



**UNIVERSITATEA TEHNICĂ**

DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**  
**DEPARTAMENTUL CALCULATOARE**

**PREDICTION AND OPTIMIZATION OF ENERGY CONSUMPTION IN PEAK INTERVALS IN NON-RESIDENTIAL BUILDINGS USING METAHEURISTIC ALGORITHMS**

LUCRARE DE LICENȚĂ

Absolvent: **Daniel-Ioan BOAR**

Coordonator: **Asist. Ing. Mircea Gabriel ANTONESCU**  
științific:

2025

## Cuprins

|  |           |
|--|-----------|
| <b>Capitolul 1. Introducere .....</b>                          | <b>1</b>  |
| 1.1. Contextul aplicației.....                                 | 1         |
| 1.2. Motivația.....  | 2         |
| <b>Capitolul 2. Obiectivele proiectului .....</b>              | <b>3</b>  |
| 2.1. Obiectivul principal .....                                | 3         |
| 2.2. Obiective secundare.....                                  | 3         |
| <b>Capitolul 3. Studiu bibliografic.....</b>                   | <b>6</b>  |
| <b>Capitolul 4. Analiză și fundamentare Teoretică.....</b>     | <b>12</b> |
| 4.1. Estimarea consumului de referință .....                   | 12        |
| 4.2. Coeficientul de corelație Pearson .....                   | 13        |
| 4.3. Profilul de consum energetic .....                        | 14        |
| 4.4. Algoritmi de predicție energetică .....                   | 15        |
| 4.4.1. Random Forest (RF) .....                                | 15        |
| 4.4.2. Multi-Layer Perceptron (MLP) .....                      | 16        |
| 4.4.3. Long Short-Term Memory (LSTM) .....                     | 18        |
| 4.5. Algoritmi de optimizare a consumului energetic.....       | 19        |
| 4.5.1. Model Predictive Control (MPC) .....                    | 19        |
| 4.5.2. Genetic Algorithm (GA).....                             | 20        |
| 4.5.3. Particle Swarm Optimization (PSO).....                  | 22        |
| 4.5.4. Ant Colony Optimization (ACO) .....                     | 23        |
| 4.6. Cerințe funcționale și non-funcționale.....               | 24        |
| 4.7. Tehnologii.....   | 25        |
| 4.7.1. Python.....   | 25        |
| 4.7.2. Flask.....  | 25        |
| 4.7.3. Flasgger .....  | 26        |
| 4.7.4. PostgreSQL.....   | 26        |
| 4.7.5. React .....   | 26        |
| <b>Capitolul 5. Proiectare de detaliu și implementare.....</b> | <b>27</b> |
| 5.1. Calculul baseline-ului .....                              | 28        |
| 5.2. Predicția consumului energetic.....                       | 29        |
| 5.2.1. Implementarea modelului Random Forest .....             | 30        |

|   |           |
|---|-----------|
| 5.2.2. Implementarea modelului Multi-Layer Perceptron .....       | 30        |
| 5.2.3. Implementarea modelului LSTM .....                         | 31        |
| 5.3. Calcularea profilului de consum.....                         | 32        |
| 5.4. Optimizarea consumului energetic .....                       | 33        |
| 5.4.1. Implementarea unui Model Predictive Control .....          | 33        |
| 5.4.2. Implementarea optimizatorului ACO .....                    | 34        |
| 5.4.3. Implementarea optimizatorului PSO .....                    | 35        |
| 5.4.4. Implementarea optimizatorului GA.....                      | 36        |
| 5.5. Implementarea dashboard-ului .....                           | 37        |
| 5.5.1. Backend .....  | 37        |
| 5.5.2. Frontend.....  | 44        |
| <b>Capitolul 6. Testare și validare .....</b>                     | <b>47</b> |
| 6.1. Alegerea setului de date.....                                | 47        |
| 6.2. Identificarea caracteristicilor meteorologice (Pearson)..... | 49        |
| 6.3. Procesul de feature engineering.....                         | 50        |
| 6.4. Metrici de performanță utilizate.....                        | 52        |
| 6.5. Analiza modelelor de predicție .....                         | 53        |
| 6.6. Analiza algoritmilor de optimizare .....                     | 58        |
| <b>Capitolul 7. Manual de instalare si utilizare.....</b>         | <b>64</b> |
| 7.1. Instalarea aplicației .....                                  | 64        |
| 7.1.1. Cerințele software.....                                    | 64        |
| 7.1.2. Cerințele hardware.....                                    | 64        |
| 7.1.3. Configurare backend.....                                   | 65        |
| 7.1.4. Configurare frontend .....                                 | 66        |
| 7.2. Utilizarea aplicației .....                                  | 66        |
| 7.2.1. Pagina de Dashboard .....                                  | 66        |
| 7.2.2. Pagina Prediction.....                                     | 67        |
| 7.2.3. Pagina Optimization .....                                  | 68        |
| 7.2.4. Navigarea între pagini .....                               | 68        |
| <b>Capitolul 8. Concluzii.....</b>                                | <b>69</b> |
| 8.1. Contribuții și realizări .....                               | 69        |
| 8.2. Limitări ale sistemului propus .....                         | 70        |
| 8.3. Direcții viitoare de dezvoltare .....                        | 70        |
| <b>Bibliografie.....</b>  | <b>72</b> |

# Capitolul 1. Introducere

În ultimii ani, consumul de energie în clădirile non-rezidențiale s-a transformat într-o provocare semnificativă pentru domeniul optimizării energetice. Digitalizarea sistemelor HVAC (încălzire, ventilație și aer condiționat), împreună cu creșterea constantă a prețurilor la energie, au determinat tot mai multe instituții și companii să identifice soluții inteligente pentru monitorizarea și optimizarea consumului energetic. Integrarea tehnologiilor avansate de analiză a datelor și a algoritmilor de învățare automată, a facilitat apariția unor noi direcții de cercetare. În special, aceasta vizează dezvoltarea unor modele predictive tot mai precise și implementarea unor strategii de optimizare adaptive, cu scopul de a spori eficiența și sustenabilitatea energetică în mediul construit.

Această lucrare abordează problema optimizării consumului de energie electrică în clădirile non-rezidențiale, apelând la o metodologie algoritmică care combină predicții automate și strategii de optimizare avansate. Proiectul integrează mai multe componente importante, precum prelucrarea unui set complex de date istorice referitoare la consumul energetic și la condițiile meteorologice, dezvoltarea unor modele de predicție bazate pe algoritmi, precum Random Forest, MLP și LSTM, cât și implementarea unor tehnici de optimizare inspirate din natură, cum ar fi ACO, PSO și GA. Aceste elemente sunt integrate într-un sistem unitar, accesibil prin intermediul unei interfețe web, care oferă în timp real atât prognoze orare și zilnice, cât și recomandări de optimizare a consumului.

Astfel, lucrarea demonstrează modul în care metodele moderne de inteligență artificială pot fi integrate pentru a sprijini adoptarea unor decizii stabile privind consumul energetic la nivelul clădirilor.

## 1.1. Contextul aplicației

Sectorul clădirilor non-rezidențiale, cum ar fi școlile, birourile, centrele comerciale sau spațiile publice, reprezintă o parte semnificativă în consumul total de energie electrică. În această situație, este importantă optimizarea consumului energetic, atât pentru reducerea costurilor, cât și pentru creșterea stabilității și diminuarea impactului ecologic. Adoptarea unor soluții bazate pe analiza datelor (*data-driven*), oferă administratorilor acestor clădiri o perspectivă mai clară asupra funcționării sistemelor HVAC, ajutând la anticiparea vârfurilor de consum și intervenția preventivă.

Tehnologiile moderne din domeniul rețelelor inteligente și al infrastructurilor IoT, permit monitorizarea detaliată și aproape instantanee a consumului de energie. Totuși, analiza în timp real nu are un impact semnificativ fără integrarea unor sisteme capabile să învețe din trecut și să ofere prognoze solide pentru consumul viitor. Mai mult, nu este suficient să realizăm doar predicții, ci este esențial să existe mecanisme active de optimizare, care să ajusteze dinamic consumul estimat, având în vedere obiective precum reducerea vârfurilor de sarcină sau diminuarea consumului total de energie.

Prin urmare, această lucrare abordează o temă relevantă în sfera cercetării actuale, concentrându-se pe dezvoltarea unui sistem, capabil să ofere nu doar prognoze precise, ci și soluții de optimizare energetică personalizate pentru fiecare zi și fiecare clădire în parte. Proiectul utilizează date reale, extrase din setul de date publicat de Miller et al. [32], folosindu-se exclusiv de clădirile din Florida, pentru perioada 2016-

2017, incluzând atât măsurători energetice, cât și meteorologice. Astfel, setul de date rezultat include variabile esențiale precum data și ora, temperatura aerului, direcția și viteza vântului, umiditatea și precipitațiile, oferind o bază solidă pentru analiza și dezvoltarea soluțiilor propuse.

Prin urmare, sistemul prezentat aici permite integrarea predicțiilor automate direct în procesul de optimizare, adaptându-se la profilul specific de consum și la comportamentul anticipat al clădirii. Soluțiile propuse nu se rezumă la simple analize statistice, ci urmăresc efectiv reducerea consumului în intervalele de vârf și asigurarea unei distribuții energetice cât mai echilibrate.

## **1.2. Motivația**

Motivația acestei cercetări este strâns legată de nevoia de a îmbunătăți precizia și eficiența instrumentelor de prognoză și optimizare a consumului energetic în clădirile non-rezidențiale. În prezent, majoritatea sistemelor de management energetic funcționează reactiv, fără să integreze predicții automate sau soluții inteligente de redistribuire a sarcinii energetice. Prin urmare, proiectul de față urmărește să depășească aceste limitări printr-o soluție, care combină tehnici de învățare automată cu algoritmi de optimizare. Punctul de plecare îl reprezintă analiza datelor istorice și a parametrilor meteorologici, cu scopul de a anticipa consumul energetic pe intervale orare și de a ajusta în mod eficient utilizarea resurselor.

Un alt aspect important îl reprezintă absența unor soluții specializate pentru birouri, școli sau spații comerciale, care au un profil de consum energetic mult mai variabil comparativ cu locuințele. În astfel de cazuri, consumul de energie depinde de factori precum programul de utilizare, comportamentul colectiv al ocupanților, condițiile climatice locale sau structura instalațiilor HVAC. Acest context impune dezvoltarea unor modele adaptabile, capabile să reflecte și să răspundă acestor fluctuații complexe.

Având în vedere creșterea accentuată a prețurilor la energie și presiunile tot mai mari impuse de politicile europene privind eficiența energetică, prognoza consumului și optimizarea acestuia devin absolut esențiale. Nu mai este suficientă estimarea aproximativă deoarece sistemele actuale trebuie să integreze o gamă largă de date, de la istoricul consumului zilnic până la programe meteorologice detaliante. Scopul este generarea unor decizii rapide, orientate spre reducerea consumului de energie și atenuarea vârfurilor orare de consum, asigurând astfel o eficiență operațională, precum și o stabilitate financiară.

Pe măsură ce sistemele energetice devin tot mai interconectate și axate pe date, devine esențial să dispunem de metodologii riguroase, capabile să surprindă atât comportamentul uman, cât și impactul factorilor externi asupra cererii de energie. Scopul principal al acestei inițiative este de a dezvolta un instrument care să poată anticipa consumul energetic la nivel orar sau zilnic și, pornind de la aceste programe, să genereze un profil de consum optimizat, adaptat specificului fiecărei clădiri în parte.

Obiectivul principal constă în dezvoltarea unui sistem capabil să anticipeze și să optimizeze procesele, astfel încât să poată fi implementat în situații reale. Prin această abordare, se urmărește reducerea consumului energetic și promovarea sustenabilității în mediile urbane, contribuind la îmbunătățirea calității vieții și la protejarea mediului.

## **Capitolul 2. Obiectivele proiectului**

### **2.1. Obiectivul principal**

Scopul principal al acestei lucrări este dezvoltarea unei soluții pentru predicția și optimizarea consumului de energie electrică în clădiri non-rezidențiale, cu o focalizare specială pe sistemele HVAC (Heating, Ventilation and Air Conditioning). Lucrarea urmărește să analizeze comportamentul energetic specific acestor tipuri de clădiri, utilizând atât date istorice, cât și informații meteorologice relevante. Pe baza acestor date, se va implementa o metodologie care să permită anticiparea consumului zilnic de energie și optimizarea distribuției acestuia, ținând cont de intervalele orare caracterizate prin vârfuri de consum și perioade cu cerere redusă. Astfel, această metodologie va ajuta la optimizarea utilizării resurselor energetice.

Această lucrare abordează două aspecte complementare. Pe de o parte, predicția consumului pe termen scurt cu ajutorul unor modele avansate de Machine Learning, iar pe de altă parte, optimizarea redistribuirii energiei în decursul zilei, folosind algoritmi inspirați din procese naturale. Pentru validarea acestor metode, s-au utilizat date reale de consum energetic provenite de la clădiri din statul Florida (USA), disponibile dintr-un set de date științific public, la care se adaugă informații meteorologice relevante.

În ceea ce privește partea de predicție, au fost testate și comparate mai multe modele, precum Random Forest, MLP și LSTM, urmărindu-se performanțele acestora cu ajutorul unor metri tradiționale. În final, modelul LSTM a fost selectat pentru etapa de optimizare datorită preciziei sale ridicate și capacitatei de a interpreta într-un mod eficient secvențe de date temporale. Pe baza valorilor prezise de acest model, au fost generare profiluri orare de consum energetic, care ulterior au fost folosite pentru a aplica diferite metode de optimizare, precum Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) și Genetic Algorithm (GA).

Prin urmare, ideea centrală a fost să se dezvolte un sistem care poate face simultan predicții și optimizări adaptive ale consumului de energie, astfel încât să se reducă cât mai mult valoarea totală a energiei utilizate, mai ales în momentele de vârf. Acest demers contribuie atât la optimizarea rețelelor locale, cât și la scăderea costurilor operaționale, asigurând o funcționare mai stabilă și mai durabilă a infrastructurii energetice.

### **2.2. Obiective secundare**

Conform celor prezentate în subcapitolul de mai sus, tabelul următor prezintă obiectivele secundare care contribuie la îndeplinirea scopului principal, împreună cu capitoile relevante în care acestea sunt analizate.

| Nume   | Descriere   | Capitol corespunzător |
|--|---|-----------------------|
| Realizarea unei analize a algoritmilor de predicție și optimizare energetică | Se analizează diverse tehnici de predicție, împreună cu metode algoritmice de optimizare, evidențiindu-se avantajele și limitările fiecăreia. | Capitolul 3           |

|  |   |             |
|--|---|-------------|
| Fundamente teoretice privind predicția și optimizarea consumului energetic                           | Aceste fundamente reprezintă calcularea baseline-ului, analiza corelațiilor meteorologice, profilul de consum, modelele de predicție și de optimizare, precum și tehnologiile folosite în crearea sistemului.   | Capitolul 4 |
| Proiectarea și dezvoltarea algoritmilor de predicție   | Se utilizează modele precum Random Forest, MLP și LSTM, aplicate pe seturi de date istorice și meteorologice. Accentul cade pe ajustarea parametrilor și maximizarea performanței acestor modele, pentru a obține rezultate cât mai relevante și precise.   | Capitolul 5 |
| Construirea profilului zilnic de consum pe baza predicției și valorilor de referință                 | Se realizează o structură de date ce integrează baseline-ul, valorile reale ale consumului și predicția LSTM, oferind o prezentare orară a evoluției consumului pentru ziua selectată. Această abordare permite observarea variațiilor consumului în timp și ajută la compararea între valorile estimate și cele reale.   | Capitolul 5 |
| Aplicarea metodelor de optimizare pentru redistribuirea consumului                                   | Se implementează metodele de optimizare precum ACO, PSO și GA, cu scop în reducerea consumului total de energie și în diminuarea curbei de sarcină.   | Capitolul 5 |
| Realizarea interfeței de tip dashboard, integrarea API-urilor REST și stocarea datelor în PostgreSQL | Se dezvoltă o aplicație frontend care ajută la vizualizarea grafică a datelor și permite accesul la informații prin intermediul API-urilor REST. În plus, rezultatele obținute pot fi salvate în tabele gestionate (PostgreSQL), utilizând ORM-ul SQLAlchemy. Astfel, aplicația integrează funcționalități de afișare vizuală, prelucrare de date și stocare eficientă. | Capitolul 5 |
| Compararea modelelor de predicție și selecția celui mai eficient                                     | Fiecare model este evaluat critic folosind metricile MSE, MAE, $R^2$ și SMAPE, iar modelul care obține cea mai înaltă performanță este selectat pentru utilizare.   | Capitolul 6 |
| Evaluarea și vizualizarea comparativă a optimizatorilor  | Fiecare algoritm de optimizare este analizat atent, având în vedere atât rezultatele grafice, cât și economiile realizate. Aceste comparații ajută la determinarea  | Capitolul 6 |

|  |   |  |
|--|---|--|
|  | variantei care oferă cea mai bună eficiență energetică. Scopul este identificarea soluției optime, bazându-se pe date concrete, nu doar pe rezultate teoretice. |  |
|--|---|--|

Tabel 2.1 Obiectivele secundare ale lucrării

## **Capitolul 3. Studiu bibliografic**

Optimizarea energiei în sistemele HVAC este esențială atunci când se dorește reducerea consumului global de energie și tranziția către clădirile inteligente. Odată cu creșterea numărului de clădiri echipate cu senzori, dispozitive inteligente și diferite surse de date energetice, a crescut interesul pentru automatizarea și optimizarea performanței acestor sisteme. Multe articole științifice din ultimii ani au propus modele avansate pentru predicția consumului de energie, analiza eficienței sistemelor HVAC și îmbunătățirea utilizării resurselor prin metode matematice, tehnici de învățare automată și algoritmi evolutivi. Aceste tehnici sunt dezvoltate pentru a optimiza eficiența energetică și a reduce cheltuielile de funcționare, în timp ce respectă normele de confort și sustenabilitate ale clădirilor contemporane.

Un element esențial în domeniul cercetării este benchmark-ul energetic, care constă în identificarea clădirilor cu consum mare în raport cu altele de performanță similară. O lucrare semnificativă în acest context este cea a lui Wang et al. [1], care introduce un sistem de măsurare a eficienței energetice utilizând clasificarea prin grupare variabilă (clustering) și o metodologie multi-criterială TOPSIS. Această cercetare utilizează ponderarea entropică pentru a evidenția influența relativă a fiecărui factor. Pe de altă parte, Li et al. [2] analizează diferite abordări de comparare a consumului energetic cu date anterioare sau preconizate, evidențiind că referințele dinamice sunt esențiale pentru analiza eficienței. În plus, Du et al. [3] propune un model dublu de benchmark, care permite evaluarea și ajustarea strategiilor de control HVAC bazate pe două tipuri de referințe, contribuind astfel la îmbunătățirea preciziei deciziilor de optimizare. De asemenea, Bichiou și Krarti [4] combină metode de analiză multiobiectivă și optimizare pentru selectarea anvelopei adecvate clădirii și sistemului HVAC.

Pentru a optimiza procesul de benchmark, cercetătorii se orientează acum spre integrarea modelelor matematice avansate și a rețelelor neuronale. De exemplu, în articolul lui Yalcintas și Ozturk [5], se aplică rețelele neuronale artificiale pentru a prezice indicele EUI (Energy Use Intensity) folosind datele CBECS, obținând rezultate mai bune decât modelele liniare clasice. Studiul lor subliniază importanța alegerii variabilelor relevante în estimarea performanței energetice. Într-o aplicație practică, Wang et al. [6] prezintă un studiu de comisionare care evaluatează consumul efectiv într-o clădire de birouri. Acest studiu folosește valorile de referință stabilite mai devreme pentru a găsi abaterile și pentru a corecta funcționarea sistemului HVAC.

În plus, benchmark-ul poate fi extins prin adăugarea evaluărilor dinamice de performanță. Un exemplu semnificativ este oferit de Singh et al. [7], care propune un cadru care combină controlul predictiv bazat pe model (MCP) cu rețelele neuronale și evaluări în timp real. Folosirea acestei tehnici permite definirea unor repere energetice mult mai exacte și mai adaptabile, ceea ce facilitează detectarea anomaliei. În cadrul acestei lucrări, sunt utilizate metode precum clustering-ul, Association Rule Mining și corectarea valorilor anormale pentru a ajuta la generarea benchmark-urilor sezoniere și stratificate. Aceste tehnici ajută la detecția rapidă a comportamentului energetic neobișnuit și permit adaptarea în timp real a strategiilor de control HVAC, asigurând o eficiență energetică superioară și un control mai exact al sistemelor.

Pe lângă benchmark, studiul comportamentului sistemelor HVAC în utilizare devine un aspect esențial al cercetării, în special în ceea ce privește detectarea

defectelor, optimizarea ajustărilor și evaluarea calității întreținerii. În această zonă, multe articole pun un accent tot mai mare pe strategiile de întreținere predictivă și pe metodele de identificare automată a anomalilor. Obiectivul lor este de a înlocui mențenanța reactivă obișnuită cu una care folosește diagnosticare automată și evaluarea duratei de viață a componentelor.

În studiul realizat de Capozzoli et al. [8], este prezentată o strategie de identificare a defectelor în clădirile inteligente, prin aplicarea unor tehnici de data mining pe un set de date adunate din birouri inteligente. Această metodă demonstrează că putem identifica comportamente anormale fără a realiza o modelare fizică completă a sistemului HVAC. Pe de altă parte, în articolul lui Taheri et al. [9] au fost aplicate rețele neuronale profunde pentru identificarea defectelor în sistemele HVAC. Ei au obținut rezultate favorabile în identificarea rapidă a comportamentelor neobișnuite, ambele studii evidențiind abilitatea învățării automate de a îmbunătății fiabilitatea echipamentelor și de a diminua costurile de întreținere.

Modul în care tehniciile de învățare automată și învățare profundă pot contribui la creșterea eficienței și optimizarea calității mediului interior a fost prezentat în articolul științific al lui Tien et al. [10] și evidențiază atât algoritmii utilizați, cât și beneficiile și constrângerile acestora. Autorii subliniază importanța includerii acestor tehnologii în sistemele de management al clădirilor, pentru a facilita reglarea dinamică și eficientă a condițiilor de mediu. De asemenea, în articolul lui Namburu et al. [11] se analizează modelarea bazată pe date, alegerea optimă a senzorilor și diagnosticarea anomalilor pentru chillere HVAC, folosind tehnici de învățare automată pentru a dezvolta modele de predicție eficiente.

O contribuție semnificativă este realizată de Gaur et al. [12] prin analiza eficienței unor tehnici de identificare a consumului neobișnuit, aplicând indicatori de precizie și examinând capacitatea algoritmilor de a face diferență între comportamente energetice normale și abateri notabile. Aceste metode se arată a fi extrem de utile pentru clădirile mari, unde complexitatea sistemului HVAC și cantitatea mare de date colectate fac analiza manuală complicată.

Un alt element esențial vizează evaluarea performanței sistemelor HVAC, în care analiza se extinde dincolo de consumul de energie, incluzând și capacitatea sistemului de a furniza confort termic, stabilitate operațională și funcționare eficientă din punct de vedere economic. Astfel, articolul realizat de Si et al. [13] prezintă o clasificare amănunțită a indicatorilor de performanță și o analiză comparativă a tehniciilor utilizate pentru a evalua eficiența algoritmilor implementați în proiectarea și reglarea sistemelor HVAC. Studiul prezintă un grup de criterii atât numerice, cât și descriptive pentru a evalua strategiile de optimizare, cum ar fi consumul total de energie, timpii de reacție și stabilitatea pe termen lung. Aceste elemente sunt esențiale pentru o evaluare solidă a performanței sistemelor de climatizare și ventilație.

Analiza performanței sistemelor include tot mai des metode de simulare și modelare semi-empirică, tocmai pentru a evalua complexitatea sistemului. În acest context, articolul Valkiloroaya et al. [14] sugerează o sinteză între modelarea empirică și simularea numerică, aplicând optimizarea pe bază de gradient pentru a asigura o ajustare exactă a sistemului HVAC. Această metodă oferă un echilibru între precizie și complexitatea procesului de calcul, fiind extrem de utilă pentru clădirile existente, unde sistemele au o structură deja stabila.

Tot în zona evaluării performanței, articolul realizat de Yu et al. [15] sugerează un algoritm care se bazează pe optimizarea Lyapunov, având ca scop reducerea costurilor energetice și asigurarea calității aerului din interior. Această soluție prezintă o viziune asupra modului în care funcționează aceste sisteme prin combinarea datelor

referitoare la prețul energiei, la numărul de utilizatori și la preferințele de temperatură pentru o optimizare în timp real.

Un alt domeniu relevant este legat de analiza profilului de consum energetic al clădirilor, precum și segmentarea acestora pe baza unor comportamente energetice similare. Acest lucru face posibilă identificarea tiparelor orare și sezoniere de utilizare a energiei, luând în calcul factori precum gradul de ocupare și condițiile meteorologice. În studiul realizat de Yang și Becerik-Gerber [16] se specifică faptul că adaptarea programelor HVAC în funcție de profilul de ocupare, împreună cu realocarea spațiilor, poate reduce considerabil consumul de energie. Rezultatele arată faptul că o metodă dinamică de gestionare a sistemului HVAC, asociată cu utilizarea efectivă a spațiului, generează o eficiență superioară. În plus, autorii evidențiază că integrarea acestor date cu algoritmi de programare poate ajuta la stabilirea unor orare HVAC mult mai eficiente din punct de vedere energetic.

În ceea ce privește profilul de consum, Ley et al. [17] evaluatează eficiența metodelor actuale de stabilire a baseline-ului pentru cererile de tip „Răspuns la cerere”, bazându-se pe ventilatoarele HVAC. Rezultatele demonstrează că metodele statice, cum ar fi media aritmetică, nu reușesc să capteze variațiile dinamice ale consumului, în special în intervalele cu mari fluctuații. Autorii susțin că este nevoie de modele de referință mai adaptabile, care să reflecte particularitățile comportamentale ale fiecărei clădiri. În completare, Yang et al. [18], aduc o contribuție suplimentară în acest sens, analizând legătura dintre diversificarea profilului de ocupare și eficiența energetică a sistemelor HVAC. Aceste studii demonstrează importanța adaptării metodelor de consum la specificul fiecărei clădiri și la comportamentul oamenilor din acea instituție.

Clustering-ul joacă un rol esențial în analizarea și gestionarea datelor despre consumul clădirilor, ajutând la gruparea acestora pe baza unor tipare similare de consum energetic. Așadar, Gao și Malkawi [19], oferă o metodologie avansată de benchmark care utilizează clustering-ul pentru a asigura comparații relevante între clădiri cu caracteristici comune. Prin utilizarea unor algoritmi de k-means, se realizează formarea unor grupuri omogene, care sunt esențiale pentru dezvoltarea unor strategii HVAC adaptate fiecărei categorii identificate. În plus, Zekić-Sušac et al. [20] integrează clustering-ul cu rețelele neuronale pentru a evalua eficiența energetică a clădirilor publice, obținând rezultate încurajatoare în identificarea oportunităților de optimizare. Autorii evidențiază faptul că o segmentare adecvată a clădirilor poate conduce la alegerea unor metode mai potrivite și, în consecință, la o gestionare mai eficientă a consumului de energie.

