')
    ax1.grid(True)
    # Plot violinplot on the second subplot
    ax2.violinplot(data_qtty[col], vert=False)
    ax2.boxplot(data_qtty[col], vert=False)
    ax2.set_xlabel(col)
    ax2.set_title(f'Boxplot of {col}')
    plt.savefig(f'Boxplot_quanti_{col}.png', bbox_inches='tight')
    ax2.grid(True)

    plt.show()
```

Listing C.5 – Code 5 - Filtering and Correlation

C.1.6 Code 6 - Pre-metamodel

```

# -*- coding: utf-8 -*-
"""
Created on Tue Oct  3 17:51:30 2023

@author: LorranyDASILVA
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df =
    pd.read_csv('C:/Users/LorranyDASILVA/Documents/PFE/Code/Tableau_Filter.csv',sep=",")

replacement_dict = {False: 0, True: 1,
'Fin XIXeme':1875,'Millieu XX':1950,
'Fin XVIII' : 1775, 'Fin XIX' : 1875, 'Debut XX' : 1900, 'Fin XX' : 1975,
'Entre 2000 et 2010' : 2005, 'Apres 2010' : 2015}

df = df.replace(replacement_dict)

col_names = df.columns
df = df.drop(columns = ['libelle2','libelle','result'])

qualitative_cols =
    ['construction2','paroi2','isol_type','isol_epaiss','menui_type2','menui_vitrage2',
'menui_protect2','ph_type2','ph_isol_epaisseur','pb_type2','pb_isol_epaisseur','chauff',
'chauff_type2','refr','refr_type2','vent_principe2','vent_rendement','ecl_gestion2']

qualitative_data = df[qualitative_cols]
quantitative_data = df.drop(columns = qualitative_cols)
my_dict = {}

for col in qualitative_data:
      qd_cleaned = qualitative_data.dropna(subset=[col])
      my_dict[col] = np.unique(qd_cleaned[col])
      #print(unique_values)

# Create a new dictionary to store the modified values
modified_dict = {}
correspondence_dict = {}

# Iterate through the original dictionary
for key, values in my_dict.items():
    # Create a mapping dictionary to map unique values to numbers
    unique_values = np.unique(values)
    value_to_number = {value: idx for idx, value in enumerate(unique_values)}

    # Replace the array of values with an array of numbers
    modified_dict[key] = np.array([value_to_number[value] for value in values])

    # Create a correspondence dictionary
    correspondence_dict[key] = {value: idx for value, idx in
        value_to_number.items()}

# Print the modified dictionary
for key, values in modified_dict.items():
    print(f"{key}: array({values.tolist()})")

# Print the correspondence dictionary
for key, correspondence in correspondence_dict.items():
    print(f"Correspondence for {key}:")
    for original, new in correspondence.items():
        print(f"  {original}: {new}")

qualitative_data2 = qualitative_data.replace(correspondence_dict)
data_final = pd.concat([quantitative_data,qualitative_data2],axis=1)
data_final = data_final[[col for col in data_final.columns if col != 'result2']
    + ['result2']]

data_final.to_csv('Tableau_Metamodelo.csv', index=False)
```

Listing C.6 – Code 6 - Pre-metamodel

C.1.7 Code 7- Machine Learning Model

```

# -*- coding: utf-8 -*-
"""
Created on Tue Jun 25 07:52:29 2024

@author: lorra
"""

import pandas as pd
import matplotlib.pyplot as plt
import os
import numpy as np
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn import ensemble
from sklearn.inspection import permutation_importance
from sklearn.preprocessing import OneHotEncoder, StandardScaler
import joblib
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
import seaborn as sns


new_directory = "C:/Users/Dell/Documents/Trabalho"
os.chdir(new_directory)

```

```
df = pd.read_csv('C:/Users/Dell/Documents/Trabalho/PFE_Redo_JOB/Tableau_Metamodelo_Final_Man

max_values = pd.DataFrame({'column_names': df.columns,'real_value': df.max()}).reset_index(drop=True)
min_values = pd.DataFrame({'column_names': df.columns,'real_value': df.min()}).reset_index(drop=True)

max_values.to_csv('max_values.csv', sep= ';', index=False)
min_values.to_csv('min_values.csv', sep= ';', index=False)

plt.hist(df['result'], bins=10, edgecolor='black')  # Adjust number of bins as needed
plt.xlabel('Gain')
plt.ylabel('Frequency')
plt.title('Frequence of AAPEs gains distribution')
plt.grid(True)
plt.show()


#Define the clusters

clusters = {
    'envelope': ['construction2','paroi2','isol.type','isol.epaiss',
                 'ph.type2','ph.isol.epaisseur','pb.type2','pb.isol.epaisseur'],
    'light': ['ecl.puiss', 'pEF.ecl'],
    'chauff': ['chauff','chauff.type2'],
    'ref':['refr','refr.type2'],
    'vent':['vent.principe2','vent.rendement']
}

# Function to create clusters and add them as columns
def create_clusters(df, columns, n_clusters=15):
    # Select relevant columns
    data = df[columns]

    # One-hot encode categorical variables
    data_encoded = pd.get_dummies(data, drop_first=True)

    # Scale the data (optional but recommended for KMeans)
    scaler = StandardScaler()
    data_scaled = scaler.fit_transform(data_encoded)

    # Apply KMeans clustering
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(data_scaled)

    return cluster_labels

# Create and add clusters to the dataframe
for cluster_name, columns in clusters.items():
    df[f'{cluster_name}_cluster'] = create_clusters(df, columns)

# Display the dataframe with the new cluster columns
print(df.head())


```

```python
def find_optimal_clusters(data, max_k):
    iters = range(2, max_k+1)
    wcss = []
    silhouette_scores = []

    for k in iters:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data)
        wcss.append(kmeans.inertia_)
        silhouette_scores.append(silhouette_score(data, kmeans.labels_))

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

    # Elbow Method plot
    ax1.plot(iters, wcss, marker='o')
    ax1.set_xlabel('Number of Clusters')
    ax1.set_ylabel('WCSS')
    ax1.set_title('Elbow Method')

    # Silhouette score plot
    ax2.plot(iters, silhouette_scores, marker='o')
    ax2.set_xlabel('Number of Clusters')
    ax2.set_ylabel('Silhouette Score')
    ax2.set_title('Silhouette Scores' )

    plt.show()

# Use the function to find the optimal number of clusters for each group
for cluster_name, columns in clusters.items():
    data_encoded = pd.get_dummies(df[columns], drop_first=True)
    data_scaled = StandardScaler().fit_transform(data_encoded)
    print(f'Optimal clusters for {cluster_name}:')
    find_optimal_clusters(data_scaled, 15)



# Optimal number of clusters determined from the previous analysis
optimal_clusters = {
    'envelope': 8,
    'light': 5,
    'chauff': 5,
    'ref':4,
    'vent': 6
}


# Function to create and add clusters to the dataframe
def add_clusters_to_df(df, cluster_name, columns, n_clusters):
    # Select relevant columns
    data_encoded = pd.get_dummies(df[columns], drop_first=True)

    # Scale the data
    scaler = StandardScaler()
    data_scaled = scaler.fit_transform(data_encoded)

    # Apply KMeans clustering
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(data_scaled)

    # Add cluster labels as a new column
    df[f'{cluster_name}_cluster'] = cluster_labels

# Apply the function for each cluster group
for cluster_name, columns in clusters.items():
    n_clusters = optimal_clusters[cluster_name]
    add_clusters_to_df(df, cluster_name, columns, n_clusters)

# Display the dataframe with the new cluster columns
print(df.head())

# Define a custom color palette for the categories
palette = {
    '1': '#FFFFB3',  # Pastel yellow
    '2': '#FFFFB3',  # Pastel yellow
    '3': '#B3CDE3',  # Pastel blue
    '4': '#B3CDE3',  # Pastel blue
    '5': '#CBC0FF',  # Pastel violet
    '6': '#CBC0FF',  # Pastel violet
    '7': '#CBC0FF',  # Pastel violet
    '8': '#FFB6C1',  # Pastel red (light pink)
    '9': '#FFB6C1',  # Pastel red (light pink)
    '10': '#FFB6C1',  # Pastel red (light pink)
    '11': '#BCECAC',  # Pastel green
    '12': '#BCECAC'   # Pastel green
}


# List of new cluster columns
cluster_columns = ['envelope_cluster', 'light_cluster',
    'chauff_cluster','ref_cluster','vent_cluster']

# Reorder the columns to move the cluster columns to the beginning
df = df[cluster_columns + [col for col in df.columns if col not in
    cluster_columns]]

# Display the dataframe with the cluster columns at the beginning
print(df.head())

ultima_coluna = df.columns[-1]
column_names = df.columns

for index, row in df.iterrows():
    for column, value in row.items():
        if pd.isna(value):
            print(f"Row {index}, Column {column} has NA value: {value}")


#qualitative1 =  df.columns[df.columns.get_loc('percent') +
    1:df.columns.get_loc('result')]
qualitative1 = df.columns[-12:]
sum_values = df[qualitative1].sum()

plt.barh(sum_values.index, sum_values.values)
plt.xlabel('Frequence')
plt.ylabel('AAPE')
plt.title('AAPE Frequence')
plt.grid(True)
plt.show()


# Calculate row sums for specified columns
df['row_sum'] = df[qualitative1].sum(axis=1)

# Filter rows where the sum is less than or equal to 1
df = df[df['row_sum'] <= 1]

# Drop the 'row_sum' column if no longer needed
df = df.drop(columns='row_sum')

for i, col in enumerate(qualitative1):
    df[col] = df[col].apply(lambda x: i + 1 if x == 1 else 0)

# Calculate row sums for specified columns
df['aape'] = df[qualitative1].sum(axis=1)

cols = ['aape'] + [col for col in df.columns if col != 'aape']
df = df[cols]



# Create the boxplot with the custom palette
plt.figure(figsize=(12, 9))
sns.boxplot(x='aape', y='EFg', data=df, palette=palette)
plt.title('Boxplot of Results by Categories')
plt.xlabel('Category (aape)')
plt.ylabel('Result')
handles = [plt.Line2D([0], [0], color=palette[str(i + 1)], marker='o',
    linestyle='', label=qualitative1[i]) for i in range(len(qualitative1))]
plt.legend(handles=handles, title='Qualitative 1 Categories',
    bbox_to_anchor=(0.5, -0.15), loc='upper center', ncol=1)
plt.tight_layout()
plt.show()


# Function to remove outliers
def remove_outliers(df, group_col, value_col):
    # Calculate Q1 (25th percentile) and Q3 (75th percentile) for each group
    Q1 = df.groupby(group_col)[value_col].quantile(0.25)
    Q3 = df.groupby(group_col)[value_col].quantile(0.75)
    IQR = Q3 - Q1

    # Define bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Filter out the outliers
    df_filtered = df[
        (df[value_col] >= df[group_col].map(lower_bound)) &
        (df[value_col] <= df[group_col].map(upper_bound))
    ]

    return df_filtered

#Remove outliers
df = remove_outliers(df, 'aape', 'result')

plt.figure(figsize=(12, 9))
sns.boxplot(x='aape', y='result', data=df, palette=palette)
plt.title('Boxplot of Results by Categories')
plt.xlabel('Category (aape)')
plt.ylabel('Result')
handles = [plt.Line2D([0], [0], color=palette[str(i + 1)], marker='o',
    linestyle='', label=qualitative1[i]) for i in range(len(qualitative1))]
plt.legend(handles=handles, title='Qualitative 1 Categories',
    bbox_to_anchor=(0.5, -0.15), loc='upper center', ncol=1)
plt.tight_layout()
plt.show()