Nikolaou et al. [21] aduc un plus de detaliu studiului, aplicând metode de clustering nu doar pentru a clasifica clădirile în funcție de consumul energetic, ci și pe baza nivelului de confort termic resimțit de ocupanți. Includerea acestor criterii facilitează o înțelegere mai bună a impactului pe care opiniiile utilizatorilor îl au asupra performanței energetice. Astfel, prin integrarea atât a informațiilor obiective despre consum, cât și a opinioilor subiective ale oamenilor, se pot identifica clustere care ilustrează atât eficiența energetică, cât și nivelul de confort interior. Aceste concluzii pot fi folosite pentru ajustarea strategiilor HVAC într-un mod specific, cu scopul de a crește nivelul de satisfacție al oamenilor și de a optimiza utilizarea resurselor în funcție de necesitățile reale ale acestora.

O altă componentă importantă a cercetării actuale o reprezintă dezvoltarea metodelor de predicție a consumului de energie, care oferă o viziune optimistă asupra

comportamentului sistemelor HVAC. Modelele de predicție utilizează date istorice, informații meteorologice sau variabile operaționale pentru a estima consumul viitor, permitând astfel luarea unor decizii pro-active în gestionarea energiei. Folosirea rețelelor neuronale, algoritmilor de regresie, pădurilor aleatorii și a tehnicii hibride s-a demonstrat a fi eficientă în multe articole, fiecare contribuind semnificativ la creșterea preciziei predicțiilor și la adaptabilitatea la diferite situații.

Un exemplu relevant este prezentat de Wang et al. [22], care sugerează o arhitectură LSTM concepută pentru a surprinde sezonalitatea din consumul de energie. Modelul LSTM (Long Short-Term Memory) este recunoscut pentru capacitatea sa de a memora și asocia secvențe lungi de date, ceea ce îl face ideal pentru identificarea și învățarea tiparelor periodice specifice consumului energetic. În cadrul acestui articol științific, autori au reușit să anticipeze cu exactitate modificările de consum pe termen lung, chiar și în contextul unor variații semnificative. Această performanță face ca LSTM să fie perfect pentru aplicații precum gestionarea maximelor de sarcină sau programarea inteligentă a consumului, în funcție de prețurile variabile ale energiei.

Afzal et al. [23] realizează o analiză detaliată a arhitecturilor LSTM și MLP (Multi-Layer Perceptron), concentrându-se pe optimizarea acestora pentru a obține predicții mai stabile și mai precise. Se folosesc tehnici evolutive, cum ar fi Particle Swarm Optimization (PSO) și algoritmi genetici, pentru ajustarea hiperparametrilor rețelelor. Această abordare combinată facilitează maximizarea performanței și reducerea erorilor de predicție în medii complexe, precum clădirile comerciale cu mulți utilizatori și programe de funcționare variabile. Articolul evidențiază, de asemenea, importanța selecției caracteristicilor (features) și a normalizării datelor, arătând că gestionarea atentă a datelor ajută la o generalizare superioară a modelului.

În ceea ce primește metoda Random Forest, Wang et al. [24] au implementat această tehnică pentru a prezice consumul energetic la nivel orar. Random Forest, recunoscut ca un algoritm de tip ansamblu, utilizează o multitudine de arbori de decizie pentru realizarea predicțiilor și se remarcă prin rezistența sa la fenomenul de overfitting. Studiul menționat evidențiază faptul că acest model nu doar că furnizează estimări precise, dar și oferă o ierarhie a semnificației caracteristicilor, ceea ce este util pentru interpretare. Prin urmare, se pot trage concluzii evidente privind impactul temperaturii exterioare, al umidității sau al programului de utilizare a spațiilor asupra consumului de energie. Prin această abordare, Random Forest se dovedește a fi un instrument valoros pentru prognoza și pentru alegerea deciziilor legate de măsurile de eficiență energetică.

Mocanu et al. [25] oferă o contribuție importantă în domeniul clădirilor inteligente, unde predicția consumului energetic devine un element esențial pentru gestionarea eficientă a resurselor. Studiul sugerează un sistem ce integrează tehnici de învățare automată cu strategii de alocare dinamică a resurselor. Modelul dezvoltat are abilitatea de a prezice perioadele de consum ridicat și de a modifica în mod dinamic livrarea energiei, reducând riscul de suprasolicitare a rețelei și optimizând costurile operaționale. Această abordare devine cu atât mai relevantă în contextul clădirilor care utilizează surse regenerabile și deservesc un număr mare de utilizatori, asigurând un echilibru sustenabil între cerere și ofertă. Pe de altă parte, Homod [26], propune o abordare inovatoare, care îmbină predicția consumului energetic cu strategii avansate de control intelligent. Modelul prezentat este capabil să ajusteze automat parametrii sistemului HVAC, utilizând atât prognozele realiste, cât și condițiile ambientale în timp real. Prin aceasta metodă se obține un echilibru eficient între consumul energetic optimizat și nivelul de confort pentru ocupanți, fiind esențial în contextul clădirilor comerciale cu cerințe dinamice. Din acest studiu poate să reiese relevanța utilizării modelelor predictive nu doar pentru anticiparea consumului, dar și ca instrument de

sprijin pentru deciziile automate de control, ceea ce contribuie la creșterea eficienței operaționale a sistemelor HVAC.

Pe de altă parte, în articolul științific realizat de Grillone [27], se face o analiză comparativă între modelele deterministe și cele bazate pe date, aplicate în predicția consumului și evaluarea scenariilor de renovare. Acest studiu arată ca, deși modelele fizice oferă un grad ridicat de precizie, acestea sunt mai puțin flexibile în adaptarea la situații reale. În schimb, abordările data-driven, în special cele care utilizează algoritmi de învățare profundă, se dovedesc a fi mai adaptabile la variațiile din teren. Această flexibilitate le face mai adecvate pentru sistemele HVAC complicate, unde condițiile de operare pot fluctua considerabil.

În contextul actual al eficienței energetice, optimizarea sistemelor HVAC prin algoritmi metaeuristici reprezintă o direcție de studiu destul de importantă. Acești algoritmi, inspirați din procese naturale, precum algoritmii genetici (GA), optimizarea roilor de particule (PSO), algoritmul furnicilor (ACO) sau controlul predictiv bazat pe model (MPC), sunt utilizati pentru reglarea dinamică a parametrilor sistemelor HVAC. Scopul principal este de a reduce consumul energetic, de a menține confortul termic și de a prelungi durata de viață a echipamentelor, elemente esențiale pentru o gestionare sustenabilă a resurselor și pentru o performanță optimă a sistemului.

Astfel, studiul lui Bamdad et al. [28] analizează utilizarea algoritmului Ant Colony Optimization (ACO) într-un context de optimizare a energiei clădirilor. Algoritmul ACO, inspirat din modul în care furnicile își găsesc traseul optim, are capacitatea de a explora un spațiu de soluții complicat și de a descoperi rute optime pentru reducerea consumului energetic. În acest articol se arată că ACO ajunge rapid la soluții eficiente, spre deosebire de metodele tradiționale, și oferă un beneficiu considerabil în sistemele HVAC unde echilibrul între confort și consum este crucial.

În altă ordine de idei, Ala'raj et al. [29] realizează o analiză detaliată a acestui subiect, sintetizând principalele abordări data-driven utilizate pentru optimizarea sistemelor HVAC. Pe baza acestei analize, se identifică PSO (Particle Swarm Optimization) și GA (Genetic Algorithms) drept cei mai frecvenți utilizatori algoritmi în domeniul. Algoritmii genetici, care se bazează pe principiile selecției naturale, s-au dovedit eficienți în optimizarea globală a strategiilor de control HVAC. În același timp, PSO, inspirat din comportamentul colectiv al organismelor, se diferențiază prin rapiditatea convergenței și abilitatea de a se adapta la condiții dinamice. Acest articol subliniază că ambele metode sunt adecvate în special pentru sistemele HVAC complexe, care necesită ajustări continue și precise ale variabilelor de control.

Un alt exemplu interesant legat de optimizare este articolul științific realizat de Kirubakaran et al. [30] care propun un model ce îmbină controlul predictiv (MPC) cu algoritmul PSO, o abordare care se evidențiază prin complexitate și eficiență. Controlul predictiv permite anticiparea comportamentului sistemului HVAC, facilitând schimbări pro-active, în timp ce PSO optimizează parametrii de control în vederea reducerii consumului de energie. Această combinație de metode oferă un sistem robust și adaptabil, capabil să mențină confortul termic în condiții variante de operare. Articolul evidențiază eficiența acestei strategii hibride raportat la metodele tradiționale, demonstrând reducerea semnificativă a costurilor energetice și a schimbărilor termice în spațiile climatizate.

Nu în ultimul rând, o aplicație practică relevantă este prezentată de Garces-Jimenez et al. [31], care folosesc date reale provenite dintr-o clădire pentru a evalua performanța algoritmilor de optimizare. Acest studiu se axează pe Particle Swarm Optimization (PSO) și algoritmi genetici, demonstrând capacitatea acestor algoritmi de

a optimiza simultan parametri precum consumul de energie, nivelul de confort și gradul de uzură al echipamentelor. Rezultatele evidențiază faptul ca PSO oferă o explorare eficientă a spațiului de soluții și reușește să se adapteze la variațiile condițiilor de mediu, pe când algoritmul genetic asigură stabilitatea soluțiilor identificate. Articolul științific integrează și principiile MPC în cadrul procesului de optimizare, ajutând la anticiparea evoluției parametrilor sistemului și la reglarea pro-activă a acestora.

În capitolul următor, vor fi prezentate în detaliu fundamentele teoretice care susțin soluția propusă, punând accent pe structura logică a aplicației, modelele de predicție utilizate și algoritmii de optimizare integrați.

## **Capitolul 4. Analiză și fundamentare Teoretică**

Capitolul acesta se concentrează pe explicarea conceptelor teoretice care stau la baza arhitecturii și logicii soluției dezvoltate. Scopul principal este să ofere o perspectivă asupra algoritmilor, metodelor matematice și strategiilor utilizate pentru realizarea obiectivelor aplicației, fără a intra în detalii de implementare. Se va pune accent pe înțelegerea funcțională a fiecărei componente, de la estimarea consumului de referință și analiza relațiilor dintre variabile, până la metodele de predicție și optimizare energetică.

### **4.1. Estimarea consumului de referință**

Estimarea consumului de referință, cunoscută și sub denumirea de *baseline energetic*, reprezintă un element central în analiza consumului de energie. Aceasta poate fi aplicat atât la scară largă, cum ar fi rețelele inteligente (*smart grids*) sau micro-rețelele (*microgrids*), cât și la nivelul individual al clădirilor.

În contextul rețelelor inteligente, această estimare este importantă pentru monitorizarea cererii și implementarea programelor de răspuns la cerere (*demand response*). Astfel, baseline-ul permite o predicție mai precisă a consumului energetic, oferind operatorilor posibilitatea de a ajusta producția și distribuția de energie în timp real.

În cadrul microgrids, unde consumatorii devin tot mai des și producători de energie, devine esențială stabilirea unui consum de referință. Acest lucru este important pentru a menține un echilibru optim între producția locală, stocare și consum. În acest context, baseline-ul devine un parametru dinamic, integrat în sistemele automate de control și decizie, facilitând ajustări în timp real, necesare pentru optimizarea fluxurilor energetice, cât și la managementul eficient al resurselor disponibile.

Această logică se regăsește și în domeniul clădirilor inteligente echipate cu sisteme HVAC, unde baseline-ul energetic devine esențial pentru evaluarea eficienței operaționale și pentru alegerea deciziilor de optimizare. Astfel, trecerea de la o perspectivă macro, centrată pe rețea, la o abordare micro, focalizată pe clădire, presupune ajustarea metodelor de estimare a consumului energetic. Prin urmare, baseline-ul rămâne un reper esențial de comparație pentru evaluarea metodelor de optimizare implementate ulterior.

Importanța baseline-ului energetic vine din rolul său esențial în evaluarea impactului măsurilor de predicție sau optimizare, oferind o vizionare obiectivă asupra analizei abaterilor de consum. Din punct de vedere teoretic, baseline-ul se realizează prin analiza unor date istorice de consum, care sunt selectate pentru a reflecta un comportament normal și constant. O metodă frecvent utilizată este calcularea mediei consumului pentru aceleași zile ale săptămânii din săptămânile precedente, într-un interval de timp destul de apropiat. Această strategie implică faptul că activitatea și condițiile operaționale ale clădirii respectă un anumit tipar săptămânal, iar comparația directă între zile similare garantează o evaluare mai precisă.

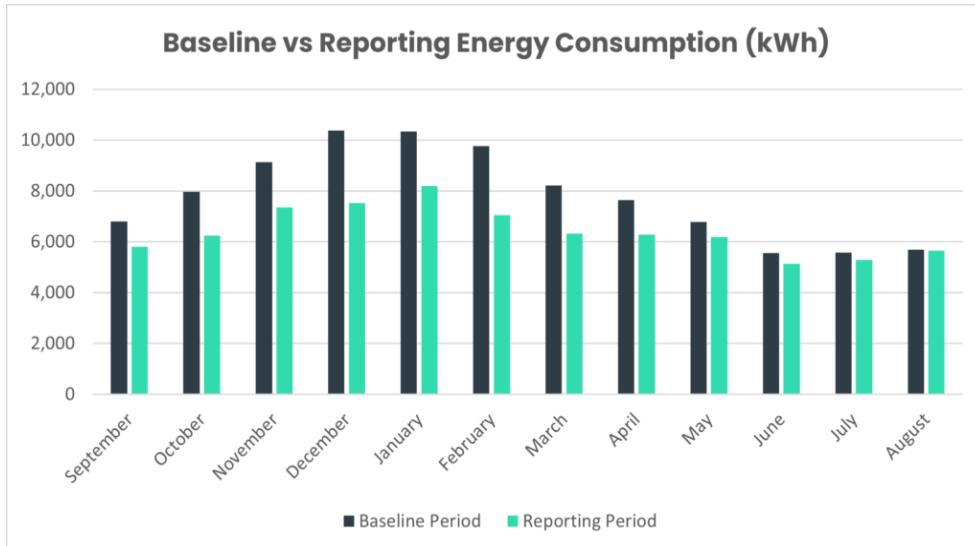


Figura 4.1 Analiză comparativă lunară între consumul real și baseline [33]

În ceea ce privește stabilirea baseline-ului, există diverse metode, de la medii aritmetice și mediane, până la abordări mai complexe bazate pe regresii sau algoritmi de tip Machine Learning. Cu toate acestea, pentru analize comparative, metodele statistice de bază sunt frecvent alese datorită clarității și consistenței lor. Un aspect esențial în definirea baseline-ului constă în alegerea unui interval temporal care să fie adecvat, cât și excluderea anomalieiilor, cum ar fi perioadele de întreținere, sărbătorile legale sau intreruperile neprevăzute, care pot afecta valorile medii și acuratețea analizei.

Într-un sistem complet de management energetic, baseline-ul are o importanță mare atât pentru analiza comparativă, cât și pentru funcționarea în timp real. Acesta nu are rol doar la evaluarea performanței istorice, ci este și un instrument de monitorizare și decizie. Atunci când valorile de consum înregistrate diferă semnificativ de baseline-ul stabilit, sistemul poate declansa automat o analiză a cauzelor sau poate interveni direct, implementând metode de control pentru restabilirea parametrilor la un nivel normal. Astfel, baseline-ul funcționează ca un mecanism dinamic de ajutor de decizie și alertare rapidă, asigurând gestionarea eficientă a resurselor energetice.

Din perspectivă teoretică, baseline-ul reprezintă un punct de referință fundamental în orice proces de optimizare a consumului de energie. Aceasta asigură consistență și comparabilitate, oferind o bază obiectivă pentru interpretarea performanței unui sistem HVAC, iar fără stabilirea unui punct de referință clar, evaluarea și îmbunătățirea eficienței energetice devin dificil de realizat.

## 4.2. Coeficientul de corelație Pearson

Coeficientul de corelație Pearson reprezintă un instrument statistic esențial utilizat pentru a evalua cât de puternică și în ce sens există o relație între două variabile numerice. În analiza performanței energetice a clădirilor inteligente, coeficientul Pearson devine relevant pentru a determina gradul în care anumite variabile statistic influențează consumul de energie al clădirilor. Acest coeficient evidențiază caracteristicile (features) esențiale, oferind o bază solidă pentru alegerile făcute în procesul de construire a modelului.

Din punct de vedere matematic, coeficientul Pearson, notat convențional cu  $r$ , se definește ca raportul dintre covarianța celor două variabile și produsul deviațiilor lor standard. Valoarea obținută se află întotdeauna între -1 și 1, astfel că valoare apropiată de 1 sugerează o corelație pozitiva puternică, ceea ce înseamnă că, pe măsură ce una

dintre variabile crește, și cealaltă tinde să crească. O valoarea aproape de -1 reflectă o corelație negativă puternică, evidențiind o relație inversă între cele două variabile. În final, o valoare apropiată de 0, indică faptul că între cele două variabile nu există o corelație liniară semnificativă.

Tabel 4.1 Valorile coeficientului Pearson

| <b>Intervalul coeficientului Pearson (r)</b> | <b>Interpretarea valorii coeficientului</b> |
|--|---|
| -1.00 ≤ r < -0.80                            | Corelație foarte puternică negativă         |
| -0.80 ≤ r < -0.60                            | Corelație puternică negativă                |
| -0.60 ≤ r < -0.40                            | Corelație moderată negativă                 |
| -0.40 ≤ r < -0.20                            | Corelație slabă negativă                    |
| -0.20 ≤ r < 0.20                             | Corelație foarte slabă sau inexistentă      |
| 0.20 ≤ r < 0.40                              | Corelație slabă pozitivă                    |
| 0.40 ≤ r < 0.60                              | Corelație moderată pozitivă                 |
| 0.60 ≤ r < 0.80                              | Corelație puternică pozitivă                |
| 0.80 ≤ r < 1.00                              | Corelație foarte puternică pozitivă         |

Prin urmare, coeficientul de corelație Pearson este un instrument fundamental atunci când vine vorba de analiza și înțelegerea datelor. Acesta nu se limitează doar la identificarea relațiilor dintre variabile, ci joacă un rol important și în optimizarea modelelor de predicție sau de control energetic. Prin intermediul său, devine mult mai ușor să se detecteze variabilele care influențează în mod direct consumul de energie, ceea ce permite dezvoltarea unor sisteme HVAC atât eficiente cât și adaptate la condițiile reale de funcționare.

### 4.3. Profilul de consum energetic

Profilul consumului energetic constituie o ilustrare temporală detaliată a modului în care o clădire utilizează energia în decursul unei zile, unei săptămâni sau al unei perioade mai lungi. Acest principiu este vital în examinarea clădirilor inteligente, deoarece oferă o perspectivă clară asupra modului în care energia este consumată în funcție de ora din zi, modul de ocupare, condițiile meteorologice și programul de funcționare al echipamentelor.

Atunci când se analizează profilul de consum energetic, cele mai evidente diferențe sunt între clădirile rezidențiale și cele de birouri. În locuințe, consumul atinge valori maxime dimineața devreme și seara, iar în cazul clădirilor de birouri, acesta se manifestă în intervalul orar 9:00 – 18:00, corespunzător programului de lucru. Această variație necesită o analiză separată pentru fiecare tip de clădire, fapt care permite ajustarea mai eficientă a strategiilor de optimizare și un control mai precis asupra acestor sisteme inteligente.

Din perspectivă teoretică, profilul de consum se formează pe baza datelor de consum istoric colectate la intervale regulate, cum ar fi la fiecare oră sau minut, iar aceste valori sunt ulterior agregate și examineate pentru a identifica tipare și anomalii. Un profil bine definit poate indica, de exemplu, dacă o clădire consumă mult în orele de vârf, dacă se observă o eficiență scăzută în momentele de neactivitate sau dacă sistemul HVAC este prea mare pentru cerințele reale.

Acesta nu servește doar ca o simplă reflectare a consumului real, ci funcționează și ca reper important pentru evaluarea performanței energetice. Compararea profilului actual cu un model ideal sau cu unul creat prin modele predictive, facilitează

recunoașterea abaterilor și adaptarea strategiilor de control în timp real. În acest context, profilul devine un instrument dinamic, nu doar explicativ, ci și pentru luare a deciziilor.

În contextul predicției și optimizării, profilul energetic joacă un rol important în folosirea algoritmilor de învățare automată sau de control. Modelele de predicție, precum rețelele neuronale sau regresiile, utilizează aceste profiluri pentru a anticipa comportamentul viitor al sistemului. Mai mult, algoritmii evolutivi și metodele de control predictiv se bazează pe aceste date istorice pentru a ajusta parametrii de funcționare a clădirii în funcție de diferite scenarii anticipate, asigurând astfel eficiența energetică și confortul utilizatorilor.

În clădirile inteligente, senzorii IoT monitorizează constant parametrii de consum și transmit datele către sistemele de management energetic (BEMS). Astfel, profilul de consum poate fi inclus în procesul decizional și poate funcționa ca referință pentru activarea alarmelor, implementarea politicilor de economisire sau detectarea automată a defectelor. De exemplu, o variație neașteptată față de profilul normal de consum ar putea indica atât o problemă la sistemul de ventilație, cât și o ocupare atipică a spațiului, o defecțiune la sistemul de climatizare sau o eroare în reglajul automat al temperaturii. Astfel de abateri pot semnala uși sau ferestre lăsate deschise, consumatori suplimentari activați din greșală ori suprasolicitarea unor echipamente aflate în afara programului normal de funcționare. În plus, o scădere bruscă a consumului poate indica o pană de curent sau chiar o defecțiune majoră în rețea. Identificarea lor permite intervenții rapide, fie prin ajustarea setărilor, fie prin notificarea personalului de mențenanță sau chiar printr-o declanșare automată a unor cazuri de economisire. Un scenariu de economisire s-ar putea referi la reducerea debitului de aer sau chiar la oprirea alimentării în zonele neocupate.

Astfel, profilul de consum energetic nu se limitează doar la rolul de instrument de analiză istorică, ci devine un mecanism dinamic de control și prevenție în contextul gestionării eficiente a unei clădiri.

## **4.4. Algoritmi de predicție energetică**

### **4.4.1. Random Forest (RF)**

Unul dintre cei mai folosiți algoritmi pentru estimarea consumului energetic în clădiri este Random Forest (RF), un model de învățare automată din grupul metodelor *ensemble*, bazat pe structurarea și agregarea mai multor arbori de decizie. Acesta este considerat un model de tip *ensemble* deoarece utilizează mai mulți arbori de decizie antrenați pe subseturi diferite ale datelor. Astfel, modelul nu se bazează pe un singur arbore, ci pe combinarea informațiilor provenite de la întreaga pădure de arbori. Prin urmare, Random Forest a fost propus pentru a spori acuratețea modelelor fundamentale pe arbori simpli, care sunt frecvent vulnerabili la fluctuațiile din date și pot cauza supraînvățare.

Funcționarea teoretică a algoritmului Random Forest implică crearea unui grup de arbori de decizie, fiecare fiind antrenat pe un subset diferit al datelor de antrenament. Aceste subseturi sunt create printr-o metodă denumită bootstrap sampling, adică eșantionare aleatorie cu înlocuire. De asemenea, în fiecare nod de decizie, modelul alege un subset random de caracteristici (*features*) și selectează atributul care asigură cea mai bună separare a datelor pe baza unor criterii precum indicele Gini sau entropia. După construirea fiecărui arbore, Random Forest realizează predicția finală prin agregarea răspunsurilor individuale ale arborilor. În cazul regresiei, relevantă pentru estimarea valorilor continue precum consumul energetic, această agregare presupune calculul mediei valorilor prezise de fiecare arbore. Acest proces contribuie la

dezvoltarea unui model robust, capabil să gestioneze variații semnificative în date și să reducă riscul de overfitting.

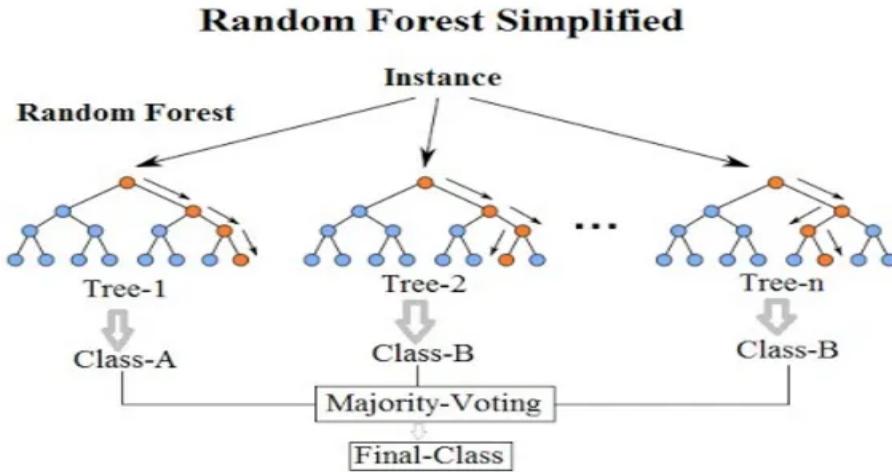


Figura 4.2. Random Forest în Machine Learning [34]

Utilizarea algoritmului Random Forest în domeniul HVAC se justifică prin abilitatea acestuia de a surprinde relații neliniare complexe între variabile precum ora din zi, temperatura exterioară, umiditatea, gradul de ocupare a locului și consumul de energie. De asemenea, metoda oferă o evaluare a relevanței fiecărei variabile de intrare, lucru care se dovedește foarte util în analiza factorilor care afectează consumul și în selecția automată a celor mai relevante caracteristici.

Din perspectivă computațională, Random Forest este eficient și scalabil, adaptându-se cu ușurință la volume mari de date colectate în timp real. În plus, asigură o interpretabilitate mai bună decât modelele „black-box”, ceea ce îl face adecvat pentru situații în care transparența procesului decizional este crucială.

Deși Random Forest este un algoritm robust, acesta nu este lipsit de limitări semnificative. În primul rând, complexitatea modelului, dată de numărul mare de arbori de decizie, poate duce la dificultăți în interpretarea detaliată a rezultatelor. În plus, dacă setul de date conține mult zgomot sau nu prezintă un tipar bine definit, procesul de agregare poate duce la simplificarea soluțiilor. Nu în ultimul rând, în comparație cu rețelele neuronale, performanța sa este limitată în captarea secvențelor temporale a datelor. Astfel, acest lucru îl face mai puțin potrivit pentru cazurile care implică analiza seriilor temporale.

#### 4.4.2. Multi-Layer Perceptron (MLP)

Rețea neuronală de tip Multi-Layer Perceptron (MLP) reprezintă una dintre cele mai recunoscute arhitecturi din domeniul rețelelor neuronale artificiale și joacă un rol esențial în învățarea automată. Structura sa include un strat de intrare, unul sau mai multe straturi ascunse, precum și un strat de ieșire. O caracteristică importantă este faptul că fiecare neuron dintr-un strat este conectat complet cu neuronii din stratul următor, ceea ce permite modelului să surprindă relații complexe și neliniare între datele de intrare și rezultatele dorite.

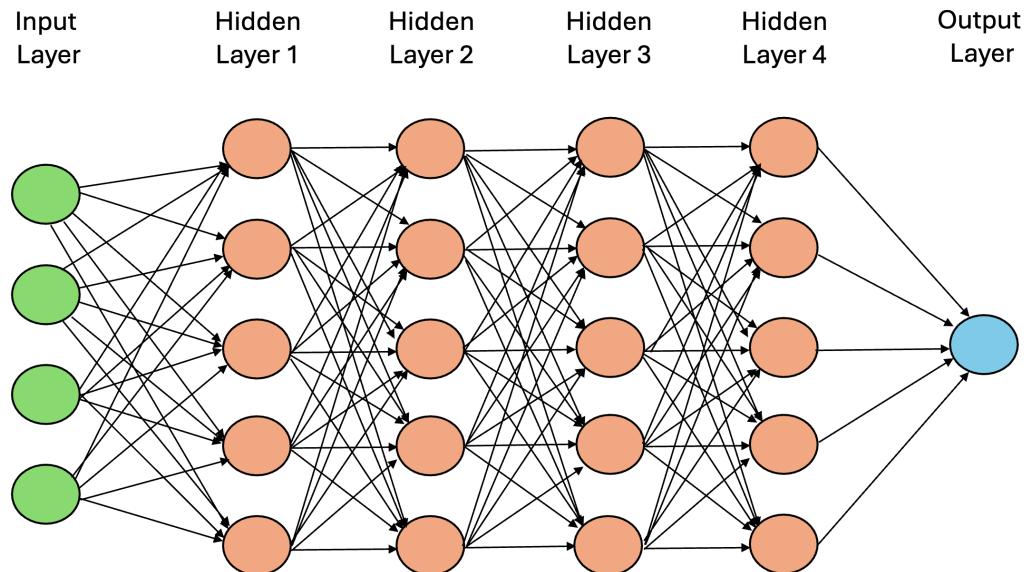


Figura 4.3 Diagrama modelului Multi-Layer Perceptron

În contextul predicției consumului energetic al sistemelor HVAC, MLP-ul este utilizat pentru a anticipa valorile viitoare ale consumului, având la bază seturi de date istorice și factori precum temperatura exterioară, momentul din zi, gradul de ocupare al clădirii sau diferiți parametri operaționali ai sistemului. Aptitudinea modelului de a identifica tipare neliniare, îl face util în circumstanțele în care legăturile dintre variabilele de intrare și consum nu pot fi captate prin modele statistice simple.

Un element important al modelului MLP este alegerea funcției de activare, care trebuie selectată în funcție de specificul fiecărei probleme și de modul în care sunt distribuite datele de intrare. În cazul regresiei energetice, funcții precum *ReLU* (Rectified Linear Unit) sau sigmoid sunt frecvent alese, deoarece introduc neliniaritate în rețea și ajută la identificarea relațiilor complexe dintre variabile. Modelul se antrenează prin algoritmi de tip *backpropagation*, utilizând o funcție de pierdere (*loss function*) selectată în funcție de scopul urmărit. Pentru sarcinile de regresie, se preferă eroarea pătratică medie (*MSE*), deoarece aceasta penalizează mai bine abaterile semnificative față de valorile reale și favorizează stabilitatea procesului de învățare. Pe parcursul antrenării, ajustarea greutăților din rețea se realizează cu ajutorul unor algoritmi de optimizare precum *Adam* sau *SGD* (Stochastic Gradient Descent). Alegerea dintre acești doi optimizatori este influențată atât de viteza de convergență, cât și de acuratețea finală a modelului.

MLP are avantajul unei structuri simple și o capacitate moderată de generalizare atunci când este antrenat corect. În plus, este mai ușor de implementat și ajustat decât modele mai sofisticate, cum ar fi LSTM, ceea ce îl face potrivit pentru aplicații HVAC sau alte situații unde datele nu sunt neapărat secvențiale.

Totuși, rețelele MLP conțin și câteva limitări. Acestea prezintă dificultăți atunci când vine vorba de relațiile temporale din date, deoarece nu dispun de memorie internă sau de mecanisme care transmit informația în timp. În plus, performanța lor depinde mult de selecția și scalarea corectă a caracteristicilor de intrare, ceea ce înseamnă că fără etapa de feature engineering, rezultatele obținute riscă să fie sub așteptări.

Astfel, MLP constituie un instrument puternic și adaptabil pentru estimarea consumului de energie în clădirile inteligente, oferind un raport favorabil între

acuratețe, complexitate computațională și interpretabilitate, în special în situațiile în care datele de intrare sunt constante și variabilele de context sunt clar definite.

#### 4.4.3. Long Short-Term Memory (LSTM)

Rețelele neuronale de tip LSTM (Long Short-Term Memory) reprezintă o evoluție a rețelelor neuronale recurente clasice (RNN), fiind proiectate special pentru a depăși limitările acestora în învățarea dependentelor pe termen lung din datele secențiale. Aceste metode sunt deosebit de eficiente în analiza și predicția consumului energetic în clădirile inteligente, deoarece pot gestiona atât caracteristicile sezoniere, cât și trendurile de consum, modelele repetitive și variațiile complexe care apar în contextul operațional al acestor clădiri. Astfel, capacitatea modelului de a învăța și reține tipare temporale pe perioade îndelungate, îl transformă într-o alegere potrivită pentru analiza evoluției consumului energetic.

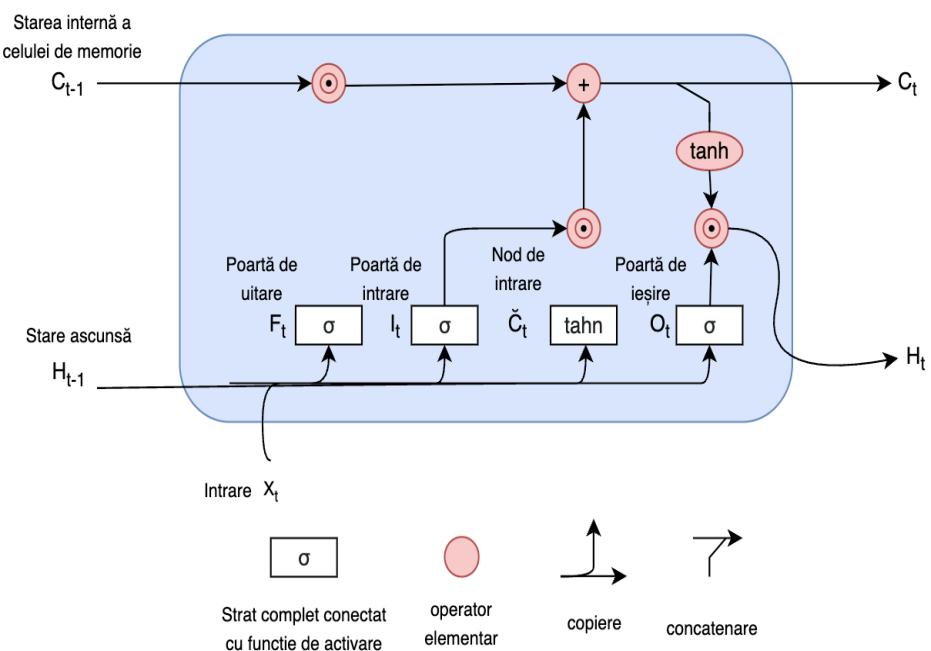


Figura 4.4 Arhitectura modelului LSTM

Din punct de vedere arhitectural, rețelele LSTM sunt construite în jurul unui mecanism intern sofisticat, alcătuit din celule de memorie și diverse tipuri de porți, cum ar fi porțile de intrare, de ieșire și de uitare. Aceste componente fundamentale permit modelului să păstreze informații relevante pe perioade extinse și să eliminate datele care nu mai sunt utile, ceea ce duce la o stabilitate sporită în procesul de antrenare și la îmbunătățirea calității predicțiilor. Astfel, LSTM are capacitatea de a învăța tipare complexe, fie că e vorba de consum zilnic, săptămânal sau sezonier, și poate anticipa cu acuratețe valorile viitoare ale consumului de energie electrică.

Un alt avantaj al LSTM este abilitatea de a procesa direct sevențe de date, eliminând nevoie de etape complexe de preprocesare pentru extragerea trăsăturilor temporale. Acest lucru este deosebit de relevant în cazul clădirilor inteligente, unde datele de intrare pot proveni din surse variate precum consumul orar de energie, temperatura mediului, umiditatea, viteza vântului și alți factori meteorologici cu impact. LSTM reușește să integreze aceste variabile într-un mod eficient din punct de vedere computațional, facilitând modelarea influențelor acestora de-a lungul timpului.

Totodată, performanța unui model LSTM este influențată de calibrarea atentă a hiperparametrilor precum dimensiunea ferestrei de analiză (*lookback*), numărul de neuroni din hidden layer, numărul de epoci de antrenament sau valoarea ratei de învățare. O ajustare corespunzătoare a acestor parametri permite modelului să depășească semnificativ metodele tradiționale în ceea ce privește acuratețea predicțiilor, în special în contexte dinamice și neliniare, cum sunt clădirile inteligente.

Chiar dacă modelul LSTM prezintă avantaje notabile, acesta conține și câteva limitări destul de serioase. Un dezavantaj ar fi că are nevoie de multe resurse de calcul, mai ales în timpul antrenării, ceea ce îl face mai puțin accesibil pentru sistemele cu performanță mai scăzută. Pe lângă asta, modelul este predispus la fenomenul de overfitting atunci când setul de date pentru antrenament nu este suficient de mare sau calitatea datelor nu este atât de bună. Nu în ultimul rând, complexitatea ridicată a arhitecturii complică destul de mult procesul de înțelegere a deciziilor generate de model, astfel că nu trebuie ignorate claritatea și transparența în modul în care sunt generate unele decizii.

În concluzie, LSTM se distinge prin capacitatea sa de a reține și procesa informații secvențiale pe termen lung, ceea ce îl face potrivit pentru estimarea consumului energetic în clădirile inteligente moderne. În acest context, evoluția consumului este puternic influențată de condițiile anterioare și de cererea din trecut, iar LSTM reușește să surprindă aceste dependențe temporale mult mai eficient decât alte modele. Totuși, este important de menționat că LSTM nu este ferit de problemele clasice ale rețelelor neuronale recurente, precum fenomenul de *vanishing gradient* sau *exploding gradient*. Aceste probleme pot apărea mai ales în cazul secvențelor foarte lungi și pot afecta atât stabilitatea, cât și eficiența procesului de antrenare.

## **4.5. Algoritmi de optimizare a consumului energetic**

### **4.5.1. Model Predictive Control (MPC)**

Controlul predictiv pe bază de model (MPC) reprezintă o metodă avansată, destul de apreciată pentru modul în care abordează gestionarea sistemelor dinamice complexe, precum cele HVAC. Acesta utilizează un model matematic pentru a anticipa evoluția viitoare a sistemului, iar apoi, la fiecare etapă, calculează în mod recurrent valorile optime ale variabilelor de control, cu scopul de a minimiza o funcție obiectiv. Acest proces are loc în condițiile respectării restricțiilor impuse sistemului, asigurând astfel un echilibru între performanță și siguranță operațională.

Funcționarea MPC se bazează pe trei etape esențiale. Mai întâi, se elaborează un model predictiv capabil să descrie dinamica termică a clădirii și a echipamentelor HVAC asociate. Acest model poate fi construit fie pe baza analitică, utilizând principii fizice și ecuații diferențiale, fie prin abordări data-driven, cum ar fi rețelele neuronale sau tehniciile de regresie multiplă, în funcție de disponibilitatea și calitatea datelor. În etapa următoare, se formulează o funcție obiectiv, menită să cuantifice performanța sistemului. În contextul clădirilor inteligente, această funcție include termeni care penalizează consumul energetic ridicat, abaterile față de temperatura de referință, precum și ciclurile repetitive de pornire și oprire ale echipamentelor, pentru a asigura atât eficiență, cât și confortul. În final, la fiecare pas de timp, MPC rezolvă o problemă de optimizare pe un anumit orizont de predicție (de exemplu 24 de ore), identifică acțiunile optime pentru intervalul imediat următor, le implementează în sistem, apoi reia întregul proces la pasul următor.

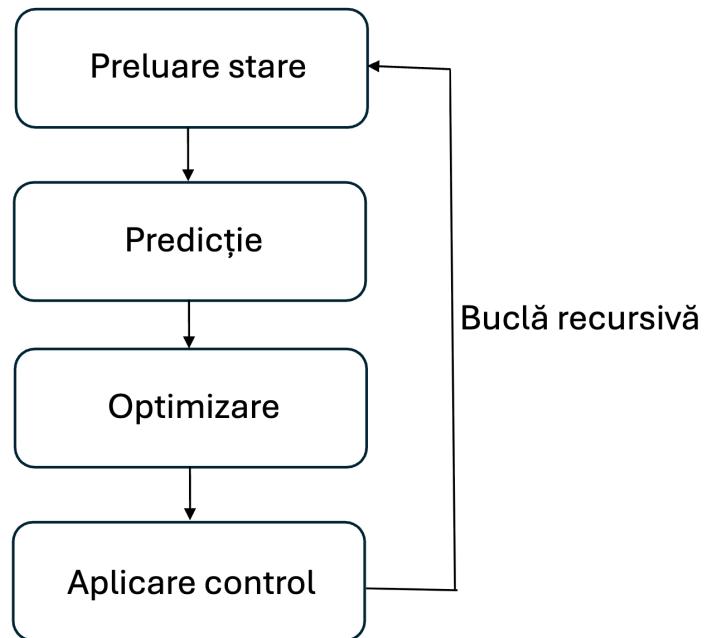


Figura 4.5 Diagrama de funcționare a MPC

Unul dintre principalele beneficii ale MPC este abilitatea de a gestiona constrângeri explicite, cum ar fi limitele temperaturii interioare, fluxului de aer sau puterii maxime ale sistemului HVAC. Prin urmare, MPC poate asigura confortul termic fără a depăși parametrii tehniči ai dispozitivelor. De asemenea, poate include prognoze meteo sau date despre nivelul de ocupare al clădirii pentru a face alegeri informate, diminuând risipa de energie și anticipând cerințele viitoare.

În practică, MPC este aplicat în clădiri comerciale, spitale sau campusuri universitare, unde sistemele HVAC sunt sofisticate, consumul este considerabil, iar fluctuațiile de cerere sunt mari. În aceste situații, aplicarea MPC ajută la economisirea considerabilă a energiei, la scăderea emisiilor de carbon și la extinderea duratei de funcționare a echipamentelor.

#### 4.5.2. Genetic Algorithm (GA)

Algoritmul genetic reprezintă o tehnică de optimizare bazată pe principiile procesului natural de evoluție a organismelor. Din perspectivă teoretică, GA imită selecția naturală, încrucișarea și mutația pentru a găsi soluții optime într-un spațiu de căutare complex. Prin flexibilitatea și robustețea sa, algoritmul este frecvent utilizat în optimizarea multi-obiectivă, inclusiv pentru reglarea și controlul sistemelor HVAC. Astfel de algoritmi sunt adaptabili și performanți pentru dezvoltarea unor probleme dificile, unde metodele clasice se pot dovedi ineficiente.

Un astfel de algoritm începe cu o populație inițială de soluții posibile, fiecare reprezentând un *individ* sau *cromozom*. Acești cromozomi conțin valorile parametrilor care trebuie să fie optimizați, de obicei sub formă de vectori numerici. În fiecare generație, fiecare individ este supus evaluării prin intermediul unei funcții de fitness, care reflectă calitatea soluției în raport cu obiectivele problemei. În contextul unui sistem HVAC, funcția de fitness poate fi definită pe baza consumului total de energie, a nivelului de confort termic sau a gradului de uzură al echipamentelor. Algoritmul selectează indivizii cu cele mai bune valori de fitness, funcționând practic ca un filtru pentru a păstra soluțiile cele mai promitătoare. Ulterior, intervin operatorii genetici,

precum încrucișarea (crossover), care combină informațiile genetice ale doi indivizi pentru a obține urmași, și mutația, menită să introducă modificări aleatorii minore, având rolul de a menține diversitatea populației și de a preveni stagnarea într-un minim local. Procesul se repetă de-a lungul mai multor generații, algoritmul perfecționând continuu calitatea soluțiilor.

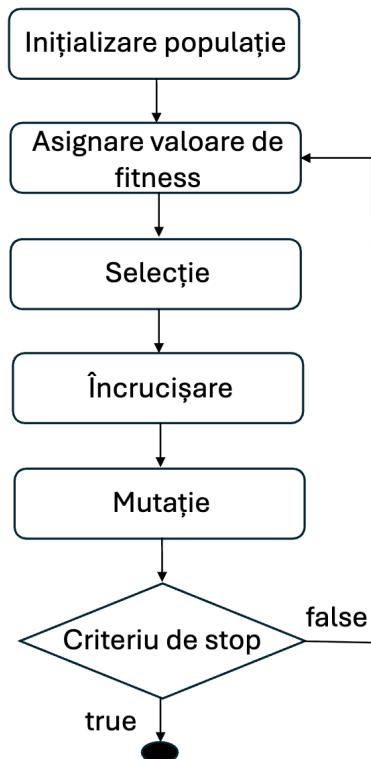


Figura 4.6 Diagrama de funcționare pentru Genetic Algorithm

Un aspect esențial al algoritmilor genetici este abilitatea lor de a gestiona funcții obiectiv complicare, cum ar fi cele neliniare, discontinue sau greu de reprezentat matematic. În cazul clădirilor inteligente, această flexibilitate devine crucială deoarece interacțiunea dintre variabile precum temperatura, viteza vântului și consumul de energie este adesea complicată și dificil de reprezentat printr-o ecuație simplă. Această metodă inspirată din biologie se combină perfect cu structura unui sistem intelligent de management energetic, având un impact semnificativ în reducerea consumului de energie și în promovarea sustenabilității clădirilor.

Pe de altă parte, o problemă semnificativă a acestor algoritmi o reprezintă timpul mare de rulare care este necesar pentru a obține o soluție eficientă. Această problemă apare atunci când spațiul de căutare este foarte mare sau problema presupune un număr ridicat de variabile. În plus, performanța lor depinde mult de alegerea parametrilor precum rata de mutare, dimensiunea populației sau criteriile de selecție, parametrii care previn convergența prematură sau blocarea procesului de optimizare.

#### 4.5.3. Particle Swarm Optimization (PSO)

Algoritmul de optimizare prin roiu de particule (PSO) este inspirat din comportamentul colectiv observat la animale precum păsări sau pești. PSO a devenit o alegere populară pentru optimizarea clădirilor inteligente, deoarece poate identifica eficient soluții în spații de căutare extinse și complexe, acolo unde metodele tradiționale întâmpină dificultăți sau devin ineficiente.

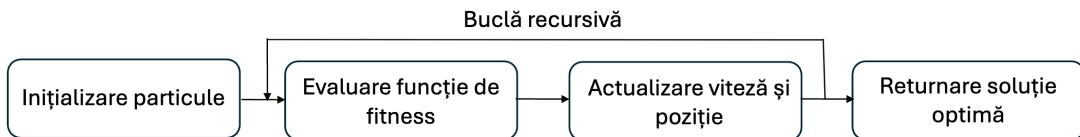


Figura 4.7 Reprezentarea funcționalității algoritmului PSO

Teoretic, PSO implică existența unui ansamblu de particule, fiecare reprezentând o posibilă soluție la problema analizată. Aceste particule explorează spațiul soluțiilor, bazându-se atât pe experiența proprie acumulată, cât și pe rezultatele obținute de celelalte particule din roi. Fiecare particulă este caracterizată de poziție și viteză, iar în fiecare etapă, poziția este actualizată în funcție de cea mai bună soluție identificată de particulă până la momentul respectiv (valoarea optimă personală), precum și de cea mai performantă soluție descoperită la nivelul întregului roi (valoarea optimă globală). Structura de bază a algoritmului presupune actualizarea vitezei și poziției particulelor, gestionarea acestora cu ajutorul unor factori, cum ar fi inerția și accelerarea, precum și evaluarea continuă a funcției de fitness. Pentru optimizarea clădirilor inteligente, funcția de fitness poate include termeni care penalizează consumul energetic excesiv, abaterea de la temperatura de referință sau alterările frecvente ale echipamentelor.

PSO ieșe în evidență prin faptul că, spre deosebire de algoritmii genetici, nu presupune operații precum încrucișarea și mutația, ceea ce îl face considerabil mai eficient din punct de vedere computațional. Cu toate acestea, particulele din algoritm continuă să exploreze intelligent spațiul de căutare, adaptându-se constant către zonele promițătoare ale soluțiilor. Această capacitate de adaptare duce la o viteză ridicată de convergență, ceea ce îl face esențial pentru aplicații HVAC în timp real, unde un răspuns prompt este esențial. De asemenea, PSO este remarcabil și prin adaptabilitatea sa, putând fi aplicat atât în cazuri de optimizare mono-obiectiv, cât și multi-obiectiv. În plus, prin utilizarea variantelor precum PSO cu topologie de vecinătate sau PSO adaptiv, se pot atinge performanțe superioare în funcție de necesitățile specifice ale sistemului.

O limitare importantă a algoritmului PSO o reprezintă convergența sa prematură, un fenomen care se regăsește în cazul problemelor cu funcțiilor obiectiv complexe sau cu multiple minime locale. De asemenea, performanța algoritmului este influențată de alegerea hiperparametrilor, cum ar fi ponderile pentru atracția individuală și globală, care implică o reglare riguroasă pentru a obține rezultate stabilă.

Prin urmare, PSO reprezintă o opțiune solidă în optimizarea energetică, având capacitatea de a identifica soluții aproape optime într-o manieră eficientă și adaptabilă. Împreună cu strategii de control precum MPC, algoritmul PSO contribuie semnificativ la creșterea performanței energetice și la asigurarea confortului în clădirile inteligente.

#### 4.5.4. Ant Colony Optimization (ACO)

Algoritmul de optimizare inspirat din coloanile de furnici, reprezintă o abordare metaeuristică fundamentală pe studiul comportamentului natural al furnicilor în procesul de căutare a hranei. Studii biologice au demonstrat că furnicile identifică drumul cel mai scurt între cuib și sursa de hrana prin depunerea de feromoni de-a lungul traseelor parcuse. Aceste substanțe chimice servesc drept semnale pentru celelalte furnici, determinându-le să urmeze rutele cu cea mai mare concentrație de feromoni, care, în timp, corespund celor mai eficiente și scurte drumuri. Această strategie evolutivă a stat la baza dezvoltării algoritmilor de optimizare, care s-au dovedit extrem de eficienți în ajustarea clădirilor inteligente.

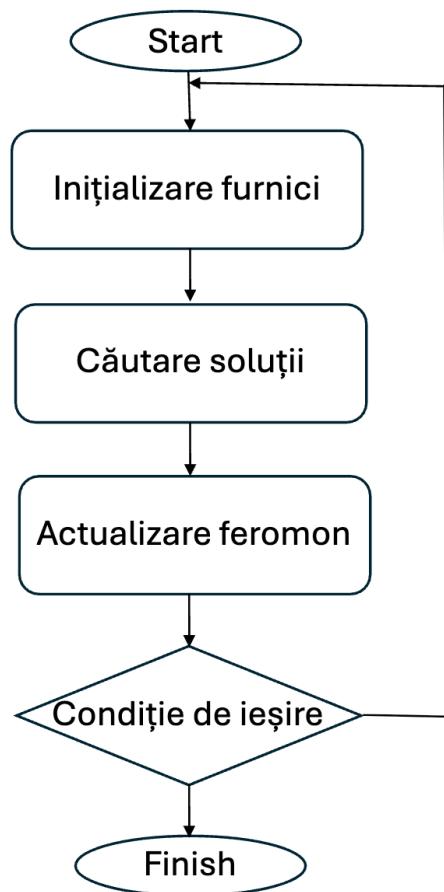


Figura 4.8 Diagrama algoritmului de optimizare ACO

Algoritmul ACO utilizează o populație de agenți artificiali, denumiți *furnici*, care explorează spațiul de soluții într-o manieră întâmplătoare. Fiecare furnică construiește incremental o soluție la fiecare iteratăie, alegând următoarea stare posibilă pe baza a doi factori principali care sunt nivelul de feromon acumulat pe o anumită tranziție și o funcție de calcul locală, cum ar fi costul sau distanța asociată acelei mutări. După ce toate furnicile își finalizează traseele și își construiesc soluțiile, are loc etapa de actualizare a feromonilor. În această fază, soluțiile cele mai performante din iteratăia curentă contribuie la consolidarea rutelor eficiente prin creșterea nivelului de feromon, în timp ce rutele mai puțin eficiente pierd din cantitatea de feromon acumulată, proces

cunoscut sub denumirea de evaporare. Astfel, algoritmul tinde să favorizeze soluțiile optime în iterațiile următoare, optimizând progresiv rezultatul obținut.

În contextul optimizării consumului clădirilor inteligente, fiecare furnică poate fi privită ca un agent care ia decizii legate de reglarea temperaturii, a debitului de aer sau de pornirea și oprirea anumitor componente. Funcția obiectiv, formulată de regulă ca un echilibru între consumul energetic și confortul termic, orientează aceste *furnici* către identificarea unor soluții eficiente. În această analogie, feromonii reflectă eficiența deciziilor adoptate în iterațiile anterioare, facilitând procesul de învățare colectivă. Această abordare se dovedește deosebit de utilă în situații cu multe combinații posibile de acțiuni, acolo unde metodele deterministe nu sunt aplicabile sau nu pot identifica soluția optimă în timp util.

Algoritmul se remarcă prin capacitatea de a explora eficient spațiile de soluții extinse, evitând blocarea în minime locale, mai ales atunci când sunt reguli dinamice de selecție. În plus, datorită naturii sale distribuite, ACO poate fi realizat eficient în paralel, îmbunătățind viteze de convergență și scalabilitatea în aplicații complexe HVAC.

Pe de altă parte, limitarea algoritmului constă în sensibilitatea sa la alegerea parametrilor de ajustare a nivelului de feromoni, care pot influența calitatea soluției. Dacă acești parametri nu sunt ajustați corespunzător, există riscul unei convergențe rapide către o soluție ineficientă și astfel, se poate reduce capacitatea de a exploata alte trasee care au potențial mai bun.

Prin urmare, algoritmul prezintă o soluție eficientă pentru îmbunătățirea funcționării clădirilor inteligente, având capacitatea de a ajusta în mod dinamic cerința de economisire a energiei cu păstrarea condițiilor de confort, în situații în care spațiul decizional este foarte extins și conține multiple interdependențe.

#### **4.6. Cerințe funcționale și non-funcționale**

Cerințele funcționale și non-funcționale reprezintă două categorii importante din proiectarea unei aplicații software. Cerințele funcționale (*Tabel 4.2*) descriu exact ce trebuie să realizeze aplicația, iar cerințele non-funcționale (*Tabel 4.3*) se referă la condițiile de calitate pe care le are sistemul. Astfel, ambele tipuri de cerințe sunt importante pentru a asigura o funcționare corectă și eficientă a aplicației propuse.

| Denumire               | Descriere   |
|------------------------|---|
| Preprocesarea datelor  | Sistemul trebuie să preia și să transforme datele brute din fișiere pentru a fi potrivite pentru implementările ulterioare. |
| Generarea predicțiilor | Aplicația trebuie să genereze consumul estimat într-o zi pentru o clădire selectată.  |
| Calculul baseline-ului | Sistemul trebuie să calculeze consumul de referință pe baza istoricului de consum din zile similare anterioare .            |
| Optimizarea consumului | Aplicația trebuie să aplique algoritmi de optimizare pentru a reduce consumul prezis pe baza profilului de consum.          |
| API-uri REST           | Backend-ul trebuie să expună endpoint-uri pentru a comunica cu frontend-ul și pentru a furniza datele necesare.             |

|                                |   |
|--------------------------------|---|
| Stocarea în baza de date       | Toate datele importante trebuie salvate în PostgreSQL pentru a fi accesate ulterior prin intermediul API-urilor.                  |
| Vizualizarea grafică a datelor | Utilizatorul poate vizualiza în frontend grafice comparative între valorile reale, baseline, predicții și consumurile optimizate. |

Tabel 4.2 Cerințele funcționale ale aplicației

| Denumire                | Descriere  |
|-------------------------|--|
| Scalabilitatea          | Sistemul trebuie să permită adăugarea de clădiri sau date noi fără a schimba arhitectura existentă.                                  |
| Disponibilitatea        | Sistemul trebuie să fie disponibil în orice moment pentru utilizarea într-un mediu local.  |
| Responsivitatea         | Interfața web trebuie să răspund rapid la interacțiunile utilizatorului, fără a avea întârzieri.                                     |
| Ușurința în utilizare   | Interfața trebuie să fie prietenoasă și intuitivă, pentru a putea fi folosită și de utilizatori fără cunoștințe tehnice.             |
| Documentarea API-urilor | Toate rutele backend-ului sunt documentate, pentru a permite testarea interactivă și înțelegerea rapidă a funcționalităților propuse |

Tabel 4.3 Cerințele non-funcționale ale aplicației

## 4.7. Tehnologii

### 4.7.1. Python

Python este un limbaj de programare de referință în mediul academic și profesional. Prin natura sa orientată pe obiect, dar și datorită sintaxei clare, Python ajută atât procesul de învățare, cât și dezvoltarea aplicațiilor complexe. Aceasta permite abordări procedurale, funcționale sau orientate pe obiect, oferind o flexibilitate mare în crearea soluțiilor software.