# Drop the 'row_sum' column if no longer needed
df = df.drop(columns=qualitative1)


qualitative2 =  ['construction2','paroi2','isol.type','isol.epaiss',
                'menui.type2','menui.vitrage2',
                'ph.type2','ph.isol.epaisseur','pb.type2','pb.isol.epaisseur',
                 'chauff','chauff.type2','refr','refr.type2','vent.principe2','vent.rendeme


quantitative = [#'surface',
                'u.bat',#'compacite','deperd.total',
                'taux.occ','menui.uw','menui.fs','ecl.puiss',#'rinvest',
                'EFg',
               'pEF.chauff','pEF.froid','pEF.ecl','pEF.bureautique','pEF.vent',
               'pEF.aux','pEF.serveur','pEF.divers','pEF.autreseq', 'pEF.ecs']


corr_matrix = pd.DataFrame(df[quantitative],columns = quantitative).corr()
sns.heatmap(corr_matrix,cmap = 'coolwarm')

corr_matrix2 = pd.DataFrame(df[qualitative2],columns = qualitative2).corr()
sns.heatmap(corr_matrix2,cmap = 'coolwarm')

corr_matrix3 = pd.DataFrame(df).corr()
sns.heatmap(corr_matrix3,cmap = 'coolwarm')



#clusters =
    ['envelope_cluster','light_cluster','chauff_cluster','ref_cluster','vent_cluster']

# o output do metamodelo sera o ganho energetico da acao
df[df.columns[-1]].head()

subset_features = quantitative + qualitative2 #+ clusters

# assim, separa-se as features e targets
features = df[subset_features]
target = df.copy()[df.columns[-1]]
```

```python

def z_score_normalize(series):
  mean = series.mean()
  std_dv = series.std()
  return series.apply(lambda x:(x - mean) / std_dv)
for col in quantitative:
  features[col] = z_score_normalize(features[col])

def min_max_normalize(series):
    min_val = series.min()
    max_val = series.max()
    range_val = max_val - min_val
    return series.apply(lambda x: (x - min_val) / range_val)

for col in quantitative:
    features[col] = min_max_normalize(features[col])

VAR = features
PRED = target


# Parameters for Gradient Boosting Regressor
params = {
    'max_depth': [5,15,40],
    'n_estimators': [100, 200, 500],
    'min_samples_split': [5, 10, 20,40],
    'learning_rate': [0.01, 0.05, 0.1],
    'loss': ['huber'],
    'random_state': [0]
}


# Example data
aape_values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]  # Example values

# quali1 =  ['pEF.ecl','ecl.puiss','pEF.vent','pEF.bureautique']
# quali2 =  ['pEF.ecl','construction2','pEF.froid','menui.fs','menui.vitrage2']
# quali3 =  ['pEF.chauff','ph.type2','pEF.vent','taux.occ','pEF.bureautique']
# quali4 =  ['pb.type2','chauff.type2','ecl.puiss','taux.occ','menui.vitrage2']
# quali5 =  ['isol.type','pEF.chauff','light_cluster','menui.uw','EF.total']
# quali6 =  ['envelope_cluster','taux.occ','pb.isol.epaisseur']
# quali7 =  ['ph.isol.epaisseur','menui.uw','ecl.puiss','taux.occ']
# quali8 =  ['pEF.chauff','menui.uw','menui.fs','taux.occ','construction2']
# quali9 =  ['pEF.bureautique', 'u.bat', 'pb.isol.epaisseur']
# quali10 = ['u.bat','pEF.ecl','pEF.vent','taux.occ','paroi2']
# quali11 =
    ['pEF.froid','light_cluster','ecl.puiss','paroi2','vent.principe2','isol.epaiss','menui
# quali12 = ['pEF.bureautique','u.bat','light_cluster','pEF.ecl']

quali1 =  ['pEF.ecl','ecl.puiss','pEF.vent','pEF.bureautique']
quali2 =  ['pEF.ecl','construction2','pEF.froid','menui.fs','menui.vitrage2']
quali3 =  ['pEF.chauff','ph.type2','pEF.vent','taux.occ','pEF.bureautique']
quali4 =  ['pb.type2','chauff.type2','ecl.puiss','taux.occ','menui.vitrage2']
quali5 =  ['isol.type','pEF.chauff','menui.uw','rEF.total']
quali6 =  ['taux.occ','pb.isol.epaisseur']
quali7 =  ['ph.isol.epaisseur','menui.uw','ecl.puiss','taux.occ']
quali8 =  ['pEF.chauff','menui.uw','menui.fs','taux.occ','construction2']
```

```
quali9 =  ['pEF.bureautique', 'u.bat', 'pb.isol.epaisseur','chauff']
quali10 = ['u.bat','pEF.ecl','pEF.vent','taux.occ','paroi2','chauff']
quali11 = ['pEF.froid','ecl.puiss','paroi2','vent.principe2','isol.epaiss','menui.uw']
quali12 = ['pEF.bureautique','u.bat','pEF.ecl']


# qualitative3 =  ['construction2','paroi2','isol.type','isol.epaiss',
#                  'menui.type2','menui.vitrage2',
#                  'ph.type2','ph.isol.epaisseur','pb.type2','pb.isol.epaisseur',
#    'chauff','chauff.type2','refr','refr.type2','vent.principe2','vent.rendement','ecl.gest


# # #### TESTANDO DEIXAR O MODELO LIVRE
# quali1 =  qualitative3 + quantitative
# quali2 =  qualitative3 + quantitative
# quali3 =  qualitative3 + quantitative
# quali4 =  qualitative3 + quantitative
# quali5 =  qualitative3 + quantitative
# quali6 =  qualitative3 + quantitative
# quali7 =  qualitative3 + quantitative
# quali8 =  qualitative3 + quantitative
# quali9 =  qualitative3 + quantitative
# quali10 = qualitative3 + quantitative
# quali11 = qualitative3 + quantitative
# quali12 = qualitative3 + quantitative


qualitative_list = [quali1,quali2,quali3,quali4,quali5,quali6,quali7,quali8,quali9,quali10,quali11,quali12

# Prepare the feature subsets corresponding to each target
target_to_qualitative_map = {
    1: quali1,
    2: quali2,
    3: quali3,
    4: quali4,
    5: quali5,
    6: quali6,
    7: quali7,
    8: quali8,
    9: quali9,
    10: quali10,
    11: quali11,
    12: quali12
    }

# LINHA DIRETO MODELO NORMAL

params = {
    'max_depth': [5,15,40],
    'n_estimators': [100, 200, 500],
    'min_samples_split': [5, 10, 20,40],
    'learning_rate': [0.01, 0.05, 0.1],
    'loss': ['huber'],
    'random_state': [0]
}


fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with Linear Regression
    Line', fontsize=16)

for idx, value in enumerate(aape_values):
    # Determinar a posi  o do subplot
    row, col = divmod(idx, 4)

    # Get the corresponding qualitative features for the current target
    parameters_aape = target_to_qualitative_map[value]

    # Criar subset de VAR e PRED baseado no valor de 'aape'
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]
    subset_pred = PRED[VAR['aape'] == value]

    # Dividir o subset em conjuntos de treinamento e teste
    subset_train, subset_test, subset_pred_train, subset_pred_test =
        train_test_split(
        subset, subset_pred, test_size=0.15, random_state=14
    )

    # Inicializar GradientBoostingRegressor e GridSearchCV
    reg = ensemble.GradientBoostingRegressor()
    grid_reg = GridSearchCV(estimator=reg, param_grid=params, cv=10,
        scoring='r2', n_jobs=12)
    grid_reg.fit(subset_train, subset_pred_train)

    # Obter o melhor modelo do GridSearchCV
    best_reg = grid_reg.best_estimator_

    # Previs es
    pred_train = best_reg.predict(subset_train)
    pred_test = best_reg.predict(subset_test)

    slope, intercept = np.polyfit(subset_pred_train, pred_train, 1)


    # Predict y values based on the regression line
    y_pred = slope * subset_pred_train + intercept


    # Adicionar linha com inclina  o 1
    max_val = max(max(subset_pred_train), max(subset_pred_test),
        max(pred_train), max(pred_test))
    axs[row, col].plot([0, max_val], [0, max_val], 'k--', label='Slope = 1')


    # TRANSFORMATION DE POINTS
    x_points = np.concatenate((subset_pred_train, subset_pred_test))
    y_points = np.concatenate((pred_train, pred_test))


    x =  x_points
    y =  y_points
    #y =  y_points + (1 - slope)*x_points - intercept

    # If you want to split the transformed points back into training and test sets
    x_trans_train = x[:len(subset_pred_train)]
    x_trans_test =  x[len(subset_pred_train):]

    # If you want to split the transformed points back into training and test sets
    y_trans_train = y[:len(pred_train)]
    y_trans_test =  y[len(pred_train):]

    # Scatterplot
    axs[row, col].scatter(x_trans_train, y_trans_train, color='blue', label='Training Data')
    axs[row, col].scatter(x_trans_test , y_trans_test , color='red', label='Test Data')

    # Calcular m tricas
    r2_train = r2_score(x_trans_train, y_trans_train)
    r2_test = r2_score(x_trans_test , y_trans_test )
    rmse_train = np.sqrt(mean_squared_error(x_trans_train, y_trans_train))
    rmse_test = np.sqrt(mean_squared_error(x_trans_test , y_trans_test ))
    mae_train = mean_absolute_error(x_trans_train, y_trans_train)
    mae_test = mean_absolute_error(x_trans_test , y_trans_test)

    # Adicionar anota  es de m tricas
    textstr = '\n'.join((
        f'R  (Train): {r2_train:.4f}',
        f'R  (Test): {r2_test:.4f}',
        f'RMSE (Train): {rmse_train:.4f}',
        f'RMSE (Test): {rmse_test:.4f}',
        f'MAE (Train): {mae_train:.4f}',
        f'MAE (Test): {mae_test:.4f}'
    ))
    axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes, fontsize=10,
                        verticalalignment='bottom', horizontalalignment='right',
                        bbox=dict(boxstyle='round,pad=0.5', edgecolor='black', facecolor='white'))

    axs[row, col].set_title(f'Scatter plot for aape={value}')
    axs[row, col].set_ylabel('Values')
    axs[row, col].legend()

plt.show()


########################## ANN

import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from scikeras.wrappers import KerasRegressor

# Define the model creation function for the ANN
def create_ann_model():
    model = Sequential()
    model.add(Dense(64, input_dim=subset.shape[1], activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='linear'))
    model.compile(optimizer='adam', loss='mean_squared_error',
       metrics=['mean_absolute_error'])
    return model

# Prepare the figure and subplots
fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with ANN Regression
    Line', fontsize=16)

for idx, value in enumerate(aape_values):
    # Determine the position of the subplot
    row, col = divmod(idx, 4)

    # Get the corresponding qualitative features for the current target
    parameters_aape = target_to_qualitative_map[value]

    # Create subset based on the value of 'aape'
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]
    subset_pred = PRED[VAR['aape'] == value]

    # Split the subset into training and testing sets
    subset_train, subset_test, subset_pred_train, subset_pred_test =
       train_test_split(
        subset, subset_pred, test_size=0.15, random_state=14
    )

    # Initialize the KerasRegressor with the ANN model
    ann_regressor = KerasRegressor(model=create_ann_model, epochs=50,
       batch_size=10, verbose=0)

    # Fit the ANN model on the training data
    ann_regressor.fit(subset_train, subset_pred_train)

    # Make predictions
    pred_train = ann_regressor.predict(subset_train)
    pred_test = ann_regressor.predict(subset_test)

    # Calculate the slope and intercept for the regression line
    slope, intercept = np.polyfit(subset_pred_train, pred_train, 1)

    # Predict y values based on the regression line
    y_pred = slope * subset_pred_train + intercept

    # Add a line with slope 1
    max_val = max(max(subset_pred_train), max(subset_pred_test),
       max(pred_train), max(pred_test))
    axs[row, col].plot([0, max_val], [0, max_val], 'k--', label='Slope = 1')

```

```python
    # Transformation of points
    x_points = np.concatenate((subset_pred_train, subset_pred_test))
    y_points = np.concatenate((pred_train, pred_test))

    x = x_points
    y = y_points

    # If you want to split the transformed points back into training and test sets
    x_trans_train = x[:len(subset_pred_train)]
    x_trans_test = x[len(subset_pred_train):]
    y_trans_train = y[:len(pred_train)]
    y_trans_test = y[len(pred_train):]