În domeniul inteligenței artificiale, Python s-a evidențiat prin varietatea bibliotecilor științifice. Astfel, *NumPy* și *Pandas* se folosesc la manipularea eficientă a datelor, în timp ce *Matplotlib* și *Seaborn* oferă soluții pentru vizualizarea acestora. Pentru dezvoltarea și implementarea modelelor clasice de Machine Learning, biblioteca *Scikit-learn* este foarte importantă, în schimb, pentru rețele neuronale avansate, *PyTorch* are un rol esențial, deoarece oferă un suport pentru construirea, antrenarea și evaluarea unor modele complexe precum LSTM sau MLP.

În contextul optimizării consumului de energie pentru clădirile inteligente, Python se dovedește eficient prin permiterea procesării datelor istorice, dezvoltarea și antrenarea modelelor de predicție, aplicarea unor algoritmi de optimizare, precum și generarea de reprezentări grafice. Biblioteci precum *PyTorch* permit integrarea modelelor LSTM, în timp ce *SciPy* ajută la implementarea algoritmilor inspirați din natură.

### 4.7.2. Flask

Flask reprezintă un micro-framework utilizat pentru dezvoltarea aplicațiilor web în limbaj Python, având abordări minimaliste. Deși oferă un set de funcționalități de bază, acesta permite dezvoltatorilor să extindă aplicațiile prin adăugarea de extensii

doar când este necesar, ceea ce contribuie la flexibilitatea și scalabilitatea proiectelor. Din punct de vedere arhitectural, Flask implementează modelul *WSGI* (Web Server Gateway Interface), ajutând la definirea rutelor HTTP, gestionarea cererilor și răspunsurilor, serializarea datelor, precum și integrarea eficientă cu modele de învățare automată sau baze de date.

În contextul dezvoltării unui API, Flask funcționează ca un intermediar între interfața grafică și logica de procesare, expunând endpoint-uri accesibile frontend-ului sau altor utilizatori. Această separare logică între componente favorizează modularitatea și simplifică atât testarea, cât și extinderea ulterioară a aplicației.

#### 4.7.3. Flasgger

Flasgger este o extensie pentru framework-ul Flask, concepută pentru a facilita integrarea cu Swagger, care reprezintă un set de instrumente pentru documentarea și testarea interactivă a API-urilor REST. Utilizând Flasgger, dezvoltatorii pot adnota metodele definite în Flask cu metadate relevante, detaliind structura cererilor, răspunsurilor, precum și tipurile de date implicate, ceea ce conduce la generarea automată a unei interfețe Swagger UI, accesibilă direct din browser.

Documentarea automată reprezintă un instrument esențial atât pentru dezvoltatori, cât și pentru utilizatorii aplicației, facilitând o înțelegere clară și o utilizare corectă a funcționalităților oferite de API. Acest tip de documentație contribuie semnificativ la creșterea transparenței, simplifică procesul de testare și permite o interoperabilitate eficientă cu alte sisteme software.

#### 4.7.4. PostgreSQL

PostgreSQL reprezintă una dintre cele mai apreciate soluții open-source pentru gestionarea bazelor de date relaționale. Este un SGDB care respectă principiile ACID (atomicitate, consistență, izolare, durabilitate), ceea ce îl recomandă pentru aplicații importante în care integritatea datelor este crucială. Din punct de vedere teoretic, PostgreSQL se remarcă printr-un model relațional solid, facilitând stocarea datelor structurate și operarea cu agregări complexe, proceduri, tranzacții sau indexări avansate.

Mai mult, suportul pentru date personalizate, integrarea cu JSON și extensiile precum TimescaleDB îi oferă scalabilitate și flexibilitate, făcându-l o alegere adecvată pentru aplicațiile moderne din sfera IoT și clădirilor inteligente.

#### 4.7.5. React

React este o bibliotecă specializată pentru dezvoltarea interfețelor grafice de utilizator. Una dintre principalele sale caracteristici constă în abordarea pe componente, care permite organizarea logică și modulară a codului. React oferă dezvoltatorilor un cadru intuitiv pentru a construi aplicații frontend, promovând reutilizarea componentelor și o separare clară între logică și prezentare. Aceasta poate fi utilizat pentru a prezenta în timp real grafice, formulare și tabele, ajutând totodată comunicarea eficientă cu backend-ul prin cereri HTTP sau WebSocket-uri.

Astfel, React se dovedește a fi un instrument robust pentru dezvoltarea interfețelor moderne, interactive și eficiente.

## Capitolul 5. Proiectare de detaliu și implementare

Capitolul acesta are rolul de a descrie detaliat etapele prin care a fost proiectată și implementată aplicația destinată optimizării energiei în clădirile dotate cu sisteme HVAC. Obiectivul principal constă în prezentarea clară a funcționalităților esențiale, a structurii modulare a codului și a tehnologiilor utilizate. Se vor parcurge componentele fundamentale ale aplicației, de la calculul valorilor de referință (*baseline*), continuând cu predicția consumului energetic, metodele de optimizare aplicate, precum și integrarea backend-ului cu interfața vizuală.

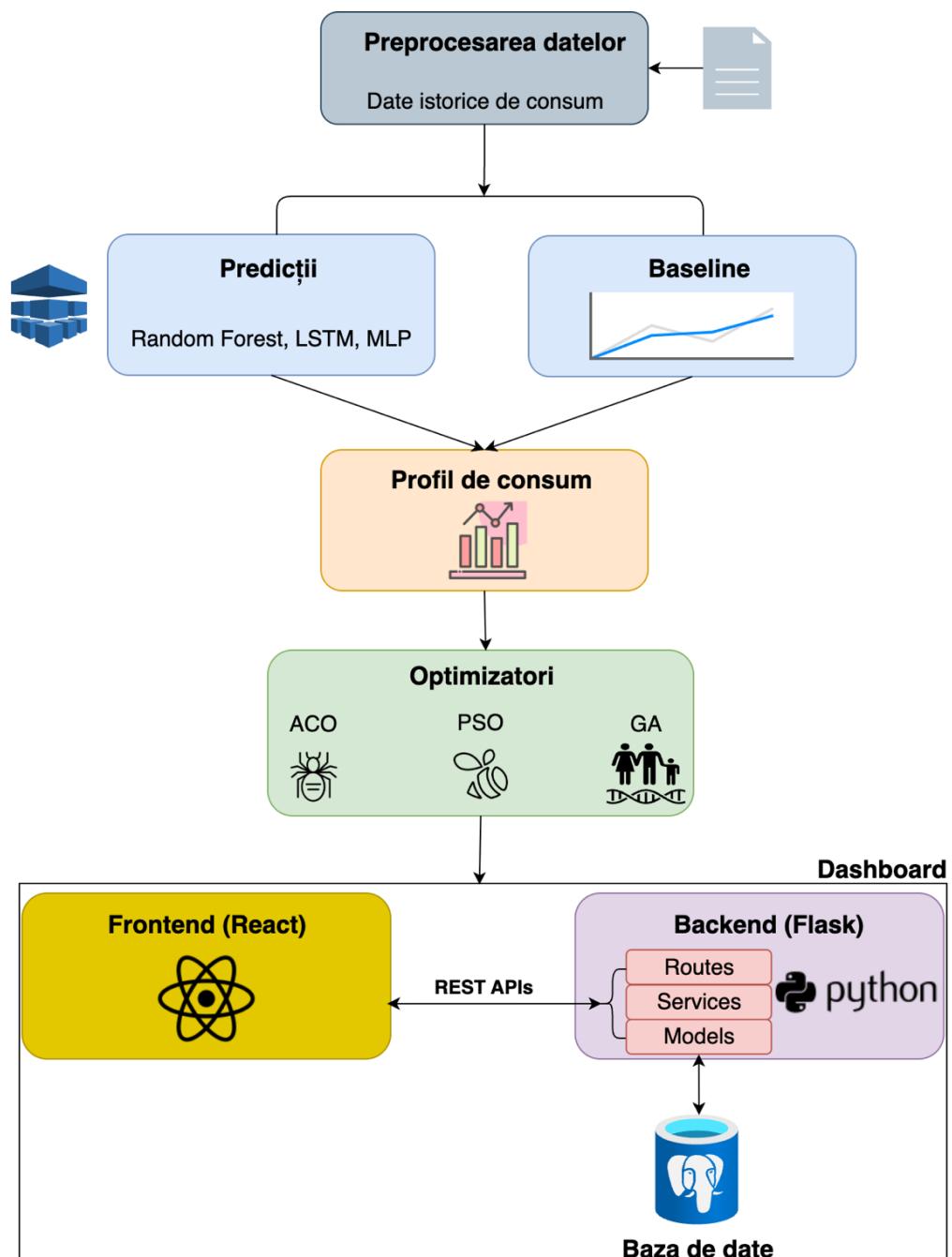


Figura 5.1 Arhitectura conceptuală a aplicației

Proiectul prezentat a fost implementat utilizând o arhitectură full-stack, cu partea de backend dezvoltată în Python, folosind framework-ul Flask, iar frontend-ul implementat în React. S-au folosit câteva biblioteci esențiale din cadrul Python pentru prelucrarea datelor, realizarea predicțiilor și aplicarea tehniciilor de optimizare. Pentru manipularea și procesarea datelor au fost folosite în principal bibliotecile *pandas* și *numpy*, care sunt considerate standard în acest domeniu. Modelele de predicție, precum Random Forest MLP sau LSTM, au fost dezvoltate cu ajutorul bibliotecii *torch*, iar vizualizarea rezultatelor a fost facilitată de *matplotlib*, care permite generarea de grafice relevante.

În continuare, voi detalia fiecare componentă, subliniind funcționalitățile implementate, logica de procesare și integrarea acesteia în sistem.

## 5.1. Calculul baseline-ului

În cadrul analizei energetice este fundamentală stabilirea unui consum de referință, cunoscut drept *baseline*. Acest reper servește ca bază pentru orice comparație între consumul adevărat, cel estimat sau rezultatele obținute în urma optimizărilor. Practic, baseline-ul reprezintă punctul obiectiv la care se raportează performanța energetică a unei clădiri, oferind criterii clare pentru evaluarea eficienței diferitelor strategii de implementare. În cadrul aplicației, metodologia utilizată pentru calculul baseline-ului este centrată pe sezonalitate și pe comportamente recurente la nivel săptămânal. Abordarea pornește de la o ipoteză cum că consumul energetic al clădirilor prezintă tipare repetitive, influențate de ziua săptămânii, intervalele orare de funcționare și anumite rutine ale utilizatorilor spațiului interior. Astfel, pentru a estima consumul considerat normal într-o anumită zi, se analizează datele aferente aceleiași ore din aceeași zi a săptămânii, înregistrate în săptămânile anterioare. Prin agregarea și compararea acestor serii temporale, se obține o referință relevantă pentru evaluarea consumului actual raportat la comportamentul energetic pentru ziua respectivă.

Procesul de calcul presupune mai multe etape diferite. În primul rând, se încarcă setul de date privind consumul energetic, colectat la nivel orar, iar apoi se realizează o filtrare pentru a selecta doar datele aferente clădirii de interes. Ulterior, sunt extrase valorile care corespund aceleiași zile din săptămână ca data analizată, spre exemplu ultimele 7 zile de miercuri anterioare. În mod obișnuit, se aleg cele mai recente 7 zile similare disponibile, menținând logica temporală și evitând includerea datelor din viitor față de momentul analizat. În urma acestei selecții, pentru fiecare dintre cele 7 zile, sunt preluate valorile orare de consum, rezultând astfel 7 serii a către 24 de puncte, corespunzătoare fiecărei ore din zi. Aceste serii sunt apoi verificate pentru a fi complete deoarece este absolut necesar ca fiecare zi să conțină setul integral de 24 de valori pentru ca media obținută să fie relevantă din punct de vedere statistic. În cazul în care anumite zile nu au toate cele 24 de valori orare, se încearcă mai întâi completarea valorilor lipsă, folosind date din zile asemănătoare din trecut. Dacă nici aşa nu pot fi recuperate, atunci se utilizează valorile medii pe oră din săptămâna precedentă, asigurând astfel un set de date complet pentru calculul valorii de referință.

Ulterior, pentru fiecare oră din intervalul 0-23, se calculează media aritmetică a valorilor corespunzătoare din cele 7 zile selectate. Astfel, rezultă un set de 24 de valori, care constituie baseline-ul pentru data analizată, fiecare reprezentând consumul mediu estimat pentru ora respectivă din zi.

La final, rezultatul este organizat într-un tabel, fiecare rând reprezentând o anumită oră, iar coloana principală conține valoarea estimată de referință, notată de obicei cu  $B(t)$ , unde  $t$  este ora respectivă. Acest tabel este folosit ulterior ca suport în

aplicație, fiind utilizat atât pentru generarea graficelor comparative, cât și ca punct de referință pentru evaluarea erorilor de predicție și a eficienței optimizatorilor implementați.

Prin aplicarea acestei metode care ia în considerare sezonalitatea și recurența, se poate stabili un baseline robust și adaptat fiecărei clădiri în parte. Această abordare oferă un cadru obiectiv pentru compararea comportamentului energetic, fie că vorbim despre date reale sau valori estimate. În acest mod, se poate fundamenta mai bine decizia privind intervențiile necesare pentru optimizarea consumului de energie, evitând astfel evaluările superficiale sau bazate pe presupuneri.

## **5.2. Predicția consumului energetic**

O componentă esențială a aplicației constă în predicția consumului de energie pentru clădirile cu sisteme HVAC, realizată prin utilizarea unor algoritmi de învățare automată. Obiectivul acestei etape este de a anticipa valorile viitoare ale consumului orar pentru o anumită zi, bazându-se pe date istorice și informații meteorologice relevante. Această capacitate de predicție permite implementarea unor decizii proactive de optimizare, adaptate contextului operațional specific fiecărei clădiri. Modelele dezvoltate sunt capabile să estimeze consumul pentru un interval de 24 de ore, utilizând fie caracteristici statistice diferite, fie în combinație cu variabile meteorologice identificate ca fiind relevante prin analize de corelație.

După realizarea selecției caracteristicilor meteorologice relevante utilizând coeficientul de corelație, etapa următoare din implementare constă în dezvoltarea unor modele predictive. Aceste modele au rolul de a estima consumul orar de energie pentru următoarele 24 de ore ale unei zile selectate, luând în considerare atât contextul temporal, cât și condițiile meteorologice asociate. Predicțiile sunt generate zilnic, fiecare model fiind conceput să surprindă un anumit tipar de consum pentru fiecare clădire, pe baza istoricului de date și a variabilelor externe relevante. În cadrul acestui proiect au fost implementate și evaluate trei modele distincte, cum ar fi Random Forest, MLP și LSTM. Fiecare dintre aceste modele a fost antrenat pentru a realiza predicții detaliate, orare, privind consumul energetic pe parcursul unei zile, utilizând caracteristicile extrase anterior.

Înainte de a dezvolta modelele de predicție care să prezică următoarele 24 de ore, s-a încercat o abordare alternativă, anume antrenarea modelelor pentru predicții orare. Practic, obiectivul era estimarea consumului de energie pentru ora următoare, utilizând contextul recent. Această strategie a presupus o arhitectură a modelelor mult mai simplificată și un set de date structurat sub forma unor ferestre temporale mobile, construite din ultimele ore disponibile.

Pentru fiecare clădire analizată, datele energetice au fost procesate astfel încât, la fiecare moment orar, să fie generate variabile care reflectă comportamentul consumului din trecutul apropiat, mai exact ultimele 3 ore. Modelele aveau ca sarcină anticiparea valorii consumului energetic din ora imediat următoare, folosind exclusiv aceste informații contextuale.

Chiar dacă această abordare a oferit rezultate decente pentru predicțiile pe termen scurt, realitatea este că nu a reușit să surprindă cu adevărat complexitatea tiparelor zilnice din consumul energetic. Limitarea a constat în faptul că fiecare estimare s-a bazat exclusiv pe contextul imediat anterior, ignorând imaginea de ansamblu a întregii zile. Din această cauză, s-a trecut la predicția pe 24 de ore, ceea ce a permis modelelor să analizeze structura completă a ciclului zilnic și a adus o creștere semnificativă atât pentru acuratețe, cât și pentru stabilitatea predicțiilor.

### 5.2.1. Implementarea modelului Random Forest

În cadrul acestei implementări, pentru fiecare clădire a fost construit un set detaliat de trăsături temporale, precum și indicatorul de weekend. Acest set a fost completat cu informații sezoniere, definite în funcție de lună, și extins prin aplicarea codificării one-hot atât pentru ziua săptămânii, cât și pentru anotimp. În plus, au fost introduse 72 de lag-uri de consum care oferă modelului contextul temporal necesar pentru identificarea și învățarea tiparelor energetice zilnice.

Implementarea începe cu încărcarea datelor și separarea lor în variabile de intrare ( $X$ ) și etichete ( $y$ ), unde etichetele reflectă consumul energetic estimat pentru următoarele 24 de ore. În cadrul scriptului, am implementat un filtru care selectează exclusiv momentele corespunzătoare orei 0, adică începutul fiecărei zile. Această abordare garantează că fiecare instanță folosită pentru antrenare corespunde unei zile complete, asigurând totodată includerea celor 24 de valori de predicție asociate. Întregul set de date a fost împărțit în trei subseturi: 80% pentru antrenare, 10% pentru validare și 10% pentru testare. Această împărțire s-a realizat fără amestecarea datelor, pentru a păstra integritatea secvențială specifică seriilor temporale.

Modelul Random Forest este construit folosind clasa *RandomForestRegressor* din biblioteca *sklearn.ensemble* și a fost configurat cu 200 de arbori de decizie și o adâncime maximă stabilă la 10 niveluri, menținând valoarea *random\_state* la 42 pentru a garanta reproductibilitatea rezultatelor. După antrenare, performanța modelului a fost evaluată utilizând metrii precum eroarea medie pătratică (MSE), eroarea medie absolută (MAE), coeficientul de determinare  $R^2$  și SMAPE, exprimat procentual, pentru a reflecta precizia relativă a predicțiilor. Pe urmă, pentru fiecare clădire analizată, atât predicțiile, cât și valorile reale au fost stocate într-un fișier, fiind generate automat și grafice comparative care evidențiază acuratețea predicțiilor pe o perioadă anume.

Modelul Random Forest a oferit rezultate bune în estimarea consumului zilnic de energie, deși nu a reușit să egaleze performanțele modelelor neuronale evaluate ulterior. Totuși, rămâne o opțiune calitativă datorită structurii sale simple și a faptului că poate fi interpretat cu ușurință.

### 5.2.2. Implementarea modelului Multi-Layer Perceptron

Implementarea modeleului MLP s-a realizat într-un mod structurat, parcurgând toate etapele fundamentale ale procesului: preprocesarea datelor, definirea arhitecturii modelului, faza de antrenare și evaluarea performanței.

În primul rând am declarat parametrii de configurare, cum ar fi *LOOKBACK\_DAYS* stabilit la 3 zile, corespondător unui interval de 72 de ore utilizate ca input, și *PREDICTION\_HORIZON* setat la 24 de ore, astfel încât modelul să efectueze predicții pentru consumul aferent unei zile întregi. Pe urmă, am creat o funcție *create\_daily\_features* care extrage o serie de caracteristici temporale și sezoniere relevante pentru învățarea modelului. Un aspect esențial al funcției îl reprezintă adăugarea de lag-uri pentru ultimele 72 de ore, în scopul furnizării unui context istoric necesar îmbunătățirii acurateței predicțiilor modelului.

Acest model a fost implementat sub forma unei clase numite *MLPModel* în PyTorch, având o structură formată din trei straturi dense, separate prin funcții de activare *ReLU* pentru a menține non-liniaritatea rețelei. În plus, au fost utilizate straturi de normalizare *BatchNorm*, cu scopul de a stabiliza și accelera procesul de învățare. Arhitectura propusă include un strat de intrare având dimensiunea în funcție de numărul de caracteristici, urmat de un strat ascuns cu 256 de neuroni și un strat de ieșire care

generează 24 de valori, corespunzătoare predicțiilor pentru fiecare oră din zi. Setul de date a fost normalizat utilizând *MinMaxScaler* atât pentru datele de intrare, cât și pentru cele de ieșire. Această procedură aduce toate valorile în același interval, ajutând astfel procesul de învățare al rețelei și asigurând o convergență mai eficientă a modelului.

Pentru a antrena modelul în PyTorch, am creat o clasă *TimeSeriesDataset*, derivată din *Dataset*, care transformă datele X și y în tensori compatibil cu modelul. Pe urmă am împărțit datele în trei subseturi: antrenare (80%), validare (10%) și testare (10%), fără amestecarea acestora (*shuffle* = False). Modelul a fost antrenat timp de 200 de epoci folosind optimizatorul *Adam* cu o rată de învățare de 0.0001, iar funcția de pierdere utilizată a fost eroarea pătratică medie (*MSELoss*). După finalizarea procesului de antrenare, modelul a fost evaluat pe setul de testare. Rezultatele au fost readuse la valorile reale folosind *scaler-ul*, după care au fost calculate mai multe metrii relevante pentru performanța modelului, precum eroarea pătratică medie (MSE), eroarea absolută medie (MAE), coeficientul de determinare  $R^2$  și eroarea procentuală simetrică medie (SMAPE). Aceste valori oferă o imagine detaliată asupra acurateței și robusteții modelului antrenat.

În final, rezultatele obținute au fost salvate într-un tabel din baza de date, pentru fiecare clădire, iar pentru perioada de testare a fost generat un grafic comparativ între valorile reale și cele prezise.

Deși modelul MLP a demonstrat o anumită eficiență în estimarea consumului de energie și a reușit să identifice unele tipare din date, performanța sa nu s-a ridicat la nivelul modelului LSTM. În cele ce urmează voi arăta cum LSTM captează mult mai bine dependențele temporale pe termen lung, ceea ce duce la predicții semnificativ mai apropiate de valorile reale.

### 5.2.3. Implementarea modelului LSTM

Modelul LSTM (Long Short-Term Memory) a fost selectat datorită eficienței sale în procesarea datelor secvențiale, fiind recunoscut pentru capacitatea de a surprinde modele temporale complexe. În aplicația propusă, modelul a fost implementat pentru a estima consumul energetic orar pe un interval de 24 de ore, folosind atât istoricul recent de consum, cât și datele meteorologice relevante.

Datele referitoare la consumul energetic și condițiile meteorologice au fost inițial agregate, grupate pe baza timestamp-ului, iar apoi, pentru fiecare clădire analizată, s-au generat secvențe care au integrat atât contextul temporal, cât și valorile istorice de consum energetic. În vederea stabilizării numerice și optimizării procesului de antrenare al modelului, datele au fost ulterior prelucrate și scalate utilizând *MinMaxScaler*.

Pentru fiecare clădire, au fost create serii de intervale orare, fiecare exemplu de antrenament reprezentând contextul unei ore, cu scopul de a prezice consumul pentru următoarele 24 de ore. Pe urmă, datele au fost împărțite astfel: 80% pentru antrenare, 10% pentru validare și 10% pentru testare, păstrând integritatea secvenței temporale. Aceste subseturi au fost ulterior încărcate în obiecte *DataLoader*, ajutând antrenarea eficientă în batch-uri de 64.

Modelul LSTM prezentat utilizează o arhitectură cu trei straturi suprapuse, fiecare având 256 de neuroni ascunși. Aceste straturi sunt urmate de un start complet conectat care generează 24 de valori de ieșire, reprezentând predicțiile pentru fiecare oră a zilei. Pentru a limita fenomenul de overfitting, între straturi am aplicat o tehnică de regularizare de tip dropout, cu o probabilitate de 0.2. Optimizarea parametrilor rețelei s-a realizat prin intermediul algoritmului Adam, utilizând o rată de învățare

redusa, mai exact 0.0001, iar funcția de cost aleasă a fost eroarea pătratică medie (MSE), care a fost adecvată pentru astfel de probleme de regresie.

Pentru antrenare s-au folosit 200 de epoci, iar la finalul fiecărei epoci a fost calculată și afișată pierderea medie, pentru a monitoriza evoluția procesului de învățare. După finalizarea procesului de antrenare, modelul LSTM, împreună cu scalar-ele aferente pentru datele de intrare și ieșire, au fost stocate separat pentru fiecare clădire. De asemenea, a fost salvată și lista completă a caracteristicilor (*features*) utilizate în etapa de training. Această abordare a facilitat integrarea ulterioară a modelului în aplicația finală, permitând efectuarea predicțiilor zilnice în timp real, fără a necesita reantrenarea modelului. În plus, modelele salvate au fost utilizate și în faza de optimizare a consumului energetic, constituind un element esențial atât pentru evaluarea profilului de consum anticipat, cât și pentru generarea strategiilor eficiente de optimizare energetică.

În final s-a realizat evaluarea performanței pe setul de test, utilizând metrii precum MSE, MAE, coeficientul de determinare  $R^2$  și SMAPE. Rezultatele au indicat că acest model a obținut cele mai bune performanțe dintre toate modelele testate în cadrul proiectului.

### **5.3. Calcularea profilului de consum**

Pentru a obține o imagine de ansamblu a comportamentului energetic al unei clădiri, într-o anumită zi, am implementat o funcție care construiește profilul de consum orar. Acest profil reprezintă o componentă fundamentală în etapa de optimizare, fiind utilizat ulterior de algoritmii evolutivi pentru a compara consumul prezis cu valorile de referință și pentru a identifica cele mai eficiente strategii de reducere a consumului.

Funcția *generate\_energy\_profile()* are rolul de a genera un profil detaliat al consumului energetic pentru o anumită clădire, la o dată specificată (format YYYY-MM-DD). Această funcție parcurge toate etapele necesare pentru a construi profilul de consum aferent zilei selectate. În cadrul său, este apelată metoda *predict\_energy\_for\_day()*, responsabilă de încărcarea modelului LSTM antrenat pentru clădirea respectivă și de realizarea predicției consumului energetic pe un interval de 24 de ore. Pe urmă, rezultatul obținut este salvat într-un fișier, la o locație specificată, ajutând astfel analiza ulterioară a datelor.

Pe urmă, se preia atât baseline-ul aferent zilei respective, cât și predicția generată de modelul LSTM, combinând două resurse într-un singur *DataFrame*, denumit *profil*. Acest profil include valorile orare ale baseline-ului  $B(t)$ , consumul prezis  $P(t)$  și, acolo unde datele sunt disponibile, consumul real  $R(t)$ . În plus, sunt incluse și diverse metrii de performanță pentru predicție, precum MSE, MAE, SMAPE și  $R^2$ . Totodată, profilul este completat cu variabile meteorologice relevante pentru fiecare oră analizată, cum ar fi temperatura aerului, umiditatea, viteza și direcția vântului sau cantitatea de precipitații. Acestea sunt stocate și ulterior accesate prin intermediul rutelor API dezvoltate pentru aplicație, facilitând astfel obținerea rapidă a datelor necesare pentru vizualizare și analiză direct în interfața utilizatorului.

Un aspect esențial al acestei funcționalități implică determinarea abaterii procentuale dintre valori prezise, cele reale și baseline, pentru a identifica perioadele în care modelul anticipează o deviație semnificativă de la comportamentul standard al cădirii. Aceste informații sunt stocate într-un fișier final, care servește drept bază pentru generarea a două grafice explicative: primul oferă o comparație între valorile reale, cele prezise și baseline, iar al doilea evidențiază abaterile procentuale ale acestor valori față de referință. Astfel, se facilitează o analiză detaliată a performanței modelului și a eventualelor anomalii apărute în comportamentul clădirii.

Prin această metodă, profilul de consum oferă o perspectivă clară și detaliată asupra modului în care clădirea va utiliza energia într-o anumită zi. Acesta devine baza esențială pentru orice proces ulterior de optimizare, asigurând o înțelegere solidă a consumului înainte de implementarea unor soluții suplimentare.

## **5.4. Optimizarea consumului energetic**

Secțiunea dedicată optimizării consumului energetic constituie o componentă fundamentală a acestui proiect, având drept obiectiv principal diminuarea vârfurilor de sarcină și asigurarea unei distribuții mai echilibrate a energiei utilizate pe parcursul unei zile. Procesul începe de la un profil de consum stabilit anterior, care servește drept reper și care include atât valorile estimate de referință ( $B(t)$ ), cât și predicțiile de consum energetic ( $P(t)$ ) furnizate de modelul LSTM pentru ziua analizată. Pe această bază, se implementează strategii ce urmăresc eficientizarea consumului și reducerea solicitărilor excesive asupra rețelei.

Cele două componente furnizează informații detaliate despre modul obișnuit de consum energetic al clădirii, precum și despre consumul proiectat pentru ziua analizată. Pe baza acestui profil, sunt utilizati diversi algoritmi de optimizare, cum ar fi Model Predictive Control (MPC), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) și Genetic Algorithm (GA). Fiecare dintre acești algoritmi urmărește să ajusteze și să redistribue valorile de consum într-un mod inteligent, având ca scop principal minimizarea consumului total sau a vârfurilor de consum, respectând totodată anumite constrângeri predefinite.

În continuare, voi explica pe larg cum se aplică concret fiecare metodă de optimizare asupra profilului energetic al clădirilor.

### **5.4.1. Implementarea unui Model Predictive Control**

În cadrul acestui proiect, a fost analizată utilizarea metodei MPC ca strategie inițială pentru optimizarea consumului energetic zilnic din clădirile inteligente. Implementarea a avut la bază profilul de consum generat anterior, care a integrat atât valorile estimate de referință (*baseline*), cât și predicțiile oferite de modelul LSTM pentru ziua analizată. Obiectivul a constat în determinarea unui vector de ajustare orară  $z(t)$ , menit să modifice consumul estimat astfel încât să se minimizeze consumul total de energie și să se reducă valoarea maximă înregistrată pe parcursul unei zile.

La implementare s-a optat pentru algoritmul *trust-constr* din cadrul bibliotecii *scipy.optimize.minimize*. Acest algoritm este specializat pentru probleme de optimizare neliniară cu constrângeri, permitând formularea de condiții de tip inegalitate și restricții asupra variabilelor. *Trust-constr* combină metode secvențiale cu tehnici de tip interior-point, rezolvând problema prin iterații succesive într-o regiune de încredere definită în jurul soluției curente. Astfel, algoritmul își ajustează dinamica în funcție de progresul obținut, optimizând astfel eficiența procesului de căutare a soluției.

Optimizarea a fost realizată pe un interval de 24 de ore ( $H = 24$ ), iar pentru consumul mediu estimat în fiecare ora ( $B_{avg}(t)$ ), calculat ca media aritmetică dintre valorile de baseline și predicția LSTM, a fost introdus un factor de reglaj,  $z(t)$ . Acest factor a fost constrâns în următoarele limite:

- $Z_{min} = 0.5$  : în orele considerate necritice, consumul putea fi redus până la 50% din valoarea estimată.
- $Z_{max} = 0.9$  : reducerea maximă admisă a consumului a fost de 10% față de estimarea inițială..

Pentru a face diferența între intervalele critice și cele necritice, s-a ținut cont de programul de funcționare al fiecărui tip de clădire. De exemplu, în cazul clădirilor de birouri, intervalul 9:00 – 18:00 a fost ales ca fiind perioadă critică, în timp ce pentru clădirile educaționale, intervalul folosit a fost 8:00 – 15:00. În afara acestor ore, activitatea este mult mai redusă, ceea ce permite implementarea unor strategii mai stricte de reducere a consumului. Astfel, factorul de reglaj oferă atât flexibilitate în gestionarea consumului, dar în același timp, asigură respectarea cerințelor funcționale specifice fiecărui tip de clădire.

În intervalul orelor de vârf, s-a ales o constrângere clară,  $z(t) = 1$ , ceea ce înseamnă că în această perioadă, clădirile operează la capacitate maximă, iar reducerile de consum nu sunt permise. Astfel, funcția obiectivă a fost formulată pentru a minimiza un compromis între consumul total de energie și penalizarea asociată vârfului de sarcină, având forma:

$$Cost = \sum_{t=1}^{24} B_{avg}(t) \cdot z(t) + \gamma \cdot \max_t(B_{avg}(t) \cdot z(t)) \quad (5.1)$$

, unde  $\gamma = 0.5$  a reprezentat un parametru care a reglat importanța vârfului de sarcină în funcția obiectiv. Această abordare a permis modelului să fie capabil să caute un compromis între eficiență energetică și controlul asupra vârfurilor de consum.

La finalul procesului, vectorul optimizat  $z(t)$  a fost aplicat valorii  $B_{avg}(t)$ , obținându-se astfel consumul energetic optimizat pentru fiecare oră. Pe urmă, rezultatele au fost salvate în baza de date, pentru analiză ulterioară și au fost reprezentate grafic, fiind comparate cu valorile de referință și predicția inițială.

#### 5.4.2. Implementarea optimizatorului ACO

Pe lângă celelalte metode de optimizare, s-a implementat și metoda Ant Colony Optimization (ACO), inspirată din comportamentul natural al furnicilor, pentru optimizarea consumului energetic al clădirilor. Scopul principal a fost ajustarea valorilor de consum, estimate anterior prin intermediul modelului LSTM, astfel încât acestea să se apropie cât mai mult de un profil de referință (*baseline*). În același timp, s-a urmărit evitarea consumului excesiv în intervalele de vârf, precum și prevenirea variațiilor bruște în programul sistemului HVAC.

Implementarea s-a bazat inițial pe identificarea orelor de vârf și a celor non-vârf, folosind valorile din baseline, iar pentru acest scop, s-a calculat media consumului orar și abaterea standard. Orele care depășeau pragul format din media plus jumătate din abaterea standard au fost clasificate drept ore de vârf, în timp ce restul au fost considerate non-critice. Această diferențiere a avut un rol semnificativ în cadrul funcției de fitness, întrucât penalizările pentru deviațiile față de baseline au fost aplicate diferit, în funcție de momentul zilei.

Pe urmă am dezvoltat o funcție de fitness care a avut rolul principal în evaluarea performanței fiecărui program HVAC generat. Pentru fiecare oră din intervalul analizat, s-a determinat abaterea absolută dintre consumul ajustat, obținut prin înmulțirea consumului estimat cu valoarea selectată din programul HVAC, și valoarea de referință. Această abatere a fost supusă unei penalizări intensificate în intervalele de vârf, astfel am ales penalizarea  $\omega = 2$ , în timp ce în orele non-vârf penalizarea aplicată a fost numai  $\omega = 0.5$ . De asemenea, în funcție s-a introdus o penalizare suplimentară  $\omega = 0.5$  pentru fiecare modificare a valorii HVAC între două ore consecutive, pentru a descuraja variațiile bruște ale setărilor sistemului. Astfel, funcția de fitness a asigura stabilitatea și eficiența programelor generate.

Optimizarea efectivă a fost implementată în cadrul funcției *run\_aco()*, care a executat algoritmul pe parcursul a 100 de iterații, utilizând un grup de 40 de agenți, denumiți *furnici*. Fiecare agent a generat un program HVAC complet, reprezentat printr-o secvență de 24 de valori, câte una pentru fiecare oră, selectate dintr-un set prestabilit de patru niveluri de control: 0.25, 0.5, 0.75 și 1. Selecția s-a realizat printr-un echilibru între concentrația de feromoni și atracția fiecărei alegeri, echilibru reglat prin intermediul a doi parametri alpha ( $\alpha$ ) și beta ( $\beta$ ). După fiecare etapă de construcție, scorurile obținute de agenți au fost utilizate pentru actualizarea nivelurilor de feromoni, aplicând o rată de evaporare de 0.2, astfel încât soluțiile performante să fie favorizate în iterațiile ulterioare. Acest mecanism a permis algoritmului să evidențieze și să prioritizeze cele mai bune soluții pe parcursul execuției.

După finalizarea procesului de optimizare, a rezultat o secvență optimă de ponderi HVAC, denumită *HVAC\_ACO*, care fost aplicată valorilor prezise ale consumului pentru a genera consumul optimizat. Valorile obținute au fost salvate într-un fișier, împreună cu un grafic comparativ între valorile de baseline, predicție și consumul ajustat prin metoda ACO. Implementarea integrală a fost inclusă în funcția *optimize\_consumption\_aco()*, care primește ca parametri numele clădirii și data de interes, citește profilul de consum aferent zilei respective și execută pașii menționați mai sus.

Prin urmare, această metodă a demonstrat o flexibilitate considerabilă în adaptarea la variațiile zilnice ale consumului, oferind soluții mai dinamice comparativ cu abordările deterministe, precum MPC.

#### 5.4.3. Implementarea optimizatorului PSO

În cadrul proiectului, am implementat Particle Swarm Optimization, care este o metodă inspirată din dinamica socială a roirilor, cu scopul de a ajusta în mod eficient consumul estimat de energie electrică. Obiectivul principal a fost alinierea cât mai precisă a acestui consum la profilul de referință al clădirii. Optimizarea s-a realizat la fel pentru un interval de 24 de ore, utilizând predicțiile generate anterior de modelul LSTM și valorile de baseline.

Acest proces a debutat cu identificarea automată a intervalelor de vârf și a celor off-peak din profilul de referință (*baseline*). Concret, orele a căror valoare depășea media baseline-ului cu cel puțin o jumătate de abatere standard, au fost clasificate drept ore de vârf, în timp ce restul au fost incluse în categoria off-peak. Astfel de diferențiere a fost esențială pentru definirea funcției de fitness, care penalizează abaterile de la profilul de referință, ținând cont de importanța fiecărui interval analizat.

Funcția de fitness, denumită *fitness\_pso*, evaluează fiecare soluție generată de algoritm prin raportarea consumului ajustat, valoare care reprezintă predicția înmulțită cu intensitatea HVAC, la o valoarea de referință. În intervalele de vârf, orice deviație de la această valoare este multiplicată cu un coeficient de penalizare  $\omega = 2$ , descurajând astfel creșterea consumului în perioadele critice. Pentru orele off-peak, penalizarea se reduce la  $\omega = 0.5$ , stimulând mutarea consumului către intervale cu cerere mai redusă.

Optimizarea propriu-zisă a fost implementată prin metoda *run\_pso()*, care pornește cu 30 de particule, adică soluții candidate, desfășurând procesul de căutare pe parcursul a 100 de iterații. Fiecare particulă constă într-o secvență de 24 de valori, selectate dintr-un set de niveluri de control HVAC: 0.25, 0.5, 0.75 și 1. Pe parcursul fiecărei iterații, particulele își ajustează poziția din spațiul de căutare, utilizând atât informația proprie, adică cea mai bună poziție identificată individual, cât și cea colectivă, reprezentată de cea mai bună soluție găsită de întregul roi. Acest tip de comportament este gestionat prin intermediul unor parametri specifici, cum ar fi:

- $w = 0.5$ : acesta reprezintă factorul de inerție, care stabilește în ce măsură particula își păstrează direcția de deplasare anteroară, evitând schimbări brusă și menținând o anumită trajectorie.
- $c_1 = 1.5$ : acesta este coeficientul cognitiv, responsabil de cât de mult este influențată particula de propria sa experiență anteroară. Practic, particula倾nează să se bazeze pe ceea ce a funcționat pentru ea în trecut.
- $c_2 = 1.5$ : reprezintă coeficientul social, care reglează influența celei mai bune soluții globale asupra particulei. Astfel, particula倾nează să fie atrasă de succesul colectiv și să urmeze exemplul oferit de cea mai performantă soluție identificată de întregul grup.

După ce fiecare particulă își actualizează poziția și viteza, valorile sunt rotunjite la cel mai apropiat nivel permis din lista [0.25, 0.5, 0.75, 1.0], pentru a reflecta constrângerile reale ale controlului HVAC. În situația în care scorul obținut de o particulă în noua poziție depășește performanța anteroară, aceasta își actualizează automat cea mai bună poziție personală identificată până în acel moment.

La sfârșitul celor 100 de iterații, se obține soluția globală optimă, care este aplicată direct asupra valorilor de consum estimate de modelul LSTM pentru ziua respectivă. Astfel, rezultatul constă într-un profil de consum ajustat ( $P(t)$ \_adjusted\_PSO), salvat într-un fișier și este generat un grafic comparativ între valorile baseline, predicția inițială și consumul optimizat prin PSO.

Prin urmare, această abordare a oferit un control mult mai flexibil și adaptabil al consumului, facilitând ajustarea rapidă a intensității sistemului HVAC în funcție de variațiile orare ale cererii.

#### **5.4.4. Implementarea optimizatorului GA**

Un ultim optimizator implementat a fost un algoritm genetic, inspirat din principiile selecției naturale, pentru optimizarea consumului estimat de energie dintr-o zi selectată. Procesul s-a bazat pe profilul de consum, construit din valorile de referință și predicțiile generate de modelul LSTM. Optimizarea a presupus identificarea unei combinații adecvate de intensități pentru sistemul HVAC, reprezentate printr-un vector de 24 de poziții, corespunzătoare fiecărei ore din zi. Astfel, fiecare poziție putea lua una dintre valorile discrete permise: 0.25, 0.5, 0.75 sau 1, valori care reflectă procentul din consumul estimat care poate fi păstrat la fiecare oră, permitând astfel un control granular asupra consumului energetic pe parcursul zilei.

La începutul procesului, s-a realizat automat identificarea orelor de vârf și a celor de consum redus, aplicând aceeași logică utilizată și în celelalte strategii de optimizare. Mai exact, s-au calculat media și abaterea standard a valorilor de referință, fiind alese ore de vârf acele intervale în care consumul depășea media cu cel puțin jumătate din abaterea standard, restul valorilor fiind clasificate drept ore off-peak. Această diferențiere a fost esențială pentru funcția de fitness, care a evaluat fiecare candidat, reprezentat sub forma unui vector cu 24 de poziții, penalizând mai sever abaterile de la baseline în orele de vârf ( $\omega = 2$ ), respectiv mai puțin în orele off-peak ( $\omega = 0.5$ ). În plus, asemănător celorlalte metode, s-a introdus o penalizare suplimentară pentru a susține stabilitatea programului HVAC și pentru a evita variațiile brusă și intre nivelurile de intensitate de la o oră la alta.

Pe urmă s-a generat aleatoriu populația inițială, fiecare individ reprezentând o posibilă combinație zilnică a nivelurilor HVAC. În fiecare generație, selecția perechilor de indivizi s-a realizat în funcție de scorurile de fitness, iar noile generații au rezultat prin combinarea genomurilor acestora, utilizând un punct de tăiere selectat aleator. Fiecarui individ nou i s-a aplicat o probabilitate de 10% de mutație, ceea ce presupunea

modificarea unor ore cu un nivel HVAC ales random din lista prestabilită (0.25, 0.5, 0.75 sau 1.0).

După stabilirea inițială a populației și definirea strategiilor de selecție, încrușare și mutație, algoritm genetic a fost rulat pe parcursul a 100 de generații, urmărind identificarea unor soluții tot mai eficiente pentru optimizarea consumului energetic. În fiecare generație, scorurile de fitness au fost calculate pentru toți indivizii, utilizând o funcție de evaluare care a ținut cont de abaterile față de valoarea de referință, penalizările pentru intervalele de peak și off-peak, precum și de variațiile brusăte. Pentru a asigura o calculare corectă a probabilităților de selecție, scorurile au fost convertite într-o formă pozitivă prin scăderea celui mai mic scor și adăugarea unei valori epsilon ( $\varepsilon$ ), pentru a evita eventualele probleme legate de valori nule. Ulterior, pentru fiecare pereche de indivizi necesară, au fost selectate două soluții candidate, care au fost apoi combinate printr-o încrușare într-un punct aleator pentru a genera o nouă pereche de copii. Acești copii au fost supuși unei mutații cu o rată de 10%, ceea ce însemna că anumite ore random din zi puteau primi un alt nivel de HVAC. Astfel, noua generație a înlocuit complet populația anterioară, iar procesul a continuat până la atingerea ultimei generații. În final, a fost selectat individul cu cel mai ridicat scor de fitness, acesta fiind considerat soluția optimă pentru ajustarea consumului.

Astfel, rezultatele aferente zilei optimizate au fost stocate într-un fișier și ulterior au fost reprezentate pe un grafic comparativ cu valorile de referință și cu predicțiile realizate, facilitând astfel o analiză vizuală a diferențelor.

## **5.5. Implementarea dashboard-ului**

În ceea ce privește implementarea aplicației, un aspect central a constat în realizarea unei arhitecturi complete, de tip full-stack, cu scopul de a facilita vizualizarea, gestionarea și optimizarea consumului energetic, aşa cum a fost definit de modelele anterioare. Structura aplicației s-a bazat pe două componente principale: backend-ul construit utilizând framework-ul Python Flask, respectiv frontend-ul, dezvoltat în React. Această arhitectură a fost proiectată pentru a asigura extensibilitate și ușurință în mențenanță, permitând utilizatorilor să acceseze rapid datele relevante, precum și să vizualizeze predicțiile și optimizările energetice aferente fiecărei clădiri pentru o zi anume selectată.

### **5.5.1. Backend**

Pentru componenta de backend am optat pentru framework-ul Flask, recunoscut pentru flexibilitatea și simplitatea oferite dezvoltatorilor. Această alegere s-a dovedit eficientă, întrucât permite definirea rapidă a rutelor API și integrarea facilă cu baze de date relaționale. De asemenea, documentarea automată a endpoint-urilor a fost realizată cu ajutorul pachetului Flassger (Swagger UI), ceea ce a ajutat la o testare mult mai ușoară a metodelor.

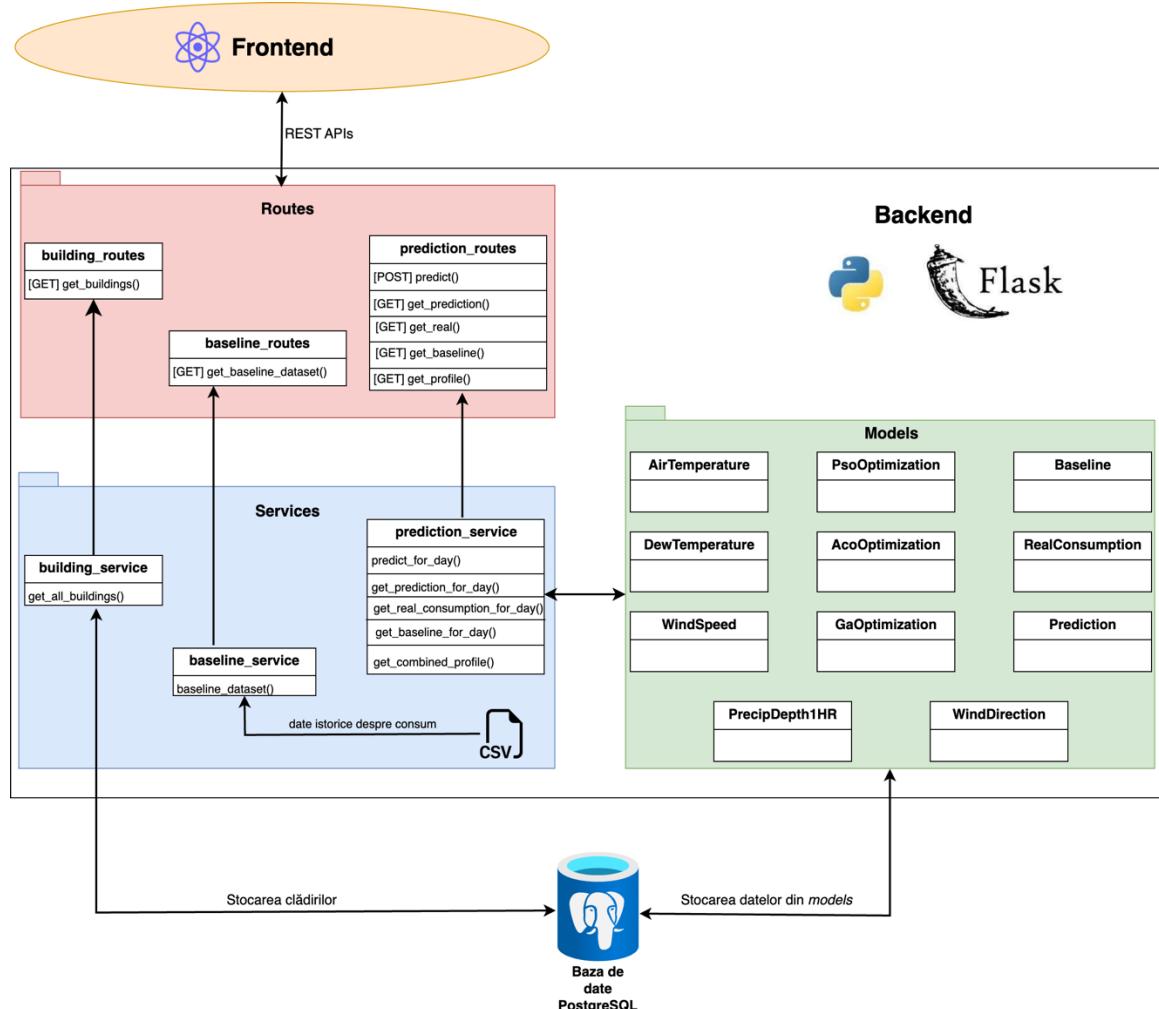


Figura 5.2 Diagrama arhitecturală a aplicației

Structura backend-ului a fost organizată în patru componente principale: *models* (modele de date), *services* (logica de business), *routes* (definirea rutelor API) și fișierul principal de initializare numit *main.py*. Această organizare clară asigură separarea responsabilităților între modulele aplicației, ceea ce contribuie semnificativ la menținere și la posibilitatea de extindere ulterioară a proiectului.

În cadrul dezvoltării backend-ului, am optat pentru utilizarea bazei de date relaționale PostgreSQL, cunoscută pentru fiabilitatea și performanța sa, iar integrarea cu aplicația Flask s-a realizat prin intermediu bibliotecii *SQLAlchemy*, care servește drept *ORM* (Object-Relational Mapping). Această abordare a permis manipularea datelor prin intermediu obiectelor Python, eliminând necesitatea scrierii manuale a interogărilor SQL. În consecință, codul a devenit mai clar, mai bine structurat și considerabil mai ușor de întreținut, facilitând astfel dezvoltarea și modificarea ulterioară a aplicației.

Pentru fiecare tip de date prezent în aplicație, fie că este vorba de predicții obținute prin modelul LSTM, valori meteorologice orare sau rezultate generate de algoritmi de optimizare, am definit câte o clasă de model distinctă în directorul *models*. Aceste clase derivă din *Base*, obiectul fundamental furnizat de *declarative\_base()*, din biblioteca *sqlalchemy.ext.declarative*. Această abordare permite asocierea directă a

unei tabele din baza de date cu o clasă Python, ajutând astfel la maparea obiectelor într-un mod eficient și structurat.

Fiecare model reprezintă, de fapt, o tabelă în baza de date PostgreSQL, având coloane prestabilite, alese în funcție de scopul fiecărui model. Structura acestor modele este una comună, prin folosirea unui identificator unic, care poate fi fie doar *building\_name*, fie combinația dintre *building\_name* și *target\_date*, care reprezintă numele clădirii și data din calendar aleasă. Această asociere funcționează drept cheie primară, asigurând unicitatea fiecărui rând, adică majoritatea informațiilor se referă la o anumită clădire, la o anumită dată. O caracteristică esențială pentru fiecare clasă de model este existența celor 24 de valori numerice, notate de la  $h_0$  până la  $h_{23}$ , fiecare reprezentând o oră din zi. În funcție de tipul datelor analizate, aceste coloane pot conține informații precum consumul de energie, temperatura sau viteza vântului. Această structură orară standardizată permite o analiză mai bună a comportamentului energetic pe parcursul unei zile și facilitează comparația sistematică între diverse modele.

Dintre toate clasele, clasa *Prediction* se distinge prin faptul că extinde structura de bază și integrează cele patru metrii de performanță, cum ar fi eroarea medie pătratică (MSE), eroarea medie absolută (MAE), coeficientul de determinare  $R^2$  și SMAPE. Aceste metrii sunt calculate automat pentru fiecare zi, odată cu predicțiile generate de modelul LSTM, oferind astfel instrumente relevante pentru evaluarea calității predicțiilor, compararea diferitelor metode și monitorizarea acurateței în timp.

Integrarea tabelelor PostgreSQL împreună cu ORM-ul *SQLAlchemy* a asigurat flexibilitate și scalabilitate, ajutând la adăugarea de noi tipuri de date sau funcționalități fără a afecta integritatea arhitecturii existente.

Pachetul *services* joacă un rol central în arhitectura aplicației backend, deoarece separă clar logica de procesare față de infrastructura API-urilor. Practic, acest modul este un intermediar între datele procesate, fie că este vorba despre baseline-uri, predicții sau meta-date despre clădiri, și stratul de prezentare expus prin endpoint-urile Flask. Organizarea sa modulară, cu fișiere dedicare fiecărui tip de operație, precum *building\_service.py*, *baseline\_service.py* și *prediction\_service.py*), permite nu doar o structură clară a funcționalităților, ci și o extensibilitate mai bună a proiectului.

Primul fișier prezentat este *building\_service.py* și servește drept componentă esențială pentru gestionarea interogărilor asupra bazei de date, cu accent pe extragerea listei de clădiri existente în sistem. La inițializare, este construit un engine SQLAlchemy folosind valoarea *DB\_URI*, specificată într-un fișier de configurare. Ulterior, engine-ul este asociat unei sesiuni (*Session*), iar metoda *Base.metadata.create\_all(engine)* este apelată pentru a genera automat tabelele definite de modelele SQLAlchemy, în situația în care acestea nu există deja în schema PostgreSQL. Astfel, fișierul asigură inițializarea corectă a conexiunii la baza de date și crearea infrastructurii tabelului necesar pentru funcționarea sistemului. Funcția principală din acest fișier este *get\_all\_buildings()* și are un rolul central, permitând extragerea eficientă a datelor din baza de date. Mai exact, aceasta inițializează o sesiune de lucru și executa o interogare SQL, vizând tabela *buildings* pentru obținerea numelor clădirilor existente. Rezultatul este o listă de tuple, iar funcția extrage exclusiv primul element din fiecare rând, reprezentând astfel doar denumirile clădirilor. Astfel, lista rezultată este apoi returnată sub formă de obiect JSON, având cheia „*buildings*”, împreună cu un cod HTTP 200 pentru a semnala succesul operației. Această funcție este deosebit de utilă pentru partea de frontend, deoarece ajută la afișarea dinamică a clădirilor disponibile în sistem.