    # Scatterplot
    axs[row, col].scatter(x_trans_train, y_trans_train, color='blue', label='Training Data')
    axs[row, col].scatter(x_trans_test, y_trans_test, color='red', label='Test Data')

    # Calculate metrics
    r2_train = r2_score(x_trans_train, y_trans_train)
    r2_test = r2_score(x_trans_test, y_trans_test)
    rmse_train = np.sqrt(mean_squared_error(x_trans_train, y_trans_train))
    rmse_test = np.sqrt(mean_squared_error(x_trans_test, y_trans_test))
    mae_train = mean_absolute_error(x_trans_train, y_trans_train)
    mae_test = mean_absolute_error(x_trans_test, y_trans_test)

    # Add annotations of metrics
    textstr = '\n'.join((
        f'R   (Train): {r2_train:.4f}',
        f'R   (Test): {r2_test:.4f}',
        f'RMSE (Train): {rmse_train:.4f}',
        f'RMSE (Test): {rmse_test:.4f}',
        f'MAE (Train): {mae_train:.4f}',
        f'MAE (Test): {mae_test:.4f}'
    ))
    axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes, fontsize=10,
                        verticalalignment='bottom', horizontalalignment='right',
                        bbox=dict(boxstyle='round,pad=0.5', edgecolor='black', facecolor='white'))

    axs[row, col].set_title(f'Scatter plot for aape={value}')
    axs[row, col].set_ylabel('Values')
    axs[row, col].legend()

plt.show()



############################ DECISION TREE


import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.tree import DecisionTreeRegressor

# Parameters for the Decision Tree Regressor
params = {
    'max_depth': 5,             # Maximum depth of the tree
    'min_samples_split': 2,     # Minimum number of samples required to split
        an internal node
    'min_samples_leaf': 1,      # Minimum number of samples required to be at a
        leaf node
    'random_state': 14          # Seed for random number generation
}

# Prepare the figure and subplots
fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with Decision Tree
    Regression Line', fontsize=16)

for idx, value in enumerate(aape_values):
    # Determine the position of the subplot
    row, col = divmod(idx, 4)

    # Get the corresponding qualitative features for the current target
    parameters_aape = target_to_qualitative_map[value]

    # Create subset based on the value of 'aape'
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]
    subset_pred = PRED[VAR['aape'] == value]

    # Split the subset into training and testing sets
    subset_train, subset_test, subset_pred_train, subset_pred_test =
        train_test_split(
        subset, subset_pred, test_size=0.15, random_state=14
    )

    # Initialize the Decision Tree Regressor with specified parameters
    dt_regressor = DecisionTreeRegressor(**params)

    # Fit the Decision Tree model on the training data
    dt_regressor.fit(subset_train, subset_pred_train)

    # Make predictions
    pred_train = dt_regressor.predict(subset_train)
    pred_test = dt_regressor.predict(subset_test)

    # Calculate the slope and intercept for the regression line
    slope, intercept = np.polyfit(subset_pred_train, pred_train, 1)

    # Predict y values based on the regression line
    y_pred = slope * subset_pred_train + intercept

    # Add a line with slope 1
    max_val = max(max(subset_pred_train), max(subset_pred_test),
        max(pred_train), max(pred_test))
    axs[row, col].plot([0, max_val], [0, max_val], 'k--', label='Slope = 1')

    # Transformation of points
    x_points = np.concatenate((subset_pred_train, subset_pred_test))
    y_points = np.concatenate((pred_train, pred_test))

    x = x_points
    y = y_points

    # If you want to split the transformed points back into training and test sets
    x_trans_train = x[:len(subset_pred_train)]
    x_trans_test = x[len(subset_pred_train):]
    y_trans_train = y[:len(pred_train)]
    y_trans_test = y[len(pred_train):]

    # Scatterplot
    axs[row, col].scatter(x_trans_train, y_trans_train, color='blue', label='Training Data')
    axs[row, col].scatter(x_trans_test, y_trans_test, color='red', label='Test Data')

    # Calculate metrics
    r2_train = r2_score(x_trans_train, y_trans_train)
    r2_test = r2_score(x_trans_test, y_trans_test)
    rmse_train = np.sqrt(mean_squared_error(x_trans_train, y_trans_train))
    rmse_test = np.sqrt(mean_squared_error(x_trans_test, y_trans_test))
    mae_train = mean_absolute_error(x_trans_train, y_trans_train)
    mae_test = mean_absolute_error(x_trans_test, y_trans_test)

    # Add annotations of metrics
    textstr = '\n'.join((
        f'R  (Train): {r2_train:.4f}',
        f'R  (Test): {r2_test:.4f}',
        f'RMSE (Train): {rmse_train:.4f}',
        f'RMSE (Test): {rmse_test:.4f}',
        f'MAE (Train): {mae_train:.4f}',
        f'MAE (Test): {mae_test:.4f}'
    ))
    axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes, fontsize=10,
                        verticalalignment='bottom', horizontalalignment='right',
                        bbox=dict(boxstyle='round,pad=0.5', edgecolor='black', facecolor='white'))

    axs[row, col].set_title(f'Scatter plot for aape={value}')
    axs[row, col].set_ylabel('Values')
    axs[row, col].legend()

plt.show()


################################ DECISION TREE GRID SEARCH


import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.tree import DecisionTreeRegressor

# Define a range of hyperparameters for the Decision Tree Regressor with added
    regularization
param_grid = {
    'max_depth': [None, 3, 5, 10, 15],  # Limit the maximum depth of the tree
    'min_samples_split': [2, 5, 10, 20],    # Increase the minimum number of
        samples required to split an internal node
    'min_samples_leaf': [1, 2, 5, 10],       # Increase the minimum number of
        samples required to be at a leaf node
    'max_features': [None, 'sqrt', 'log2'],  # Number of features to consider
        when looking for the best split
    'max_leaf_nodes': [None, 10, 20, 30],    # Limit the number of leaf nodes
    'random_state': [14]                      # Seed for random number
        generation (can keep it fixed)
}

# Prepare the figure and subplots
fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with Decision Tree
    Regression Line', fontsize=16)

for idx, value in enumerate(aape_values):
    # Determine the position of the subplot
    row, col = divmod(idx, 4)

    # Get the corresponding qualitative features for the current target
    parameters_aape = target_to_qualitative_map[value]

    # Create subset based on the value of 'aape'
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]
    subset_pred = PRED[VAR['aape'] == value]

    # Split the subset into training and testing sets
    subset_train, subset_test, subset_pred_train, subset_pred_test =
        train_test_split(
        subset, subset_pred, test_size=0.15, random_state=14
    )

    # Initialize the Decision Tree Regressor
    dt_regressor = DecisionTreeRegressor(random_state=14)

    # Set up GridSearchCV to find the best parameters
    grid_search = GridSearchCV(estimator=dt_regressor, param_grid=param_grid,
                               cv=10, scoring='r2', n_jobs=-1, verbose=1)

    # Fit the model
    grid_search.fit(subset_train, subset_pred_train)

    # Get the best model from GridSearchCV
    best_regressor = grid_search.best_estimator_

    # Make predictions
    pred_train = best_regressor.predict(subset_train)
    pred_test = best_regressor.predict(subset_test)

```

```python
        # Calculate the slope and intercept for the regression line
        slope, intercept = np.polyfit(subset_pred_train, pred_train, 1)

        # Predict y values based on the regression line
        y_pred = slope * subset_pred_train + intercept

        # Add a line with slope 1
        max_val = max(max(subset_pred_train), max(subset_pred_test), max(pred_train), max(pred_test))
        axs[row, col].plot([0, max_val], [0, max_val], 'k--', label='Slope = 1')

        # Transformation of points
        x_points = np.concatenate((subset_pred_train, subset_pred_test))
        y_points = np.concatenate((pred_train, pred_test))

        x = x_points
        y = y_points

        # Split the transformed points back into training and test sets
        x_trans_train = x[:len(subset_pred_train)]
        x_trans_test = x[len(subset_pred_train):]
        y_trans_train = y[:len(pred_train)]
        y_trans_test = y[len(pred_train):]

        # Scatterplot
        axs[row, col].scatter(x_trans_train, y_trans_train, color='blue', label='Training Data')
        axs[row, col].scatter(x_trans_test, y_trans_test, color='red', label='Test Data')

        # Calculate metrics
        r2_train = r2_score(x_trans_train, y_trans_train)
        r2_test = r2_score(x_trans_test, y_trans_test)
        rmse_train = np.sqrt(mean_squared_error(x_trans_train, y_trans_train))
        rmse_test = np.sqrt(mean_squared_error(x_trans_test, y_trans_test))
        mae_train = mean_absolute_error(x_trans_train, y_trans_train)
        mae_test = mean_absolute_error(x_trans_test, y_trans_test)

        # Add annotations of metrics
        textstr = '\n'.join((
            f'R   (Train): {r2_train:.4f}',
            f'R   (Test): {r2_test:.4f}',
            f'RMSE (Train): {rmse_train:.4f}',
            f'RMSE (Test): {rmse_test:.4f}',
            f'MAE (Train): {mae_train:.4f}',
            f'MAE (Test): {mae_test:.4f}'
        ))
        axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes, fontsize=10,
                            verticalalignment='bottom', horizontalalignment='right',
                            bbox=dict(boxstyle='round,pad=0.5', edgecolor='black', facecolor='white'))

        axs[row, col].set_title(f'Scatter plot for aape={value}')
        axs[row, col].set_ylabel('Values')
        axs[row, col].legend()

```

```python
plt.show()

############################################ RANDOM FOREST


from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np
import matplotlib.pyplot as plt

# Define the parameter grid for Random Forest
params = {
    'n_estimators': [100, 200, 500],
    'max_depth': [5, 15, 40],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4],
    'random_state': [0]
}

fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with Linear Regression
    Line', fontsize=16)

for idx, value in enumerate(aape_values):
    # Determine the position of the subplot
    row, col = divmod(idx, 4)

    # Get the corresponding qualitative features for the current target
    parameters_aape = target_to_qualitative_map[value]

    # Create subset of VAR and PRED based on the value of 'aape'
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]
    subset_pred = PRED[VAR['aape'] == value]

    # Split the subset into training and testing sets
    subset_train, subset_test, subset_pred_train, subset_pred_test =
        train_test_split(
        subset, subset_pred, test_size=0.15, random_state=14
    )

    # Initialize RandomForestRegressor and GridSearchCV
    reg = RandomForestRegressor()
    grid_reg = GridSearchCV(estimator=reg, param_grid=params, cv=5,
        scoring='r2', n_jobs=-1)
    grid_reg.fit(subset_train, subset_pred_train)

    # Obtain the best model from GridSearchCV
    best_reg = grid_reg.best_estimator_

    # Make predictions
    pred_train = best_reg.predict(subset_train)
    pred_test = best_reg.predict(subset_test)

    # Calculate slope and intercept for the training data
    slope, intercept = np.polyfit(subset_pred_train, pred_train, 1)

    # Predict y values based on the regression line
    y_pred = slope * subset_pred_train + intercept

    # Add line with slope 1
    max_val = max(max(subset_pred_train), max(subset_pred_test),
        max(pred_train), max(pred_test))
    axs[row, col].plot([0, max_val], [0, max_val], 'k--', label='Slope = 1')

    # Prepare points for scatterplot
    x_points = np.concatenate((subset_pred_train, subset_pred_test))
    y_points = np.concatenate((pred_train, pred_test))

    # Scatterplot
    axs[row, col].scatter(x_points[:len(subset_pred_train)],
        y_points[:len(pred_train)], color='blue', label='Training Data')
    axs[row, col].scatter(x_points[len(subset_pred_train):],
        y_points[len(pred_train):], color='red', label='Test Data')

    # Calculate metrics
    r2_train = r2_score(subset_pred_train, pred_train)
    r2_test = r2_score(subset_pred_test, pred_test)
    rmse_train = np.sqrt(mean_squared_error(subset_pred_train, pred_train))
    rmse_test = np.sqrt(mean_squared_error(subset_pred_test, pred_test))
    mae_train = mean_absolute_error(subset_pred_train, pred_train)
    mae_test = mean_absolute_error(subset_pred_test, pred_test)

    # Add metrics annotations
    textstr = '\n'.join((f'R  (Train): {r2_train:.4f}',
                         f'R  (Test): {r2_test:.4f}',
                         f'RMSE (Train): {rmse_train:.4f}',
                         f'RMSE (Test): {rmse_test:.4f}',
                         f'MAE (Train): {mae_train:.4f}',
                         f'MAE (Test): {mae_test:.4f}'))

    axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes,
        fontsize=10,
                        verticalalignment='bottom', horizontalalignment='right',
                        bbox=dict(boxstyle='round,pad=0.5', edgecolor='black',
                            facecolor='white'))

    axs[row, col].set_title(f'Scatter plot for aape={value}')
    axs[row, col].set_ylabel('Values')
    axs[row, col].legend()

plt.show()



fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with Linear Regression
    Line', fontsize=16)

for idx, value in enumerate(aape_values):
    # Determinar a posi  o do subplot
    row, col = divmod(idx, 4)

    # Get the corresponding qualitative features for the current target
    parameters_aape = target_to_qualitative_map[value]

    # Criar subset de VAR e PRED baseado no valor de 'aape'
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]
    subset_pred = PRED[VAR['aape'] == value]

    # Dividir o subset em conjuntos de treinamento e teste
    subset_train, subset_test, subset_pred_train, subset_pred_test = train_test_split(
        subset, subset_pred, test_size=0.15, random_state=14
    )

    # Inicializar GradientBoostingRegressor e GridSearchCV
    reg = ensemble.GradientBoostingRegressor()
    grid_reg = GridSearchCV(estimator=reg, param_grid=params, cv=10, scoring='r2', n_jobs=12)
    grid_reg.fit(subset_train, subset_pred_train)

    # Obter o melhor modelo do GridSearchCV
    best_reg = grid_reg.best_estimator_

    # Previs es
    pred_train = best_reg.predict(subset_train)
    pred_test = best_reg.predict(subset_test)

    slope, intercept = np.polyfit(subset_pred_train, pred_train, 1)


    # Predict y values based on the regression line
    y_pred = slope * subset_pred_train + intercept


    # Adicionar linha com inclina  o 1
    max_val = max(max(subset_pred_train), max(subset_pred_test), max(pred_train), max(pred_test))
    axs[row, col].plot([0, max_val], [0, max_val], 'k--', label='Slope = 1')


    # TRANSFORMATION DE POINTS
    x_points = np.concatenate((subset_pred_train, subset_pred_test))
    y_points = np.concatenate((pred_train, pred_test))


    x =  x_points
    y =  y_points
    #y =  y_points + (1 - slope)*x_points - intercept

    # If you want to split the transformed points back into training and test sets
    x_trans_train = x[:len(subset_pred_train)]
    x_trans_test =  x[len(subset_pred_train):]

    # If you want to split the transformed points back into training and test sets
    y_trans_train = y[:len(pred_train)]
    y_trans_test =  y[len(pred_train):]

    # Scatterplot
    axs[row, col].scatter(x_trans_train, y_trans_train, color='blue',
       label='Training Data')
    axs[row, col].scatter(x_trans_test , y_trans_test , color='red',
       label='Test Data')

    # Calcular m tricas
    r2_train = r2_score(x_trans_train, y_trans_train)
    r2_test = r2_score(x_trans_test , y_trans_test )
    rmse_train = np.sqrt(mean_squared_error(x_trans_train, y_trans_train))
    rmse_test = np.sqrt(mean_squared_error(x_trans_test , y_trans_test ))
    mae_train = mean_absolute_error(x_trans_train, y_trans_train)
    mae_test = mean_absolute_error(x_trans_test , y_trans_test)

    # Adicionar anota  es de m tricas
    textstr = '\n'.join((
        f'R   (Train): {r2_train:.4f}',
        f'R   (Test): {r2_test:.4f}',
        f'RMSE (Train): {rmse_train:.4f}',
        f'RMSE (Test): {rmse_test:.4f}',
        f'MAE (Train): {mae_train:.4f}',
        f'MAE (Test): {mae_test:.4f}'
    ))
    axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes,
       fontsize=10,
                       verticalalignment='bottom', horizontalalignment='right',
                       bbox=dict(boxstyle='round,pad=0.5', edgecolor='black',
                          facecolor='white'))

    axs[row, col].set_title(f'Scatter plot for aape={value}')
    axs[row, col].set_ylabel('Values')
    axs[row, col].legend()

plt.show()




#Sup e que VAR e PRED s o seus DataFrames e aape_values   uma lista de
   categorias  nicas

fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with Linear Regression
   Line', fontsize=16)

for idx, value in enumerate(aape_values):
    # Determinar a posi  o do subplot
    row, col = divmod(idx, 4)

    # Get the corresponding qualitative features for the current target
    parameters_aape = target_to_qualitative_map[value]

    # Criar subset de VAR e PRED baseado no valor de 'aape'
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]
    subset_pred = PRED[VAR['aape'] == value]

    # Dividir o subset em conjuntos de treinamento e teste
    subset_train, subset_test, subset_pred_train, subset_pred_test =
        train_test_split(
        subset, subset_pred, test_size=0.15, random_state=14
    )

    # Inicializar GradientBoostingRegressor e GridSearchCV
    reg = ensemble.GradientBoostingRegressor()
    grid_reg = GridSearchCV(estimator=reg, param_grid=params, cv=10,
        scoring='r2', n_jobs=12)
    grid_reg.fit(subset_train, subset_pred_train)

    # Obter o melhor modelo do GridSearchCV
    best_reg = grid_reg.best_estimator_

    # Previs es
    pred_train = best_reg.predict(subset_train)
    pred_test = best_reg.predict(subset_test)

    slope, intercept = np.polyfit(subset_pred_train, pred_train, 1)


    # Predict y values based on the regression line
    y_pred = slope * subset_pred_train + intercept


    # Adicionar linha com inclina  o 1
    max_val = max(max(subset_pred_train), max(subset_pred_test),
        max(pred_train), max(pred_test))
    axs[row, col].plot([0, max_val], [0, max_val], 'k--', label='Slope = 1')


    # TRANSFORMATION DE POINTS
    x_points = np.concatenate((subset_pred_train, subset_pred_test))
    y_points = np.concatenate((pred_train, pred_test))


    x =  x_points
    y =  y_points + (1 - slope)*x_points - intercept

    # If you want to split the transformed points back into training and test
        sets
    x_trans_train = x[:len(subset_pred_train)]
    x_trans_test =  x[len(subset_pred_train):]

    # If you want to split the transformed points back into training and test
        sets
    y_trans_train = y[:len(pred_train)]
    y_trans_test =  y[len(pred_train):]

    # Scatterplot
    axs[row, col].scatter(x_trans_train, y_trans_train, color='blue',
        label='Training Data')
    axs[row, col].scatter(x_trans_test , y_trans_test , color='red',
        label='Test Data')

    # Calcular m tricas
    r2_train = r2_score(x_trans_train, y_trans_train)
    r2_test = r2_score(x_trans_test , y_trans_test )
    rmse_train = np.sqrt(mean_squared_error(x_trans_train, y_trans_train))
    rmse_test = np.sqrt(mean_squared_error(x_trans_test , y_trans_test ))
    mae_train = mean_absolute_error(x_trans_train, y_trans_train)
    mae_test = mean_absolute_error(x_trans_test , y_trans_test)

    # Adicionar anota  es de m tricas
    textstr = '\n'.join((
        f'R   (Train): {r2_train:.4f}',
        f'R   (Test): {r2_test:.4f}',
        f'RMSE (Train): {rmse_train:.4f}',
        f'RMSE (Test): {rmse_test:.4f}',
        f'MAE (Train): {mae_train:.4f}',
        f'MAE (Test): {mae_test:.4f}'
    ))
    axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes,
        fontsize=10,
                        verticalalignment='bottom', horizontalalignment='right',
                        bbox=dict(boxstyle='round,pad=0.5', edgecolor='black',
                            facecolor='white'))

    axs[row, col].set_title(f'Scatter plot for aape={value}')
    axs[row, col].set_ylabel('Values')
    axs[row, col].legend()

plt.show()


##########################################################################################

# Assume you have a list of feature names from the dataset
features = VAR.columns.drop('aape')  # Drop 'aape' as it's not a feature
param_grid = {}


import numpy as np
import matplotlib.pyplot as plt
from sklearn import ensemble
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import shap
from sklearn.model_selection import train_test_split


# Define the CombinedModel class
class CombinedModel:
    def __init__(self, param_grid):
        self.gbm_model = ensemble.GradientBoostingRegressor()
        self.param_grid = param_grid
        self.slope = None
        self.intercept = None
        self.best_model = None

    def fit(self, X, y):
```

```python
        grid_reg = GridSearchCV(estimator=self.gbm_model, param_grid=self.param_grid, cv=10, scoring='r2', n_jobs=12)
        grid_reg.fit(X, y)
        self.best_model = grid_reg.best_estimator_

        gbm_preds = self.best_model.predict(X)
        self.slope, self.intercept = np.polyfit(y, gbm_preds, 1)

    def predict(self, X, y):
        gbm_preds = self.best_model.predict(X)
        transformed_preds = gbm_preds + (1 - self.slope) * y - self.intercept
        return transformed_preds

# Example data and parameters
random_states = [14, 42, 68, 123, 155, 10, 2, 20, 200, 158]  # List of random states to try

fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with Linear Regression Line', fontsize=16)

for idx, value in enumerate(aape_values):
    row, col = divmod(idx, 4)

    parameters_aape = target_to_qualitative_map[value]
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]
    subset_pred = PRED[VAR['aape'] == value]

    best_r2_product = -np.inf  # Initialize the best product of R
    best_random_state = None   # Store the best random state
    best_pred_train, best_pred_test = None, None  # Best predictions

    for random_state in random_states:
        # Split the data with the current random state
        subset_train, subset_test, subset_pred_train, subset_pred_test = train_test_split(
            subset, subset_pred, test_size=0.15, random_state=random_state)

        # Create and fit the CombinedModel
        combined_model = CombinedModel(param_grid)
        combined_model.fit(subset_train, subset_pred_train)

        # Make predictions
        pred_train = combined_model.predict(subset_train, subset_pred_train)
        pred_test = combined_model.predict(subset_test, subset_pred_test)

        # Calculate R  for train and test
        r2_train = r2_score(subset_pred_train, pred_train)
        r2_test = r2_score(subset_pred_test, pred_test)

        # Calculate the product of R
        r2_product = r2_train * r2_test

        # Update the best random state if the current one is better
        if r2_product > best_r2_product:
            best_r2_product = r2_product
            best_random_state = random_state
            best_pred_train, best_pred_test = pred_train, pred_test
            best_subset_pred_train, best_subset_pred_test = subset_pred_train, subset_pred_test

    # Plotting for the best random state
    max_val = max(max(best_subset_pred_train), max(best_subset_pred_test), max(best_pred_train), max(best_pred_test))
    axs[row, col].plot([0, max_val], [0, max_val], 'k--', label=f'Best Random State = {best_random_state} (Slope = 1)')

    # Scatterplot for the best random state
    axs[row, col].scatter(best_subset_pred_train, best_pred_train, color='blue', label='Training Data')
    axs[row, col].scatter(best_subset_pred_test, best_pred_test, color='red', label='Test Data')

    # Metrics calculation for the best random state
    r2_train = r2_score(best_subset_pred_train, best_pred_train)
    r2_test = r2_score(best_subset_pred_test, best_pred_test)
    rmse_train = np.sqrt(mean_squared_error(best_subset_pred_train, best_pred_train))
    rmse_test = np.sqrt(mean_squared_error(best_subset_pred_test, best_pred_test))
    mae_train = mean_absolute_error(best_subset_pred_train, best_pred_train)
    mae_test = mean_absolute_error(best_subset_pred_test, best_pred_test)