Fișierul *baseline\_service.py* are rolul de a ajuta la generarea și accesarea datelor de tip baseline, adică estimările de consum realizate în absența optimizării, utilizate drept referință comparativă. Acest fișier este integrat direct cu funcționalitățile existente din proiect, cu accent pus pe funcția care generează un fișier ce conține atât valorile estimate pentru baseline, cât și datele privind consumul real, oferind astfel un instrument esențial pentru analiza comparativă a consumului energetic. Acest fișier de service, conține o funcție *baseline\_dataset(building\_id)* care primește numele clădirii ca parametru și inițiază un proces complet de generare a baseline-ului pentru acea clădire, și ulterior verifică existența fișierului rezultat în locația predefinită. În situația în care fișierul nu este găsit, se returnează un răspuns de eroare cu status 404. Dacă fișierul este identificat, acesta este citit utilizând biblioteca *Pandas*, iar datele sunt parcuse rând cu rând și transformate într-o listă de dicționare Python, fiecare element conținând un timestamp, valoarea de baseline și valoarea reală corespunzătoare aceluui interval orar. Astfel, acest payload este returnat în format JSON, permitând structurarea clară a datelor pentru a fi afișate pe frontend sub forma unui grafic comparativ între consumul de referință și cel real.

În fișierul *prediction\_service.py* se regăsește componenta esențială a aplicației de backend. Aici este inițiat procesul de predicție și optimizare pentru o anumită zi, sunt colectate și stocate toate datele relevante în baza de date PostgreSQL, iar aceste date sunt puse la dispoziție prin intermediul unor interfețe către aplicația de frontend. La începutul fișierului, sunt importate modelele din pachetul *models*, se configurează conexiunea la baza de date folosind SQLAlchemy (engine, Session), și este importată funcția din modulul principal de optimizare. Această funcție declanșează simultan procesul de predicție a consumului, generarea baseline-ului și rularea celor trei algoritmi de optimizare (GA, PSO, ACO), asigurând astfel un flux de lucru eficient și integrat.

În cadrul acestui fișier se regăsește prima metodă, numită *predict\_for\_day()*, și reprezintă componenta centrală a acestui fișier, gestionând procesul de calcul și stocare a datelor pentru o anumită zi și o anumită clădire. La apelare, funcția declanșează generarea profilului de consum și optimizarea aferentă, apelând metoda *compare\_optimization(building\_name, target\_date)*, care generează fișiere cu datele rezultate. Ulterior, aceste fișiere sunt preluate cu ajutorul bibliotecii *Pandas* pentru extragerea valorilor de predicție (*predicted\_P(t)*), consum real (*real\_R(t)*), baseline (*baseline\_B(t)*), valorile optimizate obținute prin algoritmii ACO, PSO și GA, precum și datele meteorologice relevante, cum ar fi temperatura aerului, punctul de rouă, viteza și direcția vântului și precipitațiile. Fiecare tip de date dispune de un model corespunzător în SQLAlchemy, astfel se utilizează metoda *session.get()* pentru a verifica existența datelor în baza de date. În situația în care acestea există, valorile orare, de la  $h_0$  până la  $h_{23}$ , sunt actualizate, iar în caz contrar, se instanțiază un nou obiect aferent tipului respectiv (de exemplu *Prediction*, *AirTemperature* etc.) și acesta este adăugat în sesiune. În final, se execută *session.commit()* pentru a asigura salvarea permanentă a datelor. Predicțiile includ și metricile de performanță, acestea fiind stocate exclusiv în modelul *Prediction*, iar pentru celelalte modele, sunt relevante strict valorile la nivel orar. Astfel, la finalul funcției, în baza de date sunt salvate toate informațiile esențiale pentru vizualizare, analiză sau comparații ulterioare, cum ar fi predicția, baseline-ul, consumul real, rezultatele optimizatorilor și datele meteorologice.

O altă metodă din acest fișier o reprezintă *get\_prediction\_for\_day()*, funcție care are rolul de a extrage din baza de date predicția aferentă unei anumite clădiri, într-o zi, utilizând o cheie compusă formată din „building\_name” și „target\_date”. În situația în

care nu există înregistrarea respectivă, funcția va returna o eroare 404 pentru a semnala absența datelor. Dacă predicția este găsită, rezultatul este structurat sub formă de răspuns JSON, incluzând valorile orare organizate într-o listă de dicționare, fiecare având perechi de tipul „hour” și „value”.

Tot în acest fișier se află și funcția *get\_real\_consumption\_for\_day()* care are structura identică cu cea anterioară, însă se axează pe consumul real (*RealConsumption*). În cazul unui răspuns pozitiv, funcția returnează o listă cu valorile consumului real pentru fiecare oră a zilei respective, altfel returnează o eroare cu status 404.

Pe același principiu se bazează și metoda *get\_baseline\_for\_day()*, care caută o înregistrare pentru baseline în baza de date. La fel ca și în cazul celor două metode de mai sus, dacă baseline-ul este găsit, rezultatul este sub formă de răspuns JSON, incluzând o listă cu valorile consumului de referință pentru fiecare oră a zilei respective, altfel se returnează un mesaj de eroare.

Ultima metodă implementată este *get\_combined\_profile()*, care are rolul de a centraliza toate datele relevante într-o structură JSON. Aceasta preia din baza de date toate entitățile asociate zilei respective, cum ar fi predicția, consumul real, baseline-ul, datele meteorologice și cele trei optimizări. Pentru fiecare oră, se generează un dicționar care combină aceste valori, oferind astfel o perspectivă detaliată asupra comportamentului energetic al clădirii pentru ziua respectivă. De asemenea, răspunsul conține și metricile de performanță ale predicției, ajutând astfel analiza gradului de acuratețe a modelului utilizat.

Astfel, pachetul *services* constituie esența logicii aplicației, centralizând operațiile principale într-un mod structural și eficient. Funcțiile definite în acest pachet sunt ulterior folosite de către pachetul *routes*, care le expune sub formă de endpoint-uri publice, asigurând astfel o delimitare clară între logica de business și interfața de acces a aplicației.

Pentru a facilita accesul și manipularea datelor procesate în backend prin intermediul unei interfețe web, a fost necesară expunerea funcționalităților principale sub forma unor API-uri REST. În acest scop, s-a creat un pachet dedicat, denumit *routes*, care asigură conexiunea între metodele implementate în *services* și aplicația Flask. De asemenea, s-a utilizat biblioteca *Flasgger* pentru a documenta aceste funcționalități și pentru a genera automat interfața Swagger, sporind astfel claritatea și accesibilitatea documentației.

Structura pachetului include trei fișiere principale, fiecare dintre ele adresând una dintre funcționalitățile de bază ale aplicației:

- *building\_routes.py* – rută pentru obținerea numelor clădirilor
- *baseline\_routes.py* – rută pentru returnarea datelor de tip baseline și consum real
- *prediction\_routes.py* – rute pentru predicții, consum real, baseline și profiluri combinate

Tabel 5.1 Endpoint-urile implementate

| Endpoint URI | Metodă | Funcționalitate                                      |
|--------------|--------|--|
| /building    | GET    | Returnează lista tuturor clădirilor din baza de date |

|  |      |   |
|--|------|---|
| /baseline-dataset/{building_name}                  | GET  | Generează și returnează consumul real și baseline-ul pentru o clădire             |
| /predict/{building_name}/{target_date}             | POST | Rulează predicția și populează toate tabelele din baza de date                    |
| /prediction/{building_name}/{target_date}          | GET  | Returnează valorile orare prezise de modelul LSTM într-o zi pentru o clădire      |
| /real-consumption/{building_name}/{target_date}    | GET  | Returnează consumul real într-o zi pentru o clădire                               |
| /baseline/{building_name}/{target_date}            | GET  | Returnează baseline-ul unei clădiri într-o zi                                     |
| /profile-consumption/{building_name}/{target_date} | GET  | Returnează toate datele de profil: predicție, real, baseline, meteo și optimizări |

**HVAC Consumption API <sup>1.0</sup>**  
[/apispec\\_1.json](#)

**prediction** Operatii legate de predictii HVAC

- POST** /predict/{building\_name}/{target\_date} post\_predict\_building\_name\_\_target\_date\_
- GET** /prediction/{building\_name}/{target\_date} get\_prediction\_building\_name\_\_target\_date\_
- GET** /profile-consumption/{building\_name}/{target\_date} get\_profile\_consumption\_building\_name\_\_target\_date\_

**real-consumption** Operatii legate de consumul real

- GET** /real-consumption/{building\_name}/{target\_date} get\_real\_consumption\_building\_name\_\_target\_date\_

**building** Operatii legate de cladirile din dataset

- GET** /buildings get\_buildings

**baseline** Operatii legate de baseline-urile din dataset

- GET** /baseline-dataset/{building\_name} get\_baseline\_dataset\_building\_name\_
- GET** /baseline/{building\_name}/{target\_date} get\_baseline\_building\_name\_\_target\_date\_

[Powered by [Flasgger](#) 0.9.7.1]

Figura 5.3 Reprezentarea endpoint-urilor în Flassger

În ceea ce urmează voi trece prin fiecare fișier din pachetul *routes*. Primul fișier, cel pentru clădiri, conține o singură rută, */buildings*, documentată cu Flassger pentru a fi vizualizată în interfața Swagger. Răspunsul acestui endpoint constă într-o listă cu denumirile clădirilor, acestea fiind generate prin apelare metodei *get\_all\_buildings()*

din fișierul *services/building\_services.py*. Implementarea utilizează un blueprint Flask, care funcționează ca un container modular pentru funcționalitățile implementate, asigurând o separare clară a logicii aplicației în componente diferite și gestionabile. Mai exact, după înregistrarea blueprint-ului în aplicația principală Flask, ruta respectivă devine activă și accesibilă.

Următorul fișier, *baseline\_routes.py*, conține un endpoint GET, care generează un baseline complet pentru o clădire și oferă o comparație între valorile reale ale consumului și cele de referință. În spate, este apelată funcția *baseline\_dataset()* din *services/baseline\_service.py*, care preia fișierul generat anterior și furnizează datele sub formă de JSON. Această rută este folosită pentru a analiza comportamentul real al unei clădiri în raport cu valorile de referință pe întregul interval de date disponibil, permitând astfel o evaluare obiectivă a consumului.

Fișierul *precision\_routes.py* reprezintă componenta centrală din cadrul acestui director, având rolul de a gestiona toate funcționalitățile asociate predicțiilor energetice, consumului real, calculului baseline-ului, integrării datelor meteorologice, precum și prelucrării rezultatelor generate de algoritmi de optimizare. Acest modul definește mai multe rute, fiecare corespunzând unei etape semnificative din procesul de predicție și analiză a consumului energetic raportat la o anumită clădire și zi. În primul rând, există o rută care inițiază procesul complet de predicție pentru o anumită clădire și o dată specificată. Aceasta apelează o funcție din *prediction\_service.py* care generează predicția folosind modelul LSTM și execută anumiți algoritmi de optimizare, precum ACO, PSO și GA, iar apoi datele relevante sunt stocate în baza de date PostgreSQL. Pentru a ajuta accesul la aceste date, acest fișier definește rute suplimentare care permite vizualizarea predicțiilor orare, a valorilor reale de consum, precum și a baseline-ului asociat. De asemenea, există un endpoint dedicat pentru generarea profilului complet al unei zile, în care sunt combinate toate datele relevante, precum predicția modelului, valorile reale înregistrate, baseline-ul calculat, precum și informațiile meteorologice și valorile consumului optimizat. Aceste funcționalități sunt accesibile prin rute atent documentate, utilizând adnotarea *@swag\_from* pentru generarea automată a documentației Swagger. Astfel, fiecare rută este asociată cu o funcție din fișierul *prediction\_service.py*, care gestionează accesul, salvarea și structurarea datelor în formatul necesar pentru a fi utilizate în aplicația frontend.

Fișierul *main.py* constituie punctul central al aplicației Flask, având responsabilitatea de a iniția și configura serverul web. Rolul său este de a integra toate componentele backend-ului într-o manieră funcțională. Acest fișier combină parametrii de configurare necesari, înregistrează rutele definite, stabilește conexiunea la baza de date și configurează interfața Swagger pentru documentarea automată a API-urilor.

La începutul fișierului, se configurează aplicația Flask prin creare unei instanțe a clasei principale, iar pe urmă, se stabilește conexiunea la baza de date PostgreSQL, folosind parametrul *DB\_URI*. De asemenea, pentru a permite comunicarea între frontend și API, aplicația activează *CORS* (Cross-Origin Resource Sharing) prin utilizarea extensiei *flask\_cors*, ajutând astfel interacțiunea cu aplicații provenit din alte domenii. În continuare se integrează Flassger, care furnizează o interfață Swagger automată, bazată pe un şablon personalizat, numit *swagger\_template*, aflat într-un fișier de configurare. Apoi, este creată o instanță a obiectului *db* din SQLAlchemy, conectată la aplicație, acesta servind drept punct intermediar pentru modelele backend-ului în relația cu baza de date.

La final, blueprint-urile definite în fișierele de rutare sunt înregistrate în aplicație, ceea ce permite activarea și accesare tuturor endpoint-urilor respective prin intermediul serverului Flask.

Astfel, backend-ul a fost structurat într-o manieră modulară și extensibilă, utilizând Flask pentru expunerea API-urilor, SQLAlchemy pentru gestionarea interacțiunii cu baza de date PostgreSQL și Flassger pentru documentarea automată a rutelor. Prinț-o organizare clară în modele, servicii și rute, backend-ul asigură nu doar o arhitectură bună, ci și o eficiență sporită în stocarea, procesarea și furnizarea datelor necesare în aplicație, fiind pregătit pentru integrarea completă cu partea de frontend.

### 5.5.2. Frontend

Partea de frontend a aplicației a fost dezvoltată utilizând React, cunoscut ca un framework modern și eficient pentru crearea interfețelor web interactive. Implementarea s-a realizat în TypeScript, limbaj care oferă un grad crescut de siguranță în gestionarea tipurilor de date și contribuie la reducerea erorilor la rulare, asigurând astfel un proces de dezvoltare mai robust. Stilizarea a fost gestionată prin fișiere CSS dedicate fiecărei componente, ceea ce ajută la separarea clară între logică și prezentare, dar și la personalizare eficientă a aspectului vizual.

Aplicația funcționează sub forma unui dashboard interactiv, care consumă date provenite din backend-ul realizat în Flask, accesibile prin intermediu API-urilor. Astfel, informațiile privind consumul energetic, predicții, baseline-uri și rezultate ale optimizărilor sunt afișate organizat și intuitiv pentru utilizator. În cele ce urmează, voi prezenta structura componentelor frontend și modul de integrare cu serviciile API.

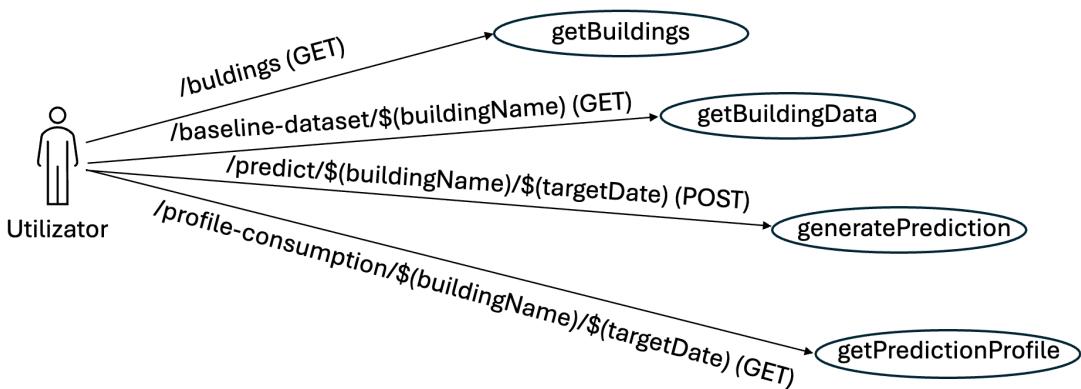


Figura 5.4 Interacțiunea utilizatorului cu API-urile dezvoltate în backend

În cadrul aplicației, utilizatorul beneficiază de diverse funcționalități puse la dispoziție prin intermediu unor API-uri dezvoltate în backend, care sunt apelate din directorul *api* al aplicației frontend. Prin interfață realizată cu React, acesta poate accesa lista completă a clădirilor disponibile în sistem, efectuând o solicitare de tip GET. Această funcționalitate facilitează selecția clădirii pentru care utilizatorul dorește să vizualizeze date sau să genereze o predicție. De asemenea, din aplicația web, utilizatorul are posibilitatea de a vizualiza istoricul consumului energetic aferent unei anumite cădiri. Acest lucru se realizează prin transmiterea unei cereri GET către endpoint-ul care returnează atât valorile real, cât și valorile de referință (*baseline*) pentru întreg intervalul de timp disponibil.

Pentru a obține predicții actualizate privind consumul energetic, utilizatorul are posibilitatea de a transmite o solicitare de tip POST. Prin această acțiune, sistemul generează estimarea consumului pentru ziua selectată și rulează automat optimizatorii, iar rezultatele sunt înregistrate direct în baza de date. Ulterior, pentru a vizualiza profilul energetic aferent unei anumite zile, utilizatorul poate efectua o solicitare GET către endpoint-ul respectiv. Răspunsul returnat conține detalii relevante precum valoarea consumului prezis, consumul real, baseline-ul, date meteorologice și rezultatele optimizărilor, facilitând o analiză detaliată a performanței energetice pentru clădirea respectivă.

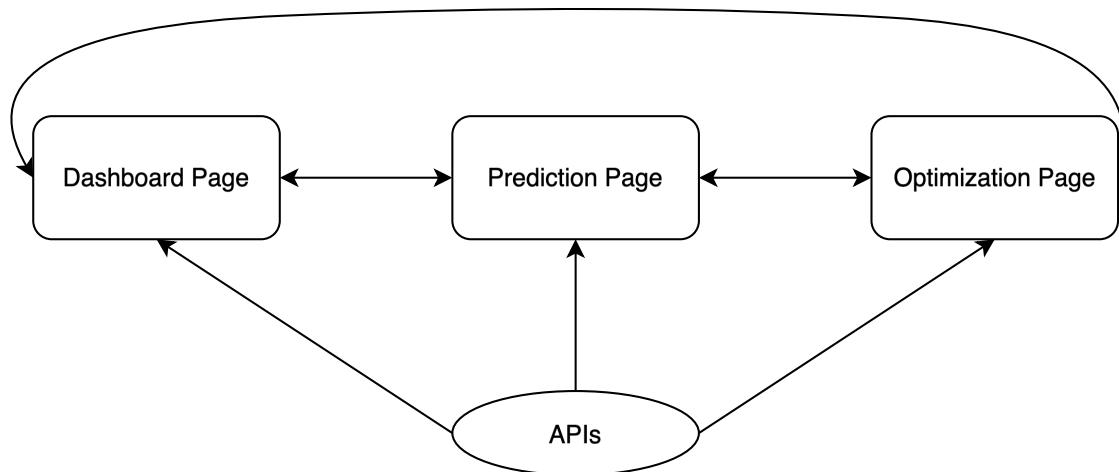


Figura 5.5 Diagrama componentelor din interfața web

Pagina de *Dashboard* din aplicația de frontend reprezintă o componentă importantă a sistemului și ajută la analiza comparativă între consumul real și valorile de referință. Structura acestei componente este împărțită în subdirectoare speciale, fiecare având o funcție clară: *charts* găzduiește componentele pentru vizualizarea grafică a datelor, *info* oferă informații contextuale relevante, *layout* se ocupă de organizarea vizuală a paginii, *types* conține tipuri TypeScript personalizate, iar *utils* combină funcții auxiliare folosite să optimizeze funcționarea generală a paginii. Funcționalitatea principală a acestei pagini constă în faptul că sistemul generează automat un grafic comparativ ce ilustrează evoluția consumului real în raport cu valorile de referință pe întreaga perioadă pentru care există date disponibile. În plus, există și posibilitatea de a vizualiza detaliat consumul real și cel de referință pentru acea zi. Astfel, această combinație între perspectiva generală și analiza punctuală oferă paginii de tip dashboard un rol esențial în monitorizarea și analiza comportamentului energetic al clădirilor.

Structura modulului *Prediction* este organizată pe subdirectoare distincte, fiecare având atribuții clar definite în cadrul dezvoltării și funcționării paginii. Subdirectorul *charts* conține componentele dedicate graficelor, *layout* gestionează organizarea vizuală a elementelor din pagină, iar în *metrics* se regăsesc componentele responsabile cu afișarea metricilor de performanță ale modelului de predicție, cum ar fi MSE, MAE, SMAPE și  $R^2$ . În cele din urmă, *utils* conține metode auxiliare reutilizabile, esențiale pentru transformarea și prelucrarea datelor. În această pagină, se utilizează un endpoint care furnizează profilul complet de consum al unei clădiri pentru o anumită zi, sub forma unui obiect JSON structurat. Datele obținute sunt ulterior procesate și împărțite în componente distincte, fiecare folosite în grafice interactive.

specifice. Printre acestea se numără un grafic comparativ, care ilustrează consumul real, baseline-ul de referință și predicția generată de modelul LSTM, facilitând analiza performanței sistemului de predicție. Pe lângă aceste date energetice, sunt prezentate și date meteorologice, organizate într-un grafic separat. Aceasta compară temperatura aerului, temperatura punctului de rouă, viteza și direcția vântului, precum și cantitatea de precipitații. Aceste informații meteorologice sunt relevante pentru interpretarea corectă a factorilor externi care pot influența consumul energetic al clădirii. În plus, a fost introdusă o funcționalitate suplimentară care generează un profil detaliat de consum pentru ziua respectivă, evidențiind intervalele orare cu cele mai ridicate sau cele mai scăzute valori raportat la media baseline-ului. Această reprezentare ajută la identificarea vârfurilor de consum și a eventualelor comportamente energetice anormale pe parcursul unei zile, transformând pagina într-un instrument eficient de evaluare a eficienței energetice.

Pagina *Optimization* se remarcă printr-o organizare clară și eficientă. Subdirectorul *charts* combină componente destinate vizualizărilor comparative între metodele de optimizare, *layout* se ocupă de organizarea vizuală a paginii, *hooks* gestionează logica interactivă a aplicației, *ui* include componente vizuale precum cardurile de economisire, iar *views* controlează afișarea principală a conținutului în funcție de selecțiile utilizatorului. În cadrul paginii dedicate algoritmilor de optimizare, procesul de implementare se concentrează pe reutilizarea datelor deja extrase din endpoint-ul care furnizează profilul complet al unei clădiri pentru o anumită zi. Aceste informații sunt folosite pentru a genera un grafic comparativ care integrează toate valorile relevante, cum ar fi consumul real, baseline-ul, predicția oferită de modelul LSTM, precum și rezultatele obținute prin aplicare celor trei algoritmi de optimizare. Pe lângă reprezentarea vizuală, s-a implementat un calcul automat privind consumul total pentru fiecare dintre cele șase curbe energetice, pe durata a 24 de ore. Pe baza acestor date, sunt evidențiate atât procentul estimat de economisire a energiei, cât și valoarea economisită, exprimată în kWh, pentru fiecare algoritm de optimizare analizat. Informațiile sunt ordonate în mod dinamic, în funcție de eficiență obținută, facilitând identificarea rapidă a metodei cu cel mai bun randament energetic. Astfel, această pagină completează procesul de analiză, oferind o perspectivă clară și structurală asupra impactului pe care îl strategiile de optimizare aplicate.

Astfel, prin structura modulară a aplicației, împărțită în pagini distincte, frontend-ul asigură o interacțiune eficientă cu datele procesate de backend. Organizarea fiecărei componente în subdirectoare dedicate contribuie semnificativ la claritatea arhitecturii codului și la ușurința proceselor de menenanță. Prin urmare, interfața permite utilizatorului să acceseze rapid și într-o manieră vizuală toate informațiile relevante privind consumul energetic și performanța sistemelor HVAC, prin apeluri directe la API-urile definite la nivel de backend.

Concluzionând, în acest capitol s-a prezentat procesul complet de implementare a aplicației, începând cu calculul baseline-ului și predicțiile consumului energetic, continuând cu eficientizarea energiei prin intermediul algoritmilor de optimizare. Toate aceste componente au fost integrate într-o arhitectură construită cu Flask și PostgreSQL, pentru a asigura o gestionare solidă a datelor. În plus, interfața web, dezvoltată în React, a fost organizată pe pagini specifice, ajutând vizualizarea și analiza comparativă a datelor energetice. În capitolul următor, vor fi detaliate procesele de testare și validare a rezultatelor, pentru a evalua performanța și acuratețea soluției propuse.

## Capitolul 6. Testare și validare

Acest capitol reprezintă o etapă importantă a prezentării, unde este analizată performanța soluției, atât în ceea ce privește acuratețea predicțiilor, cât și eficiența optimizărilor aplicate în gestionarea consumului energetic al clădirilor HVAC. Dacă în secțiunile anterioare s-a discutat despre aspectele de proiectare și implementare ale aplicației, aici accentul se mută pe evaluarea rezultatelor obținute, pentru a confirma validitatea deciziilor tehnice și a metodologiei utilizate pe parcursul dezvoltării proiectului.

Mai departe, se va analiza selecția și prelucrarea dataset-ului, alegerea caracteristicilor relevante, testarea diverselor modele de optimizare și de învățare automată, validarea rezultatelor obținute și interpretarea critică a performanței fiecărei metode folosite. Astfel, scopul final constă în evidențierea soluției propuse în vederea creșterii eficienței energetice, oferind o perspectivă clară asupra impactului real al acesteia.

### 6.1. Alegerea setului de date

Pentru a dezvolta mai bine detalierea testării și validării, este nevoie să se amintească despre setul de date folosit în cadrul aplicației. Prin urmare, în lucrarea științifică realizată de Miller et al. [32] este prezentat un set de date, care include informații orare privind consumul energetic pentru 1636 de clădiri non-rezidențiale, colectate pe o perioadă de doi ani, între 2016 și 2017, din mai multe regiuni din America de Nord și Europa. Acest set de date nu acoperă doar electricitatea, ci și alte tipuri de măsuri, precum apa caldă, aburul, apa rece, gazul, irigațiile și energia solară, oferind astfel o resursă complexă pentru cercetări viitoare în domeniul energetic.

În aplicația dezvoltată am utilizat varianta preprocesată a setului de date, respectiv fișierul *energy\_cleaned.csv*, care conține doar date privind consumul de energie electrică, fără valori lipsă. Datele din acest fișier acoperă intervalul 1 ianuarie 2016 – 31 decembrie 2017 și sunt disponibile la nivel orar. Fiecare clădire menționată în acest set de date este codificată după un anumit format standardizat astfel: fiecare nume conține un identificator de site (de exemplu *Panther*, *Fox* etc.), tipul principal de utilizare (cum ar fi *Office*, *Education* etc.) și un identificator unic (precum *Tina*, *Quintin* etc.). Astfel, o cădire din cadrul educațional ar avea următoare codificare: *Panther\_Education\_Quintin*.

Pentru această analiză am selectat exclusiv clădirile din categoria *Panther*, situate în Florida, la Universitatea Centrală din Florida, datorită consistenței ridicate a datelor asociate acestei locații. Dintre cele 136 de clădiri incluse în site, am selectat un grup de 30, pe baza fișierului de consum, astfel încât să fie reprezentate mai multe tipuri de utilizare, precum săli de evenimente, instituții de învățământ, spații de cazare, birouri, parcare și spații comerciale. Această selecție este justificată de faptul că s-a dorit o diversificare în evaluarea consumului de energie pe tipuri diferite de clădiri, care au de obicei moduri de funcționare diferite.

Un aspect important legat de acest fișier îl reprezintă unitatea de măsură inițială a consumului energetic, respectiv *kBTU* (kilo British Thermal Units). Pentru a asigura consistența datelor și compatibilitatea cu metodele de analiză utilizate, am convertit aceste valori în *kWh* (kilowatt-oră), utilizând relația standard recunoscută în domeniu:

$$1 \text{ kWh} = 1 \text{ kBTU} \times 0.293071 \quad (6.1)$$

Pe lângă fișierul care conține datele despre consumul energetic, am utilizat și fișierul *weather.csv*, care furnizează informații meteorologice orare pentru fiecare site analizat. Acest fișier oferă variabile relevante pentru corelarea cu valorile consumului energetic, cum ar fi: temperatura aerului (*airTemperature*), temperatura punctului de rouă (*dewTemperature*), acoperirea cu nori (*cloudCoverage*), cantitatea de precipitații pe oră și pe șase ore (*precipDepth1HR*, *precipDepth6HR*), presiunea la nivelul mării (*seaLvlPressure*), precum și direcția și viteza vântului (*windDirection*, *windSpeed*). Alegera mea a fost lucrul exclusiv cu datele meteorologice aferente site-ului *Panther*, acestea fiind ajustate temporal pentru a corespunde exact intervalelor de consum electric analizate.

Prin utilizarea a două fișiere deja curate și bine structurate, am eliminat necesitatea unor pași suplimentari de preprocesare a datelor, ceea ce a simplificat integrarea lor în procesul alegerii caracteristicilor și de antrenare a modelelor de predicție. Astfel, această selecție a datelor a constituit întregul proces de prelucrare, predicție și optimizare energetică implementat în cadrul aplicației.

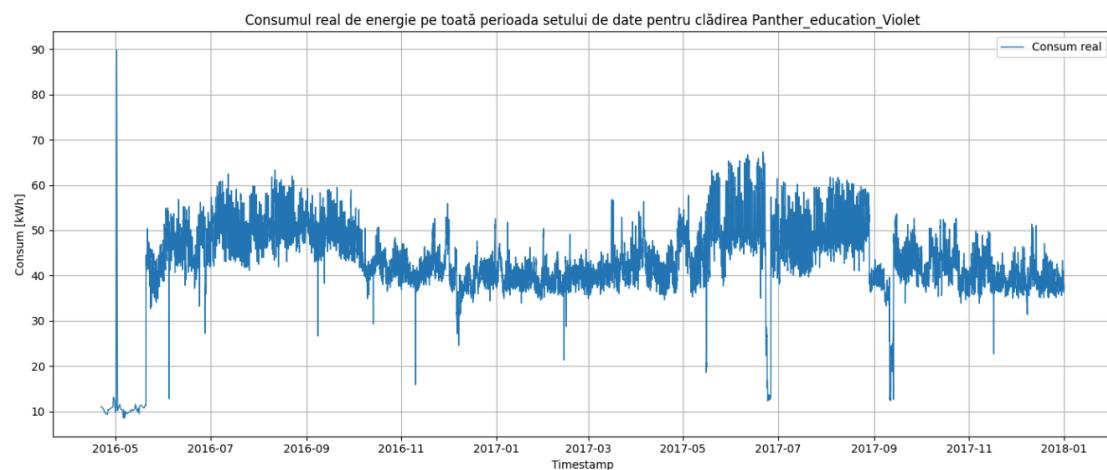


Figura 6.1 Consumul real al unei clădiri educaționale pe toată perioada setului de date

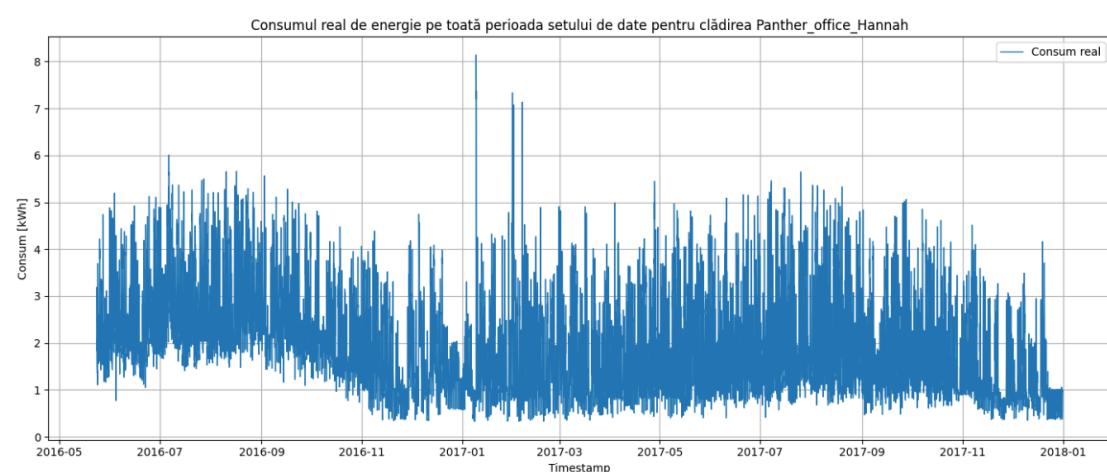


Figura 6.2 Consumul real al unei clădiri de birouri pe toată perioada setului de date

## 6.2. Identificarea caracteristicilor meteorologice (Pearson)

Pentru a obține o predicție cât mai precisă a consumului de energie, a fost necesar să fie selectate doar acele variabile de intrare care influențează în mod direct performanța sistemului HVAC. În acest sens, s-a realizat o analiză de corelație între valorile orare ale consumului de energie pentru fiecare clădire și diverși parametri meteorologici inclusi în setul de date, precum temperatura aerului, temperatura punctului de rouă, viteza și direcția vântului, gradul de acoperire a norilor și nivelul precipitațiilor.

Pentru evaluarea relațiilor dintre aceste variabile, a fost utilizat coeficientul de corelație Pearson, care măsoară intensitatea și direcția unei relații liniare între două variabile. Din punct de vedere matematic, acest coeficient este definit astfel:

$$r_{X,Y} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \cdot \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}} \quad (6.2)$$

, unde X reprezintă seria de consum energetic, iar Y seria de valori pentru o caracteristică meteorologică. Rezultatul  $r$ , variază între valorile -1 și +1, valori care se pot vizualiza în tabelul din capitolul teoretic (vezi *Tabel 4.1*).

În cadrul aplicației, datele meteorologice și energetice au fost alese pe baza timestamp-urilor comune, apoi s-a construit pentru fiecare clădire o matrice de corelație. Fiecare corelație a fost stocată într-un fișier separat, conținând un *heatmap*, generat cu ajutorul bibliotecii *seaborn*, pentru a oferi o vizualizare mai bună asupra influenței fiecărei caracteristici.

Astfel, *Figura 6.3* ilustrează cazul unei clădiri din sectorul educațional, unde se pot observa diferențele specifice între coeficienții de corelație și media generală. Pe baza acestui heatmap, am putut vedea ce tipuri de date meteorologice influențează consumul fiecărei clădiri în parte.

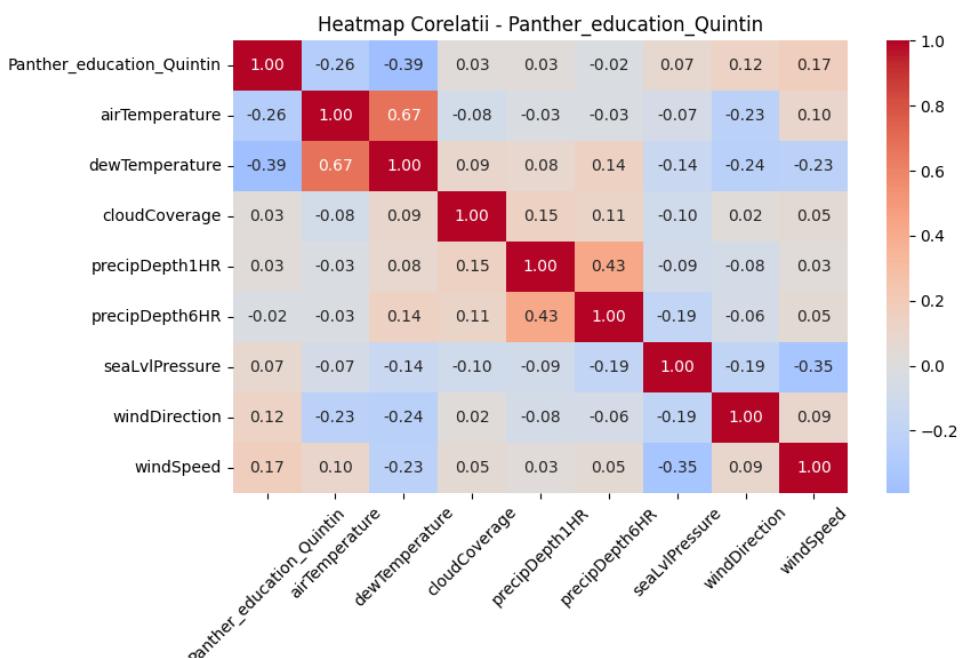


Figura 6.3 Heatmap-ul corelațiilor Pearson pentru o clădire educațională

Pe urmă, s-a realizat o analiză a corelațiilor dintre fiecare variabilă meteorologică și consumul de energie, pentru toate clădirile analizate. Pe baza acestei analize, s-a creat un clasament al importanței variabilelor, prezentat în Figura 6.4. Rezultatele evidențiază faptul că doar *airTemperature* (temperatura aerului) și *dewTemperature* (temperatura punctului de rouă) prezintă cele mai ridicate corelații cu valorile consumului de energie. Astfel, acestea două au fost selectate pentru a fi utilizate drept variabile de intrare în modelele de predicție, celelalte fiind omise din analiză.

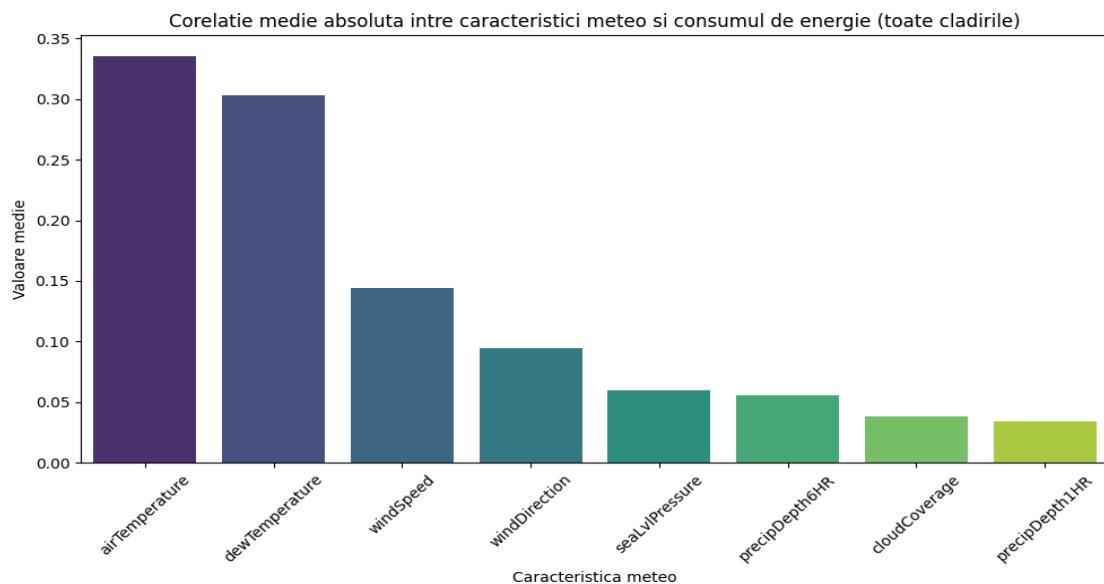


Figura 6.4 Corelația medie absolută între caracteristicile meteo și consumul de energie

Prin utilizarea acestei metode, nu doar că se realizează o selecție eficientă a caracteristicilor relevante, ci se și simplifică modelele de învățare redundante. De asemenea, această abordare susține personalizarea modelelor pentru diferite tipuri de clădiri, având în vedere că anumite caracteristici pot fi importante doar în contexte specifice, precum anumite tipologii de clădiri sau locații geografice diferite.

### 6.3. Procesul de feature engineering

Înainte de antrenarea fiecărui model de predicție, a fost esențială extragerea caracteristicilor relevante din datele brute disponibile. Așadar, procesul de „feature engineering” are un impact semnificativ asupra performanței modelelor, ajutând la identificarea mai precisă a tipelor de consum energetic. Astfel, pentru fiecare înregistrare, au fost extrase următoare caracteristici, care sunt ulterior utilizate ca input în etapa de antrenare a modelelor:

| Denumire feature                 | Descriere   |
|----------------------------------|---|
| <i>hour</i>                      | Reprezintă ora din zi (0-23) și permite identificarea variațiilor orare ale consumului de energie (de exemplu, diferențele între consumul dimineții și cel al serii).   |
| <i>day</i>                       | Marchează ziua calendaristică din lună, fiind utilă pentru a surprinde eventuale tipare repetitive legate de începutul sau sfârșitul lunii.   |
| <i>month</i>                     | Indică luna anului, ajutând la analiza variațiilor de consum cu caracter sezonier, precum creșterile din timpul iernii sau verii.   |
| <i>day_of_year</i>               | Reprezintă poziția zilei din cadrul anului (1-365/366) și oferă un context detaliat pentru sezonalitate, permitând observarea unor fenomene anuale frecvente.   |
| <i>week_of_year</i>              | Constă în săptămâna din an, utilă pentru a identifica fluctuații săptămânale ale consumului energetic, corelate cu anumite evenimente.  |
| <i>weekday</i>                   | Reprezentat printr-un cod numeric pentru ziua din săptămână (0 = luni, 6 = duminică), esențial pentru a distinge între comportamentele de consum din zilele lucrătoare și cele de weekend.  |
| <i>is_weekday</i>                | Este o variabilă binară (1 = weekend), care evidențiază impactul specific al zilelor de weekend asupra consumului energetic.  |
| <i>season</i>                    | Indică un cod numeric pentru anotimp (1 = iarnă, 2 = primăvară, 3 = vară, 4 = toamnă), integrând influența factorilor climatici sezonieri în model.   |
| <i>season_1, ..., season_4</i>   | Reprezintă vectori one-hot encoding pentru anotimpuri, permitând tratarea fiecărui anotimp independent.   |
| <i>weekday_0, ..., weekday_6</i> | Sunt vectori one-hot encoding pentru fiecare zi a săptămânii.   |
| <i>airTemperature</i>            | Indică temperatura aerului, înregistrată la fiecare oră. Aceasta influențează în mod direct funcționarea sistemelor HVAC, întrucât variațiile semnificative de temperatură determină necesitatea ajustării nivelului de încălzire sau răcire în clădiri.    |
| <i>dewTemperature</i>            | Reprezintă temperatura punctului de rouă, care oferă informații suplimentare despre nivelul de umiditate al aerului. Acest indicator este important pentru evaluarea confortului termic percepțut și pentru reglarea eficientă a sistemelor de climatizare. |

Tabel 6.1 Caracteristicile utilizate ca input în antrenarea modelelor

În plus, față de aceste variabile temporale, au fost introduse și valori de lag, respectiv valorile istorice ale consumului energetic din ultimele 72 de ore (3 zile \* 24 de ore). Acestea, notate cu *lag\_1, ..., lag\_72*, oferă modelului informații despre dinamica recentă a consumului, îmbunătățind astfel precizia predicțiilor.

Combinarea acestor surse diverse de date istorice, calendaristice, sezoniere și meteorologice, oferă modelelor de predicție o bază de antrenament semnificativ mai solidă. Integrarea acestor variabile ajută la surprinderea unor tipare complexe de consum energetic, influențate atât de contextul sezonier, cât și de specificul fiecărei zile, sau de condițiile meteo actuale. Astfel, predicțiile devin considerabil mai precise și adaptate la realitatea operațională a sistemelor HVAC din clădiri, permitând optimizarea performanței acestora în funcție de fluctuațiile reale ale mediului.

#### 6.4. Metrici de performanță utilizate

Pentru evaluarea performanței modelelor de predicție dezvoltate, am ales să folosesc patru metrici speciale privind învățarea automată și analiza seriilor temporale: MAE, MSE, SMAPE și  $R^2$ . Fiecare dintre aceste metrici oferă o perspectivă diferită asupra calității predicțiilor, motiv pentru care au fost aplicate atât în etapa de testare a modelelor, cât și în contextul aplicației finale, pentru a valida acuratețea rezultatelor zilnice obținute.

*MAE* (Mean Absolute Error) reprezintă media absolută a diferențelor dintre valorile reale și cele prezise. În practică, cu cât valoarea MAE este mai mică, cu atât modelul de predicție este considerat mai precis. Ecuația pentru MAE este prezentată mai jos (6.3).

$$MAE_{(y,\hat{y})} = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} \quad (6.3)$$

*MSE* (Mean Squared Error) indică media pătratică a diferențelor dintre valorile reale și cele estimate de model. Această metrică reacționează puternic la erorile mari, penalizându-le semnificativ, ceea ce o face utilă pentru identificarea deviațiilor importante. În general, o valoarea MSE mai mică reflectă o performanță mai bună a modelului. Formula de calcul pentru MSE se poate vedea în (6.4)

$$MSE_{(y,\hat{y})} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (6.4)$$

*SMAPE* (Symmetric Mean Absolute Percentage Error) este o metrică procentuală care evaluează în mod echilibrat diferențele dintre valorile reale și cele estimate. Fiind simetrică, tratează la fel atât subestimările, cât și supraestimările, ceea ce o face potrivită pentru analize comparative. Aceasta este utilă în special atunci când se dorește compararea performanței între mai multe clădiri, indiferent de nivelul consumului analizat. Ecuația pentru SMAPE este prezentată în ecuația (6.5).

$$SMAPE_{(y,\hat{y})} = \frac{100}{n} \sum_{i=1}^n \frac{2 * |y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|} \quad (6.5)$$

Metrica  $R^2$ , cunoscută ca și coeficientul de determinare, indică în ce măsură modelul reușește să explice variația valorilor reale. O valoare apropiată de 1 arată o relație foarte bună între model și datele observate, arătând o performanță ridicată de predicție. În schimb, valorile apropiate de 0 arată că modelul nu surprinde eficient structura reală a datelor și explică foarte puțin din variația observată. Formula de calcul pentru metrica  $R^2$  se poate vizualiza în ecuația de mai jos (6.6).

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (6.6)$$

Prin utilizarea celor patru metri, am reușit să surprind detaliat performanța fiecarui model analizat (Random Forest, MLP, LSTM), vizând și modul în care fiecare metodă reușește să prezică datele reale. Astfel, acest set de metri a fost important atât pentru selecția modelului care să fie folosit în aplicația de dashboard, cât și pentru validarea rezultatelor obținute.

În continuare, se vor prezenta rezultatele fiecarui model, realizând o comparație atât pe baza valorilor metricilor, cât și pe baza graficelor, pentru a evidenția modelul cu cea mai bună performanță în contextul aplicației dezvoltate.

## 6.5. Analiza modelelor de predicție

În etapa inițială a procesului de validare, am optat pentru testarea modelelor pe serii temporale orare, considerând că această abordare permite o evaluare mai precisă a capacitații fiecarui model de a anticipa variațiile de consum energetic de la o oră la alta. În analiza aceasta, am inclus trei modele, precum Random Forest, MLP și LSTM, aplicate pe datele provenit de la o clădire educațională, utilizată ca studiu de caz. Rezultatele obținute sunt prezentate în continuare sub forma unor grafice relevante și a metricilor de performanță pentru fiecare model în parte.

Graficul din *Figura 6.5* evidențiază performanța modelului Random Forest, demonstrând o suprapunere semnificativă între valorile reale și cele estimate. Această imagine indică o capacitate ridicată a modelului de a urmări variațiile locale și de a reacționa rapid la schimbările brusă ale consumului, rezultatele fiind susținute și de valorile metricilor afișate în tabel (*Tabel 6.2*). Astfel, nivelul redus al erorilor, împreună cu un coeficient  $R^2$  ridicat, arată că Random Forest explică în mare măsură variabilitatea datelor reale și oferă predicții stabile. Pentru antrenarea modelului, s-au utilizat 200 de arbori de decizie, o adâncime maximă de 10 niveluri și o valoarea fixă pentru *random\_state* = 42, asigurând astfel reproducibilitatea rezultatelor obținute.

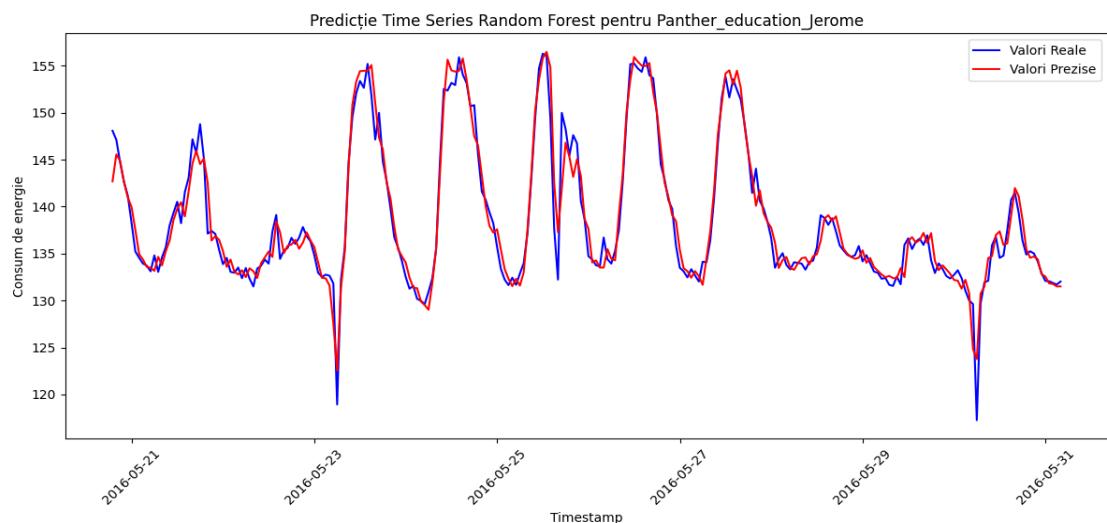
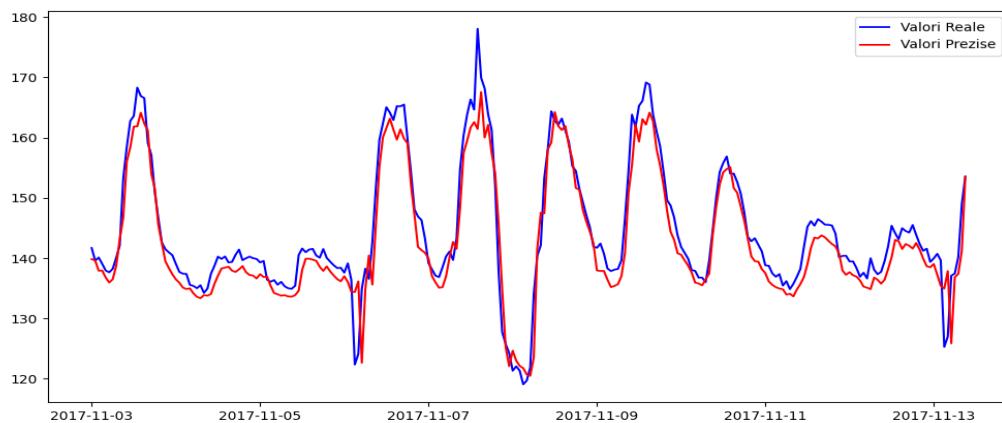
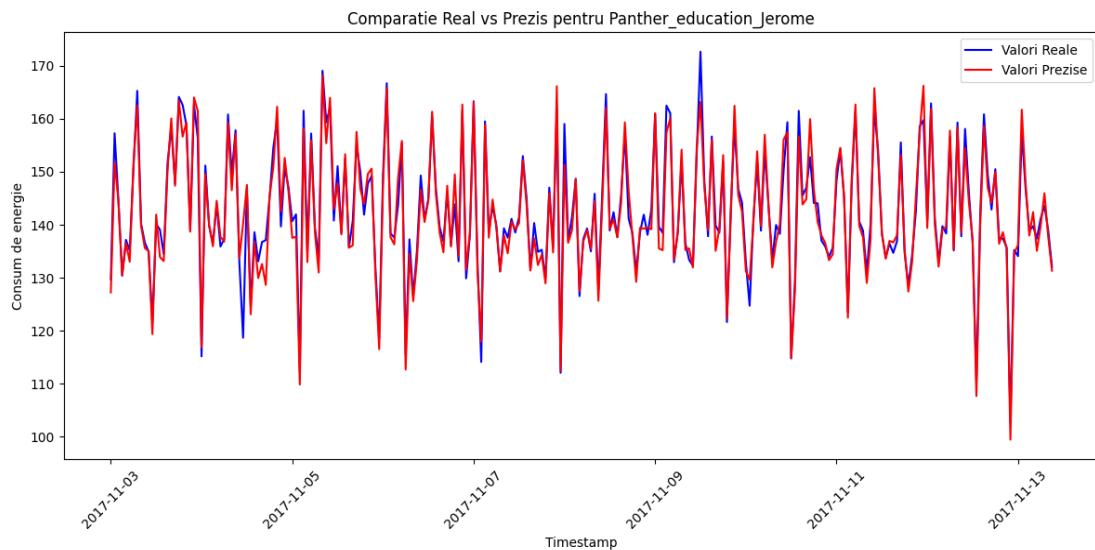


Figura 6.5 Compararea valorilor reale și a celor prezise de modelul Random Forest pentru prediciția orară a consumului de energie

Graficul de mai jos evidențiază performanța modelului MLP. În comparație cu Random Forest, se remarcă o degradare ușoară a acurateței în surprinderea valorilor extreme sau a scăderilor brusă, lucru care este reflectat și de valorile mai mari ale metricilor de performanță (*Tabel 6.2*). Chiar dacă modelul MLP reușește să prezinte tendințele generale ale datelor, performanța sa scade în condiții de variații mari ale consumului, ceea ce îl face mai puțin potrivit pentru astfel de scenarii. Din punct de vedere tehnic, arhitectura a inclus un strat ascuns cu 128 de neuroni, un batch size de 64, o rată de învățare de 0.0005, iar antrenarea s-a desfășurat pe parcursul a 200 de epoci.