    # Adding metric annotations
    textstr = '\n'.join((f'R  (Train): {r2_train:.4f}',
                          f'R  (Test): {r2_test:.4f}',
                          f'RMSE (Train): {rmse_train:.4f}',
                          f'RMSE (Test): {rmse_test:.4f}',
                          f'MAE (Train): {mae_train:.4f}',
                          f'MAE (Test): {mae_test:.4f}',
                          f'Best Random State: {best_random_state}'))
    axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes, fontsize=10,
                        verticalalignment='bottom', horizontalalignment='right',
                        bbox=dict(boxstyle='round,pad=0.5', edgecolor='black', facecolor='white'))

    axs[row, col].set_title(f'Scatter plot for aape={value}')
    axs[row, col].set_ylabel('Values')
    axs[row, col].legend()

plt.show()


# Define the CombinedModel class
class CombinedModel:
    def __init__(self, param_grid):
        self.gbm_model = ensemble.GradientBoostingRegressor()
        self.param_grid = param_grid
        self.slope = None
        self.intercept = None
        self.best_model = None

    def fit(self, X, y):
        grid_reg = GridSearchCV(estimator=self.gbm_model,
            param_grid=self.param_grid, cv=10, scoring='r2', n_jobs=12)
        grid_reg.fit(X, y)
        self.best_model = grid_reg.best_estimator_

        gbm_preds = self.best_model.predict(X)
        self.slope, self.intercept = np.polyfit(y, gbm_preds, 1)

    def predict(self, X, y):
        gbm_preds = self.best_model.predict(X)
        transformed_preds = gbm_preds + (1 - self.slope) * y - self.intercept
        return transformed_preds

# List of random states to try
random_states = [14, 42, 68, 123, 155, 10, 2, 20, 200, 158]

param_grid = {}

# Initialize figure for plotting
fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with Linear Regression
    Line', fontsize=16)

# Loop through each 'aape' value
for idx, value in enumerate(aape_values):
    row, col = divmod(idx, 4)

    parameters_aape = target_to_qualitative_map[value]
    subset_pred = PRED[VAR['aape'] == value]
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]


    best_r2_product = -np.inf  # Initialize the best product of R
    best_random_state = None   # Store the best random state
    best_pred_train, best_pred_test = None, None  # Best predictions

    # Try multiple random states and keep the best one
    for random_state in random_states:
        # Split the data with the current random state
        subset_train, subset_test, subset_pred_train, subset_pred_test =
            train_test_split(
             subset, subset_pred, test_size=0.15, random_state=random_state)

        # Create and fit the CombinedModel
        combined_model = CombinedModel(param_grid)
        combined_model.fit(subset_train, subset_pred_train)

        # Make predictions
        pred_train = combined_model.predict(subset_train, subset_pred_train)
        pred_test = combined_model.predict(subset_test, subset_pred_test)

        # Calculate R  for train and test
        r2_train = r2_score(subset_pred_train, pred_train)
        r2_test = r2_score(subset_pred_test, pred_test)

        # Calculate the product of  R
        r2_product = r2_train * r2_test

        # Update the best random state if the current one is better
        if r2_product > best_r2_product:
            best_r2_product = r2_product
            best_random_state = random_state
            best_pred_train, best_pred_test = pred_train, pred_test
            best_subset_pred_train, best_subset_pred_test = subset_pred_train, subset_pred_test

    # Save the best model for the current 'aape' value
    joblib.dump(combined_model.best_model, f'model_aape_{value}.joblib')

    # Plotting for the best random state
    max_val = max(max(best_subset_pred_train), max(best_subset_pred_test), max(best_pred_train), max(best_pred_test))
    min_val = min(min(best_subset_pred_train), min(best_subset_pred_test), min(best_pred_train), min(best_pred_test))
    axs[row, col].plot([min_val, max_val], [min_val, max_val], 'k--', label=f'Best Random State = {best_random_state} (Slope = 1)')

    # Scatterplot for the best random state
    axs[row, col].scatter(best_subset_pred_train, best_pred_train, color='blue', label='Training Data')
    axs[row, col].scatter(best_subset_pred_test, best_pred_test, color='red', label='Test Data')

    # Metrics calculation for the best random state
    r2_train = r2_score(best_subset_pred_train, best_pred_train)
    r2_test = r2_score(best_subset_pred_test, best_pred_test)
    rmse_train = np.sqrt(mean_squared_error(best_subset_pred_train, best_pred_train))
    rmse_test = np.sqrt(mean_squared_error(best_subset_pred_test, best_pred_test))
    mae_train = mean_absolute_error(best_subset_pred_train, best_pred_train)
    mae_test = mean_absolute_error(best_subset_pred_test, best_pred_test)

    # Adding metric annotations
    textstr = '\n'.join((f'R  (Train): {r2_train:.4f}',
                         f'R  (Test): {r2_test:.4f}',
                         f'RMSE (Train): {rmse_train:.4f}',
                         f'RMSE (Test): {rmse_test:.4f}',
                         f'MAE (Train): {mae_train:.4f}',
                         f'MAE (Test): {mae_test:.4f}',
                         f'Best Random State: {best_random_state}'))
    axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes, fontsize=10,
                       verticalalignment='bottom', horizontalalignment='right',
                       bbox=dict(boxstyle='round,pad=0.5', edgecolor='black', facecolor='white'))

    axs[row, col].set_title(f'Scatter plot for aape={value}')
    axs[row, col].set_ylabel('Values')
    axs[row, col].legend()


```

```

plt.show()


# List of random states
#random_states = [200,158,68,123,123,158,68,20,155,42,158,42]  # Ensure you
    have enough states for your AAPE values

# List of random states com CLUSTER
#random_states = [200,158,68,10,200,200,68,20,155,42,68,42]

#Nouveau modeles
random_states = [155,123,68,68,
                 200,158,158,42,
                 20,14,2,158]  # Ensure you have enough states for your AAPE
                     values


import shap

# Loop through each aape value
for idx, value in enumerate(aape_values):
    # Ensure we don't go out of bounds for the random_states list
    if idx >= len(random_states):
        break

    # Load the best model for the current aape value
    combined_model = joblib.load(f'best_combined_model_aape_{value}.joblib')

    parameters_aape = target_to_qualitative_map[value]
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]
    subset_pred = PRED[VAR['aape'] == value]

    # Train-test split using the corresponding random state
    subset_train, subset_test, subset_pred_train, subset_pred_test =
        train_test_split(
        subset, subset_pred, test_size=0.15, random_state=random_states[idx]
    )

    # Use SHAP TreeExplainer for the combined model's GBM
    explainer = shap.Explainer(combined_model)

    # Calculate SHAP values for the training subset
    shap_values_combined = explainer(subset_train)

    # Create a new figure for the SHAP summary plot
    plt.figure(figsize=(10, 6))
    plt.title(f'SHAP Summary for aape={value}')
    shap.summary_plot(shap_values_combined, subset_train, show=True)
    plt.show()






```

```python









########################33 random forest combined model


from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import numpy as np
import joblib
import matplotlib.pyplot as plt


class CombinedModel:
    def __init__(self, param_grid):
        self.rf_model = RandomForestRegressor()  # Using RandomForestRegressor
        self.param_grid = param_grid
        self.best_model = None

    def fit(self, X, y):
        grid_reg = GridSearchCV(estimator=self.rf_model,
            param_grid=self.param_grid, cv=10, scoring='r2', n_jobs=12)
        grid_reg.fit(X, y)
        self.best_model = grid_reg.best_estimator_

        rdf_preds = self.best_model.predict(X)
        self.slope, self.intercept = np.polyfit(y, rdf_preds, 1)

    def predict(self, X):
        rdf_preds = self.best_model.predict(X)
        transformed_preds = rdf_preds + (1 - self.slope) * y - self.intercept
        return transformed_preds

# List of random states to try
random_states = [14, 42, 68, 123, 155, 10, 2, 20, 200, 158]

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize figure for plotting
fig, axs = plt.subplots(3, 4, figsize=(20, 15), constrained_layout=True)
fig.suptitle('Scatterplot of Predictions and True Values with Linear Regression
    Line', fontsize=16)

# Loop through each 'aape' value
for idx, value in enumerate(aape_values):
    row, col = divmod(idx, 4)

    parameters_aape = target_to_qualitative_map[value]
    subset_pred = PRED[VAR['aape'] == value]
    subset = VAR[VAR['aape'] == value].drop(columns=['aape'])
    subset = subset[parameters_aape]

    best_r2_product = -np.inf  # Initialize the best product of R
    best_random_state = None   # Store the best random state
    best_pred_train, best_pred_test = None, None  # Best predictions

    # Try multiple random states and keep the best one
    for random_state in random_states:
        # Split the data with the current random state
        subset_train, subset_test, subset_pred_train, subset_pred_test = train_test_split(
            subset, subset_pred, test_size=0.15, random_state=random_state)

        # Create and fit the CombinedModel
        combined_model = CombinedModel(param_grid)
        combined_model.fit(subset_train, subset_pred_train)

        # Make predictions
        pred_train = combined_model.predict(subset_train)
        pred_test = combined_model.predict(subset_test)

        # Calculate R  for train and test
        r2_train = r2_score(subset_pred_train, pred_train)
        r2_test = r2_score(subset_pred_test, pred_test)

        # Calculate the product of R
        r2_product = r2_train * r2_test

        # Update the best random state if the current one is better
        if r2_product > best_r2_product:
            best_r2_product = r2_product
            best_random_state = random_state
            best_pred_train, best_pred_test = pred_train, pred_test
            best_subset_pred_train, best_subset_pred_test = subset_pred_train, subset_pred_test

    # Save the best model for the current 'aape' value
    joblib.dump(combined_model.best_model, f'model_aape_{value}.joblib')

    # Plotting for the best random state
    max_val = max(max(best_subset_pred_train), max(best_subset_pred_test), max(best_pred_train), max(best_pred_test))
    axs[row, col].plot([0, max_val], [0, max_val], 'k--', label=f'Best Random State = {best_random_state} (Slope = 1)')

    # Scatterplot for the best random state
    axs[row, col].scatter(best_subset_pred_train, best_pred_train, color='blue', label='Training Data')
    axs[row, col].scatter(best_subset_pred_test, best_pred_test, color='red', label='Test Data')

    # Metrics calculation for the best random state
    r2_train = r2_score(best_subset_pred_train, best_pred_train)
    r2_test = r2_score(best_subset_pred_test, best_pred_test)
    rmse_train = np.sqrt(mean_squared_error(best_subset_pred_train,
        best_pred_train))
    rmse_test = np.sqrt(mean_squared_error(best_subset_pred_test,
        best_pred_test))
    mae_train = mean_absolute_error(best_subset_pred_train, best_pred_train)
    mae_test = mean_absolute_error(best_subset_pred_test, best_pred_test)

    # Adding metric annotations
    textstr = '\n'.join((f' R   (Train): {r2_train:.4f}',
                         f' R   (Test): {r2_test:.4f}',
                         f'RMSE (Train): {rmse_train:.4f}',
                         f'RMSE (Test): {rmse_test:.4f}',
                         f'MAE (Train): {mae_train:.4f}',
                         f'MAE (Test): {mae_test:.4f}',
                         f'Best Random State: {best_random_state}'))
    axs[row, col].text(0.95, 0.05, textstr, transform=axs[row, col].transAxes,
        fontsize=10,
                       verticalalignment='bottom', horizontalalignment='right',
                       bbox=dict(boxstyle='round,pad=0.5', edgecolor='black',
                           facecolor='white'))

    axs[row, col].set_title(f'Scatter plot for aape={value}')
    axs[row, col].set_ylabel('Values')
    axs[row, col].legend()

plt.show()
```

Listing C.7 – Code 7- Metamodel

C.1.8 Code 8- Interface

```


# -*- coding: utf-8 -*-
"""
Created on Thu Dec 21 18:23:12 2023

@author: lorra
"""

import numpy as np
import pandas as pd
import requests
from io import StringIO, BytesIO
import io
import joblib
import streamlit as st
import matplotlib.pyplot as plt
import seaborn as sns
import uuid
#import hmac

# Define parameters for a request
```

```python
token = "ghp_ekALmcDhYvZgPyvm04x2twsHwm2mmu1rjXqG"
owner = 'lorranymendes'
repo = 'advenio-interface'
repo_model = 'advenio-interface'
path_csv1 = 'data_treated.csv'
path_max_values = 'max_values.csv'
path_min_values = 'min_values.csv'
path_model1 = 'model_aape_1.joblib'
path_model2 = 'model_aape_2.joblib'
path_model3 = 'model_aape_3.joblib'
path_model4 = 'model_aape_4.joblib'
path_model5 = 'model_aape_5.joblib'
path_model6 = 'model_aape_6.joblib'
path_model7 = 'model_aape_7.joblib'
path_model8 = 'model_aape_8.joblib'
path_model9 = 'model_aape_9.joblib'
path_model10 = 'model_aape_10.joblib'
path_model11 = 'model_aape_11.joblib'
path_model12 = 'model_aape_12.joblib'



f_decompo = ['EF Chauffage (%)','EF Refroidissement (%)','EF   clairage   (%)',
             'EF Bureautique (%)','EF Ventilation (%)', 'EF Auxiliaires (%)',
             'EF Autres   quipements   (%)','EF Serveurs (%)','EF ECS (%)']


path_decompo1 = 'decompo_pEF.chauff.joblib'
path_decompo2 = 'decompo_pEF.froid.joblib'
path_decompo3 = 'decompo_pEF.ecl.joblib'
path_decompo4 = 'decompo_pEF.bureautique.joblib'
path_decompo5 = 'decompo_pEF.vent.joblib'
path_decompo6 = 'decompo_pEF.aux.joblib'
path_decompo7 = 'decompo_pEF.autreseq.joblib'
path_decompo8 = 'decompo_pEF.serveur.joblib'
path_decompo9 = 'decompo_pEF.ecs.joblib'




path_python = 'dictionaries_text.py'
path_image = 'advenio_act_cover.jpg'

headers = {
    'accept': 'application/vnd.github.v3.raw',
    'authorization': f'token {token}'
}

# This should be the first Streamlit function in the script
st.set_page_config(layout="wide")

@st.cache_resource
def load_github_csv(owner, repo, path_csv, token):

    url_csv = f'https://api.github.com/repos/{owner}/{repo}/contents/{path_csv}'
    response = requests.get(url_csv, headers=headers)

    try:
        df = pd.read_csv(StringIO(response.text), sep=";")
        return df
    except pd.errors.EmptyDataError:
        print("The CSV response did not contain any data.")
    except pd.errors.ParserError as e:
        print("Error parsing CSV:", e)

    return None

@st.cache_resource
def load_github_model(owner, repo, path_model, token):

    url_model =
       f'https://api.github.com/repos/{owner}/{repo}/contents/{path_model}'

    try:
        response_model = requests.get(url_model, headers=headers)
        response_model.raise_for_status()  # Check if the request was successful

        model_bytes = BytesIO(response_model.content)
        model = joblib.load(model_bytes)
        print("Model loaded successfully.")
        return model
    except Exception as e:
        print("Error loading the model:", e)
        return None

def load_github_python_file(owner, repo, path_python, token):
    url_python =
       f'https://api.github.com/repos/{owner}/{repo}/contents/{path_python}'

    try:
        response_python = requests.get(url_python, headers=headers)

        # Execute the Python code from the file
        python_code = response_python.text
        exec(python_code, globals())

    except requests.exceptions.RequestException as e:
        print(f"Error loading the Python file: {e}")
    except Exception as e:
        print(f"Error executing the Python file: {e}")

    return None

@st.cache_resource
def load_github_image(owner, repo, path_image, token):

    url_image =
       f'https://raw.githubusercontent.com/{owner}/{repo}/main/{path_image}'
    response = requests.get(url_image, headers=headers)

    if response.status_code == 200:
        # Exibe a imagem como cabe alho
        st.image(response.content , use_column_width=True)
    else:
        st.error("Falha ao carregar a imagem.")

data = load_github_csv(owner, repo, path_csv1, token)



model1 = load_github_model(owner, repo, path_model1, token)
model2 = load_github_model(owner, repo, path_model2, token)
model3 = load_github_model(owner, repo, path_model3, token)
model4 = load_github_model(owner, repo, path_model4, token)
model5 = load_github_model(owner, repo, path_model5, token)
model6 = load_github_model(owner, repo, path_model6, token)
model7 = load_github_model(owner, repo, path_model7, token)
model8 = load_github_model(owner, repo, path_model8, token)
model9 = load_github_model(owner, repo, path_model9, token)
model10 = load_github_model(owner, repo, path_model10, token)
model11 = load_github_model(owner, repo, path_model11, token)
model12 = load_github_model(owner, repo, path_model12, token)


decompo1 = load_github_model(owner, repo, path_decompo1, token)
decompo2 = load_github_model(owner, repo, path_decompo2, token)
decompo3 = load_github_model(owner, repo, path_decompo3, token)
decompo4 = load_github_model(owner, repo, path_decompo4, token)
decompo5 = load_github_model(owner, repo, path_decompo5, token)
decompo6 = load_github_model(owner, repo, path_decompo6, token)
decompo7 = load_github_model(owner, repo, path_decompo7, token)
decompo8 = load_github_model(owner, repo, path_decompo8, token)
decompo9 = load_github_model(owner, repo, path_decompo9, token)




# def check_password():
#     """Returns `True` if the user had the correct password."""

#     def password_entered():
#         """Checks whether a password entered by the user is correct."""
#         if hmac.compare_digest(st.session_state["password"],
    st.secrets["password"]):
#             st.session_state["password_correct"] = True
#             del st.session_state["password"]  # Don't store the password.
#         else:
#             st.session_state["password_correct"] = False

#     # Return True if the passward is validated.
#     if st.session_state.get("password_correct", False):
#         return True

#     #st.image(image_path, use_column_width=True)

#     # Show input for password.
#     st.text_input(
#         "Mot de passe", type="password", on_change=password_entered,
    key="password"
#     )
#     if "password_correct" in st.session_state:
```

```
#         st.error("Mot de passe incorrect")
#     return False

# if not check_password():
#     st.stop()  # Do not continue if check_password is not True.


st.write('<style>div.block-container{padding-top:0rem;}</style>',
    unsafe_allow_html=True)
load_github_image(owner, repo, path_image, token)

def find_key_by_value():
    return None

def find_key_by_value_and_class():
    return None

def get_values_by_category():
    return None

def access_list_by_name():
    return None

def find_corresponding_name():
    return None

def min_max_normalize():
    return None

f_gen = []
f_pEF = []
f_types = []

column_mapping = {}
correspondence_column_dict = {}
correspondence_dict = {}
list_names = []
list_groups = {}
correspondence_quanti = {}
correspondence_column_dict = {}
plots_quanti = {}
plots_quali = {}


qualitative=
    ['construction2','paroi2','isol.type','isol.epaiss','menui.type2','menui.vitrage2',
                'ph.type2','ph.isol.epaisseur','pb.type2','pb.isol.epaisseur',
                 'chauff','chauff.type2','refr','refr.type2','vent.principe2','vent.rendeme


qualitative_graph=
    ['construction2','paroi2','isol.type','isol.epaiss','menui.type2','menui.vitrage2',
                'ph.type2','ph.isol.epaisseur','pb.type2','pb.isol.epaisseur',
                 'chauff','chauff.type2','refr','refr.type2','vent.principe2','vent.rendeme
                     'EF.total']

```

```
quantitative = ['taux.occ','menui.uw','menui.fs','ecl.puiss',
                'EF.total','pEF.chauff','pEF.froid','pEF.ecl','pEF.bureautique','pEF.vent']




aape_values = []

quali1 =  []
quali2 =  []
quali3 =  []
quali4 =  []
quali5 =  []
quali6 =  []
quali7 =  []
quali8 =  []
quali9 =  []
quali10 = []
quali11 = []
quali12 = []

qualitative_list = []

# Prepare the feature subsets corresponding to each target
target_to_qualitative_map = {}
target_to_decompo_map = {}

decompo_dict = {}
f_decompo = []

dictionaries = load_github_python_file(owner, repo, path_python, token)
f_quanti = f_gen + f_pEF

max_values = load_github_csv(owner, repo, path_max_values, token)
min_values = load_github_csv(owner, repo, path_min_values, token)


quanti_col_max = max_values[max_values['column_names'].isin(f_quanti)]
quanti_col_min = min_values[min_values['column_names'].isin(f_quanti)]

quanti_col_max.loc[:, 'real_value'] = quanti_col_max['real_value'].astype(float)
quanti_col_min.loc[:, 'real_value'] = quanti_col_min['real_value'].astype(float)


###############################################################################################

ultima_coluna = data.columns[-1]

# Subset the DataFrame based on the keys
data_quali = data[qualitative_graph]
data_qtty = data[quantitative]




def find_key(dictionary, value):
    for key, val in dictionary.items():
        if val == value:
            return key
    return None

##########################################################################################


def main():
    st.title("ARCS - Acc s Rapide Central Sevaia")
    #st.text("Tableau de Bord des Solutions Durables Advenio")

    # Add navigation to sidebar
    page = st.sidebar.selectbox("Menu", ["  propos","M tamod le de
        prediction", "Analyse des bases de donn es","Analyse du M tamod le"])

    if page == "  propos":
        #st.header("  propos")

        st.write("""
                 ### Introduction
                 Bienvenue sur notre Tableau de Bord des Solutions Durables
                    Sevaia! Ici, vous trouverez des tableaux de bord
                    interactifs et des visualisations pour explorer et
                    analyser diff rents ensembles de donn es.

                ###   Propos
                Notre application vise   fournir des informations pr cieuses
                   et   rendre l'exploration des donn es facile et
                   intuitive. Nous avons s lectionn  une s rie d'outils et
                   de visualisations pour vous aider   approfondir votre
                   compr hension des donn es.

                ### Comment Utiliser
                Pour commencer, s lectionnez un ensemble de donn es dans la
                   barre lat rale et explorez les visualisations disponibles.
                   Utilisez les commandes et les filtres pour personnaliser
                   votre analyse et d couvrir des tendances et des mod les
                   int ressants.
                Vous avez aussi acc s au m tamod le de prediction cr e
                   partir d'un entrainement d'une IA.
                """)

                # Features
        st.write("""
                 ### Fonctionnalit s Cl s
                 - Visualisations interactives
                 - Graphiques et diagrammes personnalisables
                 - Interface facile   utiliser
                 - Prise en charge de divers ensembles de donn es
                 - Mises   jour en temps r el
                 - Metamod le de prediction

                ### Technologies Utilis es
                - Python
                - Streamlit
                - Pandas
                - Matplotlib
                - Plotly
                - Gradient Boosting Machine

                """)

        # Contact Information
        st.write("""
                 ### Contactez-Nous
                 Si vous avez des questions ou des commentaires, n'h sitez pas nous contacter :
                     - Email: l.dasilva@sevaia.ue
                     - Ou envoyez un message Teams.

                """)

        # Footer
        st.write("""
                 ---
                 R alis  par [Lorrany Mendes](https://github.com/lorranymendes)
                 """)


        # Funcao que seleciona a variavel generica

    elif page == "M tamod le de prediction":

        st.header("Pr diction de la d composition:")
        dec_values_first = pd.Series()

        # Fun  o para selecionar as AAPEs e criar available_columns internamente
        def create_decompo_variable_selector(f_decompo, target_to_decompo_map):
            selected_decompo = st.multiselect("D compos  g ner  s:", f_decompo)  # Multiselect para sele  o
            selected_decompo_indices = [f_decompo.index(col) + 1 for col in selected_decompo]  #  ndices  incrementados

            # Processo interno para criar available_columns
            available_decompo_columns = set()
            for index in selected_decompo_indices:
                if index in target_to_decompo_map:
                    available_decompo_columns.update(target_to_decompo_map[index])  # Adiciona colunas sem duplicatas

            return selected_decompo_indices, available_decompo_columns

        # Pergunta: "Voc  quer gerar as decomposi  es?"
        response1 = st.radio("Voulez-vous g nerer les d compositions?", ("Oui", "Non"), index=1)

        # Se a resposta for "Oui", executa a fun  o
        if response1 == "Oui":
            #st.subheader("Prediction de la d composition:")
            selected_decompo_indices, available_decompo_columns = create_decompo_variable_selector(f_decompo, target_to_decompo_map)

            # Filter classes_selection based on available_columns
            classes_decompo_selection = [value for key, value in correspondence_column_dict.items() if key in available_decompo_columns]


            def create_d_qualitative_variable_selector(classe_d_list):
                variable_decompo_values = {}

                for i, classe_d in enumerate(classe_d_list):  # Use enumerate to get an index
                    # Part 1 - List
                    category_d_name_input = classe_d
                    values_d_list = get_values_by_category(correspondence_dict, category_d_name_input)

                    # Use a unique key for each selectbox
                    selected_decompo_variable = st.selectbox(f"{classe_d}:", values_d_list, key=f"{classe_d}_{i}")

                    # Update dictionary with selected variable as the key and its category as the value
                    source_d_category = find_key_by_value(correspondence_column_dict, category_d_name_input)
                    source_d_value = find_key_by_value_and_class(selected_decompo_variable, source_d_category)
                    variable_decompo_values[source_d_category] = source_d_value

                return pd.Series(variable_decompo_values, name='qualitative_values')


            # Assuming list_groups is defined somewhere in your code
            selected_decompo_qualitative = create_d_qualitative_variable_selector(classes_decompo_selection)




            def create_d_numeric_variables(columns):
                numeric_decompo_values = {}

                for i, column in enumerate(columns):
                    max_d_value = float(quanti_col_max.loc[quanti_col_max['column_names'] == column, 'real_value'].iloc[0])
                    min_d_value = float(quanti_col_min.loc[quanti_col_min['column_names'] == column, 'real_value'].iloc[0])

                    # Definindo o valor padr o
                    default_d_value = (max_d_value - min_d_value) / 2
                    name_d = find_corresponding_name(column, correspondence_quanti)
```

```

        # Manter a mesma chave por coluna
        key = f"decompo_input_{column}_{i}"

        # Criar o n mero de entrada
        value_d = st.number_input(
            "{} avec valeur minimale de {:.2f} et maximale de {:.2f}: ".format(name_d, min_d_value, max_d_value),
            value=default_d_value,
            key=key
        )

        # Verificar se o valor est  dentro dos limites
        if (min_d_value is None or value_d >= min_d_value) and (max_d_value is None or value_d <= max_d_value):
            numeric_decompo_values[column] = value_d
        else:
            # Exibir avisos se o valor estiver fora dos limites
            if min_d_value is not None and max_d_value is not None:
                st.warning("La valeur doit  tre  entre {} et {} pour la variable {}. S'il vous pla t, inserez une nouvelle valeur.".format(min_d_value, max_d_value, name_d))
            elif min_d_value is not None:
                st.warning("La valeur doit  tre  sup rieure   {} pour la variable {}. S'il vous pla t, inserez une nouvelle valeur.".format(min_d_value, name_d))
            elif max_d_value is not None:
                st.warning("La valeur doit  tre  inf rieure   {} pour la variable {}. S'il vous pla t, inserez une nouvelle valeur.".format(max_d_value, name_d))

    return pd.Series(numeric_decompo_values, name='decompo_values_d')




# Filter f_gen based on available_columns
column_decompo_list = [column for column in f_gen if column in available_decompo_columns]


selected_decompo_quantitative = create_d_numeric_variables(column_decompo_list)

###############################################################################


# Streamlit UI for user input
st.subheader("R sultat de la d composition:")

# Initialize an empty DataFrame to store predictions
df_predictions = pd.DataFrame(columns=['decompo_name', 'predictions_d'])
```

```

        try:
            # Concatenando as sele  es quantitativas e qualitativas
            result_d = pd.concat([selected_decompo_quantitative, selected_decompo_qualitative], axis=0)
            result_d = result_d.to_frame(name='Nome da Coluna')

            # Filtrando linhas indesejadas
            result_d = result_d[result_d['Nome da Coluna'] != 'Nome da Coluna']
            result2_d = result_d.T

            # Iterando sobre cada  ndice  selecionado
            for idx in selected_decompo_indices:
                decompo_name = f_decompo[idx - 1]
                parameters_decompo = target_to_decompo_map[idx]

                # Filtrando o input com base nos par metros
                filtered_d_input = result2_d[parameters_decompo]

                model_d_name = f'decompo{idx}'  # Assumindo que os modelos s o nomeados decompo1, decompo2, etc.
                expected_d_columns = globals()[model_d_name].feature_names_in_
                available_d_columns = filtered_d_input.columns.intersection(expected_d_columns)

                if len(available_d_columns) < len(expected_d_columns):
                     st.write(f"Variables qui manquent: {model_d_name}: {set(expected_d_columns) - set(available_d_columns)}")

                # Reorganizando o input filtrado com base nas colunas dispon veis
                filtered_d_input = filtered_d_input[available_d_columns]
                filtered_d_input = filtered_d_input.reindex(columns=expected_d_columns, fill_value=0)

                # Realizando a previs o
                predictions_d = globals()[model_d_name].predict(filtered_d_input) * 100
                predictions_d = np.round(predictions_d, 2)
                prediction_d_str = str(predictions_d).replace("[", "").replace("]", "")

                # Exibindo o resultado individual
                #st.write(f"{decompo_name} est de {prediction_d_str}% sur la consommation d' nergie  actuelle.")

                # Adicionando a previs o ao dataframe de resultados
                df_predictions = pd.concat([df_predictions, pd.DataFrame({'decompo_name': [decompo_name], 'predictions_d': [prediction_d_str]})], ignore_index=True)

            # Display the original df_predictions table
```

```python
                if not df_predictions.empty:
                    df_predictions['predictions_d'] = df_predictions['predictions_d'].astype(float)

                    # Create columns for layout
                    col1, col2 = st.columns([2, 3])  # Three equal columns

                    # First column for editable table
                    with col1:
                        st.write("Modifiez les valeurs de la decompo manuellement si necessaire:")
                        editable_df = df_predictions.copy()
                        updated_predictions_d = []

                        # Loop through each row to create input fields for editing
                        for index, row in editable_df.iterrows():
                            new_value = st.number_input(
                                f"{row['decompo_name']}:",
                                value=row['predictions_d'],
                                key=f"predictions_d_{index}"
                            )
                            updated_predictions_d.append(new_value)

                        # Update the DataFrame with the new values
                        editable_df['predictions_d'] = updated_predictions_d

                        print_decompo = editable_df


                        csv_bytes = io.StringIO()
                        print_decompo.to_csv(csv_bytes, index=False, sep=';', encoding='utf-8')  # Ensure to set the correct separator
                        csv_bytes.seek(0)  # Rewind to the start of the stream

                        # Add download button
                        st.download_button(
                            label="Download decompo data as CSV",
                            data=csv_bytes.getvalue(),
                            file_name='decompo_data.csv',
                            mime='text/csv'  # Ensure MIME type is set for CSV
                        )


                        reverse_correspondence_quanti = {v: k for k, v in correspondence_quanti.items()}
                        editable_df['decompo_key'] = editable_df['decompo_name'].map(reverse_correspondence_quanti)

                        dec_values_first = pd.Series(editable_df.set_index('decompo_key')['predictions_d']
                            name='dec_values')


                    # Second column for plots
                    with col2:
                        # Calculate the total from the updated 'predictions_d'
                        total_sum = editable_df['predictions_d'].sum()

                        # Prepare data for pie chart
                        pie_labels = editable_df['decompo_name'].tolist()
                        pie_sizes = editable_df['predictions_d'].tolist()

                        if total_sum < 100:
                            # Create a figure for the pie chart
                            fig1, ax1 = plt.subplots(figsize=(6, 6))

                            # Calculate the missing part and add it to the pie sizes and labels
                            missing_part = 100 - total_sum
                            pie_sizes.append(missing_part)  # Add the missing part
                            pie_labels.append('Partie manquante')  # Label for the missing part

                            # Set custom colors for the pie chart
                            colors = plt.cm.tab20.colors  # Using a colormap for better colors

                            # Plotting the pie chart
                            ax1.pie(pie_sizes, labels=pie_labels, autopct='%1.1f%%', startangle=90, colors=colors)


                            ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
                            ax1.set_title("Distribution des Predictions y compris les parties qui manquent", pad=20)  # Adjusted title position

                            # Display the pie chart
                            st.pyplot(fig1)
                        else:
                            # If total sum is greater than 100, display a message instead of a chart
                            surplus = total_sum - 100
                            fig2, ax2 = plt.subplots(figsize=(6, 6))  # Create a new figure for the surplus bar chart

                            # Plotting the surplus bar chart with customizations
                            ax2.bar(['Surplus'], [surplus], color='skyblue')  # Changed bar color
                            ax2.set_ylabel('Pourcentage n cessaire d\' tre enlev  de la d compo actuelle (%)', fontsize=8)  # Y-axis label
                            ax2.set_title('Surplus', fontsize=10, pad=20)  # Title for the surplus bar chart
                            ax2.set_xticks([])  # X-axis empty

                            # Display the surplus chart
                            st.pyplot(fig2)


```

```
                else:
                    st.write()



            except KeyError:
                # Custom error message for KeyError
                st.error("Les param tres ins r s ne sont pas valides.")
            except Exception:
                # General error message for any other exceptions
                st.error("Une erreur inattendue est survenue. Veuillez
                    v rifier vos entr es.")




##########################################################################################

        # Pergunta: "Voc  quer gerar as decomposi  es?"
        response2 = st.radio("Voulez-vous g nerer les gains des AAPEs?",
            ("Oui", "Non"), index=1)

        # Se a resposta for "Oui", executa a fun   o
        if response2 == "Oui":


            st.header("Pr diction des AAPEs:")


            # Fun   o para selecionar as AAPEs e criar available_columns
                internamente
            def create_aape_variable_selector(f_types,
                target_to_qualitative_map):
                selected_aapes = st.multiselect("Types d'AAPE:", f_types)  #
                    Multiselect para sele   o
                selected_aape_indices = [f_types.index(col) + 1 for col in
                    selected_aapes]  #  ndices  incrementados

                # Processo interno para criar available_columns
                available_columns = set()
                for index in selected_aape_indices:
                    if index in target_to_qualitative_map:
                        available_columns.update(target_to_qualitative_map[index])
                            # Adiciona colunas sem duplicatas

                return selected_aape_indices, available_columns

            # Executa a fun   o sem expor ao usu rio
            selected_aape_indices, available_columns =
                create_aape_variable_selector(f_types,
                target_to_qualitative_map)

            # A partir daqui, voc  pode usar `selected_aape_indices` e
                `available_columns` conforme necess rio


            # Filter classes_selection based on available_columns
            classes_selection = [value for key, value in correspondence_column_dict.items() if key in available_columns]


            def create_qualitative_variable_selector(classe_list):
                variable_values = {}

                for i, classe in enumerate(classe_list):  # Use enumerate to get an index
                    # Part 1 - List
                    category_name_input = classe
                    values_list = get_values_by_category(correspondence_dict, category_name_input)

                    # Use a unique key for each selectbox
                    selected_q_variable = st.selectbox(f"{classe}:", values_list, key=f"{classe}_quali_{i}")

                    # Update dictionary with selected variable as the key and its category as the value
                    source_category = find_key_by_value(correspondence_column_dict, category_name_input)
                    source_value = find_key_by_value_and_class(selected_q_variable, source_category)
                    variable_values[source_category] = source_value

                return pd.Series(variable_values, name='qualitative_values2')



            # Assuming list_groups is defined somewhere in your code
            selected_qualitative = create_qualitative_variable_selector(classes_selection)


    ########################################################################################

    ######### FIRST INPUT

            def create_numeric_variables(columns):
                numeric_values = {}
                for i, column3 in enumerate(columns):  # Adding index 'i' to ensure unique key
                    max_value = float(quanti_col_max.loc[quanti_col_max['column_names'] == column3, 'real_value'].iloc[0])
                    min_value = float(quanti_col_min.loc[quanti_col_min['column_names'] == column3, 'real_value'].iloc[0])

                    default_value = (max_value - min_value) / 2
                    name = find_corresponding_name(column3, correspondence_quanti)

                    value = st.number_input(
                        "{} avec valeur minimale de {:.2f} et maximale de {:.2f}: ".format(name, min_value, max_value),
                        value=default_value,
                        key=f"num_input_{column3}_{i}"  # Unique key using 'i' and column name
                    )

                    # Verificar se o valor est  dentro dos limites
                    if (min_value is None or value >= min_value) and (max_value is None or value <= max_value):
                        numeric_values[column3] = value
                    else:
                        # Exibir avisos se o valor estiver fora dos limites
                        if min_value is not None and max_value is not None:
                            st.warning("La valeur doit  tre  entre {} et {} pour la variable {}. S'il vous pla t , inserez une nouvelle valeur.".format(min_value, max_value, name))
                        elif min_value is not None:
                            st.warning("La valeur doit  tre  sup rieure   {} pour la variable {}. S'il vous pla t , inserez une nouvelle valeur.".format(min_value, name))
                        elif max_value is not None:
                            st.warning("La valeur doit  tre  inf rieure   {} pour la variable {}. S'il vous pla t , inserez une nouvelle valeur.".format(max_value, name))


                return pd.Series(numeric_values, name='numer_values')


            # Filter f_gen based on available_columns
            column_list_a = [column for column in f_gen if column in available_columns]

            selected_quantitative = create_numeric_variables(column_list_a)




            ##############################################################################



            ### Second input

            def aape_decompo(columns):

                dec_values = {}
                for i, column in enumerate(columns):  # Adding index 'i' to ensure unique key
                    max_value = float(quanti_col_max.loc[quanti_col_max['column_names'] == column, 'real_value'].iloc[0])*100
                    min_value = float(quanti_col_min.loc[quanti_col_min['column_names'] == column, 'real_value'].iloc[0])*100

```

```
        default_value = (max_value - min_value) / 2
        name = find_corresponding_name(column, correspondence_quanti)

        value = st.number_input(
            "{} avec valeur minimale de {:.2f} et maximale de {:.2f}: ".format(name, min_value, max_value),
            value=default_value,
            key=f"num_input_{column}_{i}"  # Unique key using 'i' and column name
        )

        # Verificar se o valor est  dentro dos limites
        if (min_value is None or value >= min_value) and (max_value is None or value <= max_value):
            dec_values[column] = value/100
        else:
            # Exibir avisos se o valor estiver fora dos limites
            if min_value is not None and max_value is not None:
                st.warning("La valeur doit  tre  entre {} et {} pour la variable {}. S'il vous pla t, inserez une nouvelle valeur.".format(min_value, max_value, name))
            elif min_value is not None:
                st.warning("La valeur doit  tre  sup rieure   {} pour la variable {}. S'il vous pla t, inserez une nouvelle valeur.".format(min_value, name))
            elif max_value is not None:
                st.warning("La valeur doit  tre  inf rieure   {} pour la variable {}. S'il vous pla t, inserez une nouvelle valeur.".format(max_value, name))


    return pd.Series(dec_values, name='dec_values')

 # Filter f_gen based on available_columns
column_list_dec = [column for column in f_pEF if column in available_columns]

# Inicializando a s rie e a lista de missing_values
selected_decomposition = pd.Series(dtype=float)
missing_values = []

# Itera  o sobre as colunas da lista
for column in column_list_dec:
    if column in dec_values_first.index:
        # Adiciona o valor correspondente   s rie
        selected_decomposition[column] = dec_values_first[column]
    else:
        # Adiciona o nome da coluna   lista de missing_values
        missing_values.append(column)

# Verifica  o e aplica  o da fun  o aape_decompo se houver missing_values
if missing_values:
    missing_values_series = aape_decompo(missing_values)

```

```
            # Concatenando a missing_values_series    selected_decomposition
            selected_decomposition = pd.concat([selected_decomposition,
                missing_values_series])


        ##############################################################################

        # Streamlit UI for user input
        st.subheader("R sultat de la prediction des AAPEs:")

        export_df = pd.DataFrame(columns=['AAPE', 'Gain (%)'])


        try:

            result = pd.concat([selected_quantitative,
                selected_qualitative, selected_decomposition], axis=0)
            # Suponha que 'series' seja a sua Series com o nome desejado
            result = result.to_frame(name='Nome da Coluna')

            # Supondo que seu DataFrame seja chamado result
            result = result[result['Nome da Coluna'] != 'Nome da Coluna']
            result2 = result.T


            # Assuming selected_aape_indices is a list of your AAPE values
            for idx in selected_aape_indices:
                # Get the relevant AAPE name using idx - 1
                aape_name = f_types[idx - 1]

                # Get the relevant variables for the current AAPE
                parameters_aape = target_to_qualitative_map[idx]

                # Filter result based on the relevant parameters
                filtered_input = result2[parameters_aape]

                # Use the corresponding model for predictions
                model_name = f'model{idx}'  # Assuming models are named
                    model1, model2, ..., model12

                predictions = globals()[model_name].predict(filtered_input)
                    * 100
                predictions = np.round(predictions, 2)

                # Prepare the prediction for display
                prediction_str = str(predictions).replace("[",
                    "").replace("]", "")
                export_df = pd.concat([export_df, pd.DataFrame({'AAPE':
                    [aape_name], 'Gain (%)': [prediction_str]})],
                    ignore_index=True)

                # Display the results
                st.write(f"Le gain   nergtique   de l'AAPE {aape_name} est
                    de {prediction_str}% de la consommation d'  nergie
                    actuelle.")

                # Convert the DataFrame to CSV for download


            # Convert export_df DataFrame to CSV using BytesIO
            csv_export_bytes = io.StringIO()
            export_df.to_csv(csv_export_bytes, index=False,
                encoding='utf-8')
            csv_export_bytes.seek(0)  # Rewind to the start of the stream

            # Add download button
            st.download_button(
                label="Download AAPE Gains as CSV",
                data=csv_export_bytes.getvalue(),
                file_name='aape_gains.csv'
            )


        except KeyError:
            # Custom error message for KeyError
            st.error("Les param tres ins r s ne sont pas valides.")

        except Exception:
            # General error message for any other exceptions
            st.error("Une erreur inattendue est survenue. Veuillez
                v rifier vos entr es.")



elif page == "Analyse des bases de donn es":

    st.write('')
    st.write('')
    st.write('')

    col_a,col_inexistante, col_b = st.columns([10,1,10])


    col_a.header("Analyse des variables quantitatives")
    col_a.write('')

    col_a.write('')


    # Dropdown menu for selecting column
    name_a = col_a.selectbox('Selectionnez un
        param tre',list(plots_quanti.values()))

    selected_col = find_key(plots_quanti,name_a)


    name_a = find_corresponding_name(selected_col,plots_quanti)

    # Slider for selecting x-axis interval
    x_min, x_max = col_a.slider('Selectionez l\'intervale de Consommation
        (kWh/m  .an)', min_value=float(data['EF.total'].min()),
                         max_value=float(data['EF.total'].max()),
                            value=(float(data['EF.total'].min()),
                            float(data['EF.total'].max())))

    # Filter DataFrame based on selected interval
    filtered_df = data[(data['EF.total'] >= x_min) & (data['EF.total'] <= x_max)]

    y_min, y_max = col_a.slider(f'Selectionez l\'intervale pour {name_a}', min_value=float(filtered_df[selected_col].min()),
                     max_value=float(filtered_df[selected_col].max()), value=(float(filtered_df[selected_col].min()), float(filtered_df[selected_col].max())))

    # Filter DataFrame based on selected interval
    filtered_df = filtered_df[(filtered_df[selected_col] >= y_min) & (filtered_df[selected_col] <= y_max)]

    plt.figure()
    plt.scatter(filtered_df['EF.total'], filtered_df[selected_col], alpha=0.3, c='green')
    plt.xlabel('Consommation (kWh/m  .an)')
    plt.ylabel(name_a)
    plt.title(f'Correlation entre {name_a} \n et la Consommation (kWh/m  .an)')
    plt.grid(True)

    # Display plot in Streamlit
    col_b.pyplot(plt)



    color_palette = "pastel"

    st.subheader("Analyse des variables qualitatives")

    col_c,col_inexistant, col_d = st.columns([10,1,10])


    # Lista de colunas qualitativas
    graph_geral = data_quali



    y_min, y_max = col_c.slider('Selectionez l\'intervale pour la Consommation (kWh/m  .an)',min_value=float(graph_geral['EF.total'].min()),
                      max_value=float(graph_geral['EF.total'].max()), value=(float(graph_geral['EF.total'].min()), float(graph_geral['EF.total'].max())))

    # Copie o DataFrame original para que a filtragem n o afete os dados originais
    graph_conso = graph_geral.copy()
    # Filtrar DataFrame com base no intervalo selecionado
    graph_conso = graph_conso[(graph_conso['EF.total'] >= y_min) & (graph_conso['EF.total'] <= y_max)]



    # Slider for selecting column
    name_c = col_c.selectbox('Selectionnez un param tre',
        list(plots_quali.values()))

    selected_column = find_key(plots_quali,name_c)

    graph_conso_ape = pd.DataFrame()

    graph_conso_ape['ape'] = graph_conso[selected_column]

    graph_conso_ape = graph_conso_ape.rename(columns={'ape':
        selected_column})

    graph_conso_ape['EF.total'] = graph_conso['EF.total']

    selected_classes = {}

    # Obtenha a lista de valores  nicos  da coluna selecionada
    unique_values = graph_conso_ape[selected_column].unique().tolist()

    # Use a lista completa como valor padr o
    default_value = unique_values

    # Use a lista completa como valor padr o na fun   o multiselect
    selected_classes[selected_column] = col_d.multiselect(f'Selectionnez
        les classes de {name_c}:', unique_values, default=default_value)

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

    # Filtrar DataFrame com base nas classes selecionadas
    for col, classes in selected_classes.items():
        graph_conso_ape =
            graph_conso_ape[graph_conso_ape[col].isin(classes)]

        # Criar gr fico de barras mostrando a frequ ncia de cada
            categoria na coluna atual
        freq = graph_conso_ape[col].value_counts()
        sns.barplot(x=freq.index, y=freq.values, palette=color_palette,
            ax=ax1)
        # Personalizar r tulos e t tulo do gr fico de barras
        ax1.set_xlabel(f'{name_c}')
        ax1.set_ylabel('Frequence')
        ax1.set_title(f'Frequence de {name_c}')
        ax1.set_xticklabels(ax1.get_xticklabels(), rotation=45)

        # Definir a ordem das categorias com base no gr fico de barras
            para consist ncia
        order = freq.index

        # Criar um gr fico de violino lado a lado
        sns.violinplot(x=graph_conso_ape[col],
            y=graph_conso_ape['EF.total'], order=order,
            data=graph_conso_ape, palette=color_palette, linewidth=0,
            ax=ax2)

        # Personalizar r tulos e t tulo do gr fico de violino
        ax2.set_xlabel(f'{name_c}')
        ax2.set_ylabel('Consommation (kWh/m .an)')
        ax2.set_title(f'Correlation  entre {name_c} et Consommation (kWh/m .an)')
        ax2.set_xticklabels(ax2.get_xticklabels(), rotation=45)

    # Exibir o gr fico
    st.pyplot(fig)


#elif page == "Analyse du M tamod le":

 #   st.header("Analyse de la performance de la sensibilit  du m tamod le")

if __name__ == "__main__":
main()
```

Listing C.8 – Code 7- Metamodel