**Figura 6.6 Compararea valorilor reale și a celor prezise de modelul MLP pentru prediciția orară a consumului de energie**



**Figura 6.7 Compararea valorilor reale și a celor prezise de modelul LSTM pentru prediciția orară a consumului de energie**

În cel de-al treilea grafic (*Figura 6.7*), este prezentată prediciția generată de modelul LSTM, un algoritm recunoscut pentru abilitățile sale în procesarea datelor temporale secvențiale. Comparativ cu celelalte două modele analizate, LSTM-ul pare

să aducă o variabilitate locală mai mare și o anumită rigiditate în urmărirea valorilor reale. Totuși, performanțele acestui model nu pot fi considerate neglijabile, metricile de performanță ale modelului putând fi vizualizate mai jos în tabel (*Tabel 6.2*). Deși MSE-ul este cel mai ridicat dintre cele trei modele, valorile obținute pentru MAE și SMAPE indică o eroare medie absolută mai redusă față de rezultatele modelului MLP. Astfel, se poate observa că modelul LSTM gestionează cu acuratețe intervalele de consum constant, dar întâmpină dificultăți semnificative atunci când datele prezintă variații bruste. Configurația utilizată a inclus două straturi, fiecare având 128 de neuroni, un batch size de 64, un *input\_dim* adaptat numărului de caracteristici de intrare, un *output\_dim* setat la 1, o rată de învățare de 0.0005 și un număr de 100 de epoci folosit la antrenare. Chiar și cu această arhitectură, performanța modelului scade în condițiile unor schimbări rapide ale datelor.

| <b>Model</b>  | <b>MSE</b> | <b>MAE</b> | <b>R<sup>2</sup></b> | <b>SMAPE</b> |
|---------------|------------|------------|----------------------|--------------|
| Random Forest | 6.20       | 1.52       | 0.93                 | 1.06%        |
| MLP           | 9.32       | 2.28       | 0.9                  | 1.59%        |
| LSTM          | 16.89      | 1.95       | 0.88                 | 1.41%        |

Tabel 6.2 Metricile de performanță pentru predicția orară

Astfel, analizând rezultatele obținute, se poate observa că modelul Random Forest a demonstrat cele mai bune performanțe statistice în ceea ce privește predicțiile orare, fiind urmat de LSTM, iar apoi de MLP. Totuși, după mai multe testări pe diverse clădiri și pe mai multe intervale de timp, a devenit evident că predicțiile orare sunt mult mai sensibile la zgomotul de date și la abaterile punctuale, ceea ce poate afecta acuratețea generală a analizei. Din acest motiv, în etapele viitoare ale proiectului, s-a optat pentru o abordare bazată pe predicții zilnice, în care sistemul estimează consumul pentru următoarele 24 de ore. Această strategie permite o analiză mai stabilă și mai relevantă pentru obiectivul final de optimizare a consumului de energie.

În continuare, se va examina modul în care cele trei modele se comportă atunci când sunt aplicate pe un interval orar de 24 de ore, adică pe parcursul unei zile întregi. Această metodă reprezintă o schimbare semnificativă față de predicțiile orare prezentate anterior, unde estimarea era realizată doar pentru ora imediat următoare, folosind un context recent. Astfel, predicția zilnică implică un grad de dificultate mult mai ridicat deoarece este necesară anticiparea întregii evoluții a consumului pe baza istoricului și a condițiilor actuale, fără posibilitatea de a actualiza predicțiile la fiecare oră.

Prin urmare, graficul din *Figura 6.8* prezintă predicția zilnică generată de modelul Random Forest, configurat cu 200 de arbori de decizie și o adâncime de 10, parametrii selectați pentru a controla complexitatea modelului și a preveni supraînvățarea. Modelul reușește să surprindă modul general al consumului, menținând o traекторie apropiată de valorile reale în zonele stabile. Totuși, nu reușește să reproducă vârfurile sau scăderile bruste, acestea fiind mai uniforme în predicție.

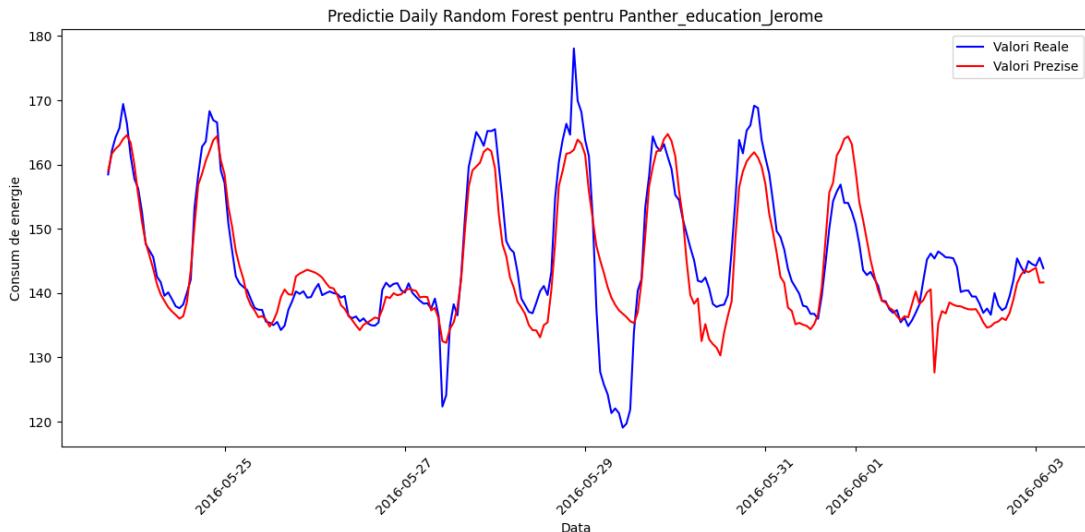


Figura 6.8 Compararea valorilor reale și a celor prezise de modelul Random Forest pentru predicția zilnică a consumului de energie

Rezultatele acestei predicții se pot vizualiza în tabelul de mai jos (*Tabel 6.3*). Comparativ cu versiunea orară a aceluiași model, unde MSE-ul era 6.20 și  $R^2$  de 0.93, se observă o degradare semnificativă a performanței. Această diferență se explică prin faptul că Random Forest nu dispune de o memorie secvențială și se bazează exclusiv pe trăsături statice, limitând-i astfel eficiența în sarcini de predicție pe mai mulți pași.

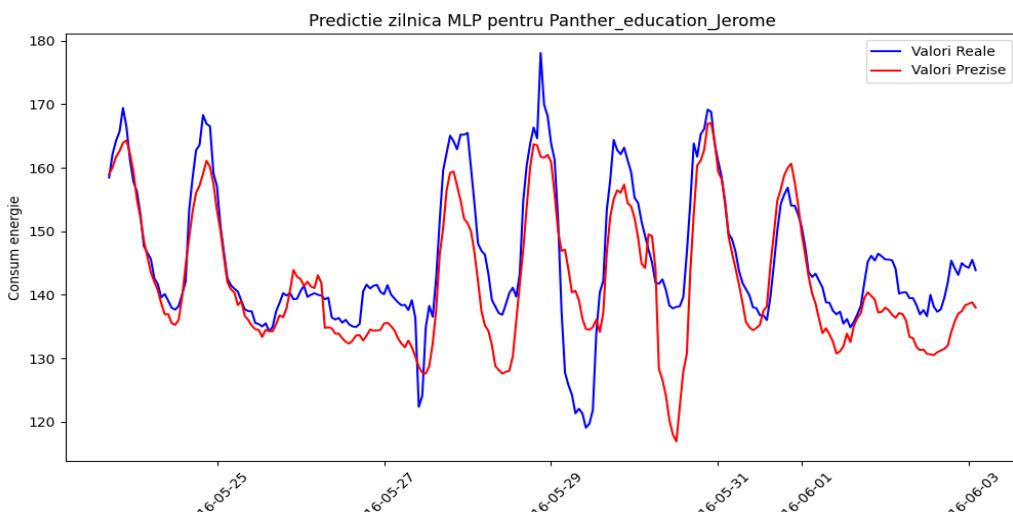


Figura 6.9 Compararea valorilor reale și a celor prezise de modelul MLP pentru predicția zilnică a consumului de energie

În figura de mai sus este prezentată predicția generată de modelul MLP, antrenat cu următorii hiperparametri: *batch size* de 64, un start ascuns de 256 de neuroni, *learning rate* de 0.0001 și 200 de epoci. Din păcate, performanțele acestui model sunt mai slabe comparativ cu cele ale modelului Random Forest, observându-se o deviere semnificativă în zonele cu consum ridicat, precum și o tendință ca valorile prognozate să fie prea netede.

În urma antrenării, valorile metricilor obținute sunt prezentate în *Tabel 6.3*, reprezentând cele mai slabe performanțe din setul actual de teste. Deși rețeaua dispune

de o structură profundă și de hiperparametri bine aleși, limitarea principală a modelului MLP suferă din incapacitatea sa de a menține contextul temporal pe termen lung, ceea ce reprezintă o caracteristică specifică acestor tipuri de rețele.

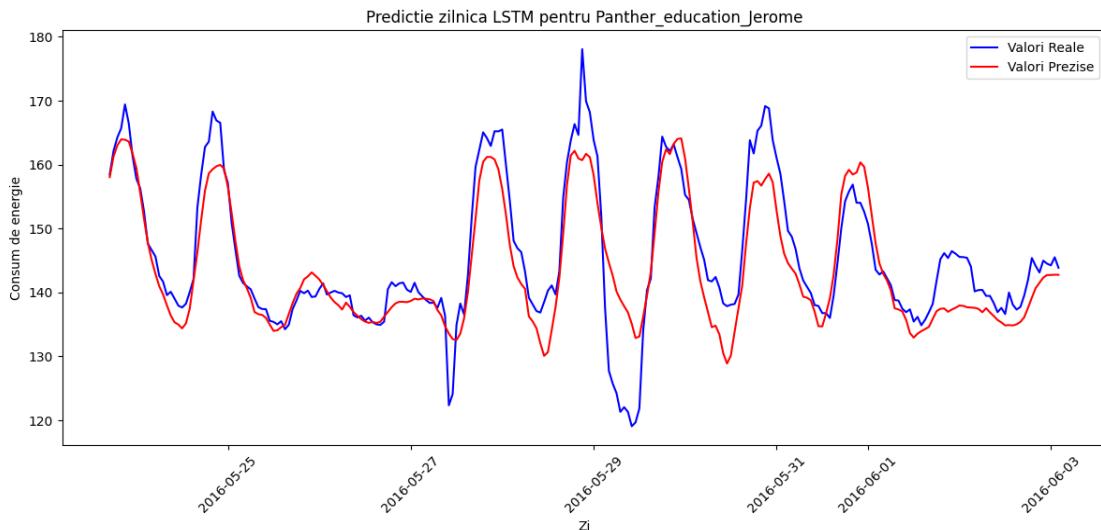


Figura 6.10 Compararea valorilor reale și a celor prezise de modelul LSTM pentru predicția zilnică a consumului de energie

În schimb, *Figura 6.10* ilustrează performanța modelului LSTM, antrenat folosind o dimensiune de batch de 64, un start ascuns de 256, un număr de 3 straturi LSTM, o rată de învățare de 0.0001 și 200 de epoci. Un avantaj important al acestui model constă în abilitatea sa de a învăța secvențe și relații temporale existente, aspect evidențiat printr-o predicție semnificativ mai bine ajustată. După cum se poate vedea și pe grafic, traectoria predicției urmează mult mai bine curbele reale, menținând legătura între maxime și minime și răspunzând mai corect la variațiile de consum.

Așadar, în urma antrenării, rezultatele metricilor de performanță se pot urmări în tabelul de mai jos (*Tabel 6.3*). Deși nu depășește modelul Random Forest la toate metricile, modelul LSTM reușește să mențină o consistență superioară, oferind predicții care păstrează forma și dinamica reală a datelor, fără a avea deviații artificiale.

| Model         | MSE   | MAE  | R <sup>2</sup> | SMAPE |
|---------------|-------|------|----------------|-------|
| Random Forest | 20.8  | 3.05 | 0.77           | 2.11% |
| MLP           | 25.7  | 3.8  | 0.72           | 2.64% |
| LSTM          | 22.25 | 3.6  | 0.76           | 2.52% |

Tabel 6.3 Metricile de performanță pentru predicția zilnică

Comparând aceste modele cu versiunile lor orare, se poate remarcă o scădere generală a performanței, care este un fenomen de altfel anticipat, având în vedere dificultatea suplimentară a predicției pe mai mulți pași. Totuși, modelul LSTM gestionează degradarea mai eficient, reușind să mențină atât forma curbei, cât și o acuratețe numerică mai bună. În schimb, Random Forest își pierde rapid din precizie din cauza lipsei contextului temporar esențial, iar MLP-ul se dovedește vulnerabil la natura secvențială a datelor, nereușind să integreze eficient această structură în procesul de predicție.

Pe baza motivelor prezentate mai sus, modelul LSTM a fost selectat pentru a fi utilizat în continuare în procesul de optimizare energetică. Această alegere nu s-a bazat doar pe metriki, ci pe o analiză critică a modului în care fiecare model răspunde cerințelor sistemului. S-a luat în considerare capacitatea de a reda corect curba de consum, stabilitatea în fața fluctuațiilor și robustețea în situații variate. Astfel, modelul LSTM s-a remarcat prin performanță consistentă la toate aceste criterii, oferind o bază solidă pentru estimări fiabile, ceea ce îl face alegerea optimă pentru generarea unor predicții ce pot fi ulterior optimizate.

## 6.6. Analiza algoritmilor de optimizare

După ce s-au generat predicțiile zilnice ale consumului energetic utilizând modelul LSTM, următorul pas a constat în aplicarea unor algoritmi de optimizare pentru a diminua consumul total și a redistribui energia într-o manieră mai eficientă, cu accent pe diferențierea între orele de vârf (peak) și cele de consum redus (off-peak). Astfel, în această etapă au fost implementate cele trei metode inspirate din procesele naturale, Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) și Genetic Algorithm (GA).

Analiza a fost realizată pornind de la un profil energetic aferent unei zile de miercuri din ultima parte a lunii noiembrie, specific unei clădiri educaționale situate în Florida. Graficul prezentat în *Figura 6.11* surprinde consumul energetic înregistrat în ziua respectivă, incluzând atât valorile reale, cât și cele de referință, alături de predicția generată de modelul LSTM. Acest profil a constituit baza pentru calcularea unor profiluri optimizate de consum pe durata a 24 de ore. Astfel, fiecare algoritm a avut ca obiectiv principal reducerea consumului de energie, precum și netezirea curbei de consum în intervalele de vârf, pentru a preveni sarcinile prea mari la nivelul sistemului HVAC.

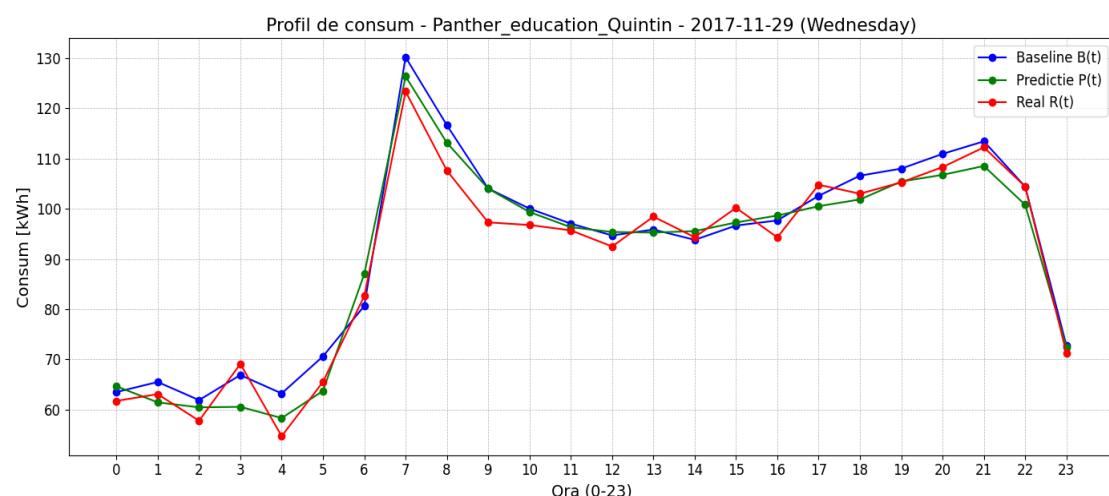


Figura 6.11 Profilul de consum al unei clădiri educaționale într-o zi de miercuri la sfârșitul lunii noiembrie

În cele ce urmează, se vor compara cei trei algoritmi, concentrându-se în special pe modul în care fiecare reușește să redistribue eficient energia în intervalele critice ale zilei. Totodată, analiza va fi susținută de grafice și date concrete, pentru a permite o evaluare obiectivă și riguroasă a eficienței fiecărei metode aplicate.

Înainte de a trece la analiza comparativă, este necesare de precizat că toți algoritmii au funcționat pe baza acelorași niveluri discrete de intensitate, cum ar fi 0.25,

0.5, 0.75 și 1.0. Astfel, aceste valori reflectă procente din consumul estimat ce puteau fi menținute în fiecare oră.

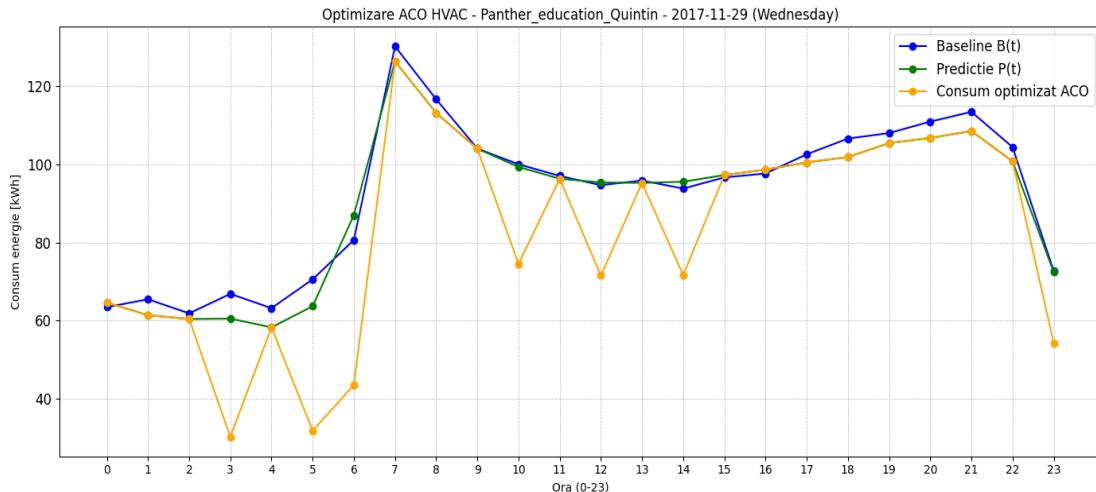


Figura 6.12 Optimizarea cu algoritmul ACO

Graficul generat de aplicarea algoritmului ACO (*Figura 6.12*) evidențiază un comportament variat al consumului optimizat, în special pe durata nopții și în primele ore ale dimineții. În intervalul 0-5, se observă o scădere constantă a consumului, atingând un minim aproximativ 35 kWh în jurul orei 3. Acest rezultat se află în contrast clar cu valorile baseline și predicția LSTM, care rămân relativ constante, între 60 și 65 kWh. Această diferență sugerează o intervenție semnificativă a algoritmului ACO în reducerea consumului în orele off-peak, ceea ce poate fi considerat un aspect favorabil din perspectiva optimizării energetice.

Totuși, în intervalul 5-7, ACO provoacă o creștere rapidă, ajungând la un vârf de peste 120 kWh la ora 7, aproape de valoarea maximă prezisă inițial. O astfel de revenire abruptă poate ridica semne de întrebare privind stabilitatea soluției, mai ales pentru sisteme HVAC, unde fluctuațiile bruște sunt de evitat în considerente operaționale. În continuare, între orele 10 și 22, se remarcă un stil de alternanță între valori ridicate și scăzute ale consumului, cu variații de 20-30 kWh între ore consecutive, cum ar fi orele 13-14 sau 14-15. Aceste oscilații indică o strategie agresivă de redistribuire a consumului, adoptată de ACO pentru a minimiza consumul total, dar care poate afecta negativ stabilitatea funcționării în aplicații reale.

Astfel, ACO demonstrează un potențial ridicat de economisire în intervalele cu preț mic al energiei, însă acest avantaj este obținut cu prețul unei instabilități accentuate în profilul de orar de consum. Prin urmare, implementarea unei soluții bazate pe ACO ar necesita integrarea unui sistem de control suplimentar pentru a asigura operarea stabilă și eficientă a echipamentelor.

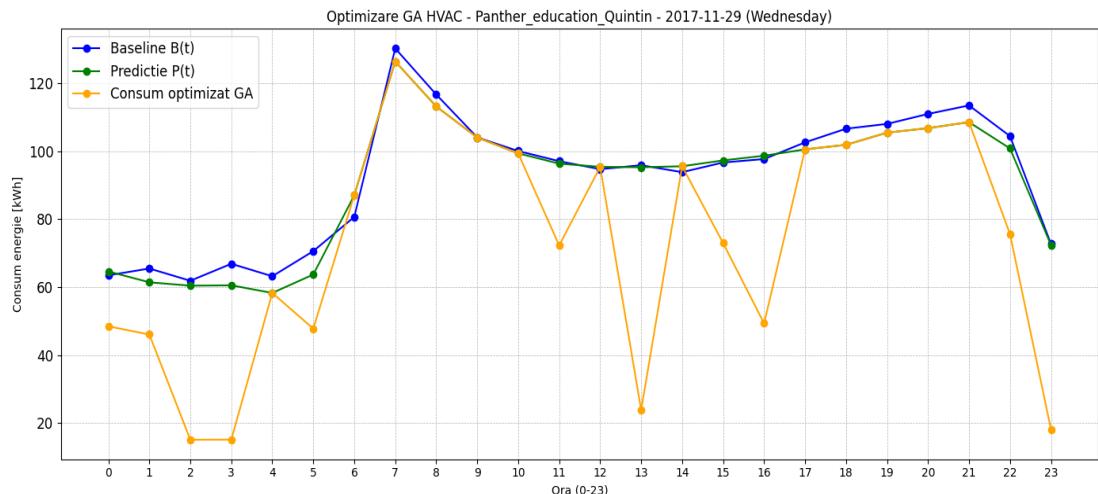


Figura 6.13 Optimizarea cu algoritmul GA

Pe de altă parte, utilizând algoritmul genetic (GA) pentru optimizarea consumului de energie, se observă un profil distinct (*Figura 6.13*) față de cel generat de ACO. GA tinde să producă o curbă energetică mai uniformă în anumite intervale, deși persistă alternanță între valori ridicate și scăzute, mai ales pe parcursul zilei. Pe baza graficului, în timpul nopții (intervalul 0-5), consumul rămâne constant și semnificativ sub nivelul de referință, atingând minime în jurul orei 2, aproximativ 15 kWh. Acest comportament este conform cu ipotezele privind intervalele off-peak, unde activitatea este redusă, iar GA restricționează consumul coresponzător.

Mergând mai departe, în intervalul 6-7 dimineață, se constată o creștere bruscă a consumului, depășind 120 kWh, fenomen asociat cu începutul activităților zilnice, similar cu rezultatele observate la ACO. Totuși, GA prezintă o creștere mai graduală către vârf și o scădere mai controlată ulterior. După ora 9, consumul pare că se stabilizează, însă între 12 și 18 se remarcă oscilații semnificative, cu variații mai ample comparativ cu predicția inițială. Spre exemplu, la ora 13 are loc o scădere abruptă la aproape 25 kWh, urmată de o creștere peste 100 kWh în următoarele ore.

Acste schimbări indică o instabilitate pronunțată în intervalul central al zilei, deși, per ansamblu, GA menține consumul la nivel redus comparativ cu baseline-ul. Spre finalul zilei, algoritmul atinge un echilibru mai eficient, optimizând consumul și reducând semnificativ vârful final (orele 21-22).

Astfel, GA pare să combine o strategie de reducere accentuată a consumului în intervalele off-peak cu o abordare mai echilibrată în perioadele de seară, dar manifestă instabilitate în mijlocul zilei. Așadar, acest tipar poziționează algoritmul genetic între ACO și PSO în ceea ce primește compromisul între eficiență și stabilitate.

Dintre cele trei metode analizate, PSO evidențiază o curbă optimizată cu un grad ridicat de realism și coerență (*Figura 6.14*). Practic, în intervalul serii, consumul energetic scade semnificativ, atingând un minim în jurul orei 1 (aproximativ 15 kWh), iar tranziția către vârful dimineții, orele 6-7, se realizează progresiv, fără salturi brusăte, spre deosebire de celelalte metode. Această evoluție indică o redistribuire treptată a energiei, ceea ce arată eficiența algoritmului PSO în gestionarea consumului. Mergând mai departe, vârful de la ora 7 pare a fi aproape de valoarea estimată, deși ușor diminuat, demonstrând că PSO nu compromite nevoia energetică a clădirii la acel moment de timp.

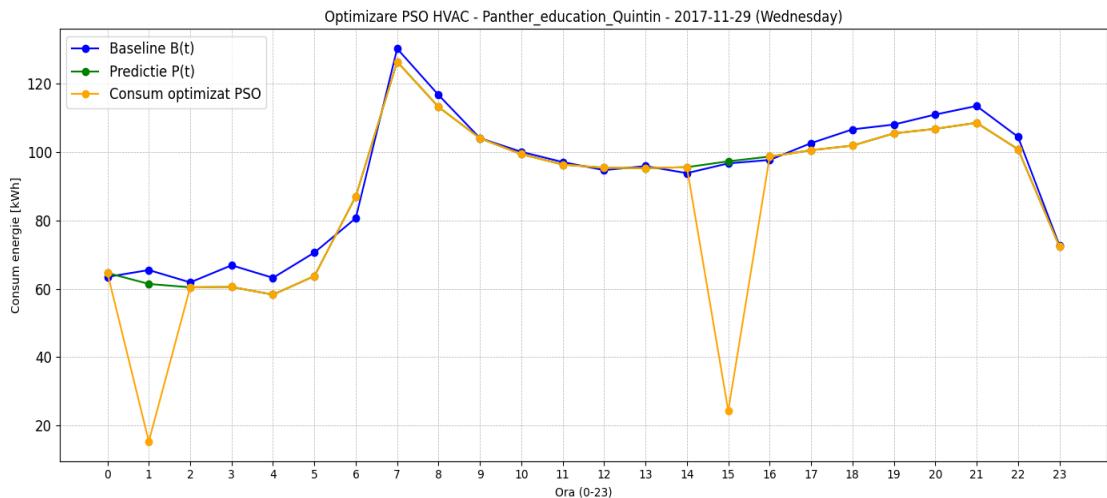


Figura 6.14 Optimizarea cu algoritm PSO

În intervalul 9-21, profilul generat de PSO urmează îndeaproape forma predicției LSTM, reușind totodată să reducă nivelul total al consumului într-un mod constant. Excepția notabilă apare la ora 15, când se observă o scădere bruscă a consumului, totuși fără a perturba semnificativ stabilitatea generală a curbei. Spre deosebire de ACO și GA, algoritmul PSO evită schimbările bruste, oferind un profil energetic caracterizat prin variații moderate și tranzitii naturale.

Prin urmare, PSO se distinge prin capacitatea de a reduce consumul total fără a afecta stabilitatea curbei și fără a introduce instabilități în intervalele critice. Această proprietate este foarte valoioasă în contextul implementării practice, unde sistemul HVAC necesită schimbări line pentru a funcționa eficient și sustenabil.

Ca o încheiere a acestei analize comparative, *Figura 6.15* sintetizează vizual performanțele celor trei algoritmi de optimizare, raportate la o clădire, evidențiind clar economiile de energie estimate în cadrul aplicației de frontend. Interfața grafică prezintă atât procentual, cât și cantitativ, economiile de energie obținute prin fiecare metodă asupra consumului de referință (*baseline*). Conform datelor prezentate în tabelul de mai jos (*Tabel 6.4*), Genetic Algorithm a obținut cea mai mare economie, demonstrând o eficiență superioară în exemplul analizat.

| Algoritm analizat           | Estimarea economisirii energiei |            |
|-----------------------------|---------------------------------|------------|
| Ant Colony Optimization     | 10.83%                          | 240.13 kWh |
| Particle Swarm Optimization | 13.06%                          | 289.66 kWh |
| Genetic Algorithm           | 19.42%                          | 430.65 kWh |

Tabel 6.4 Estimarea economisirii energiei a celor trei algoritmi de optimizare analizați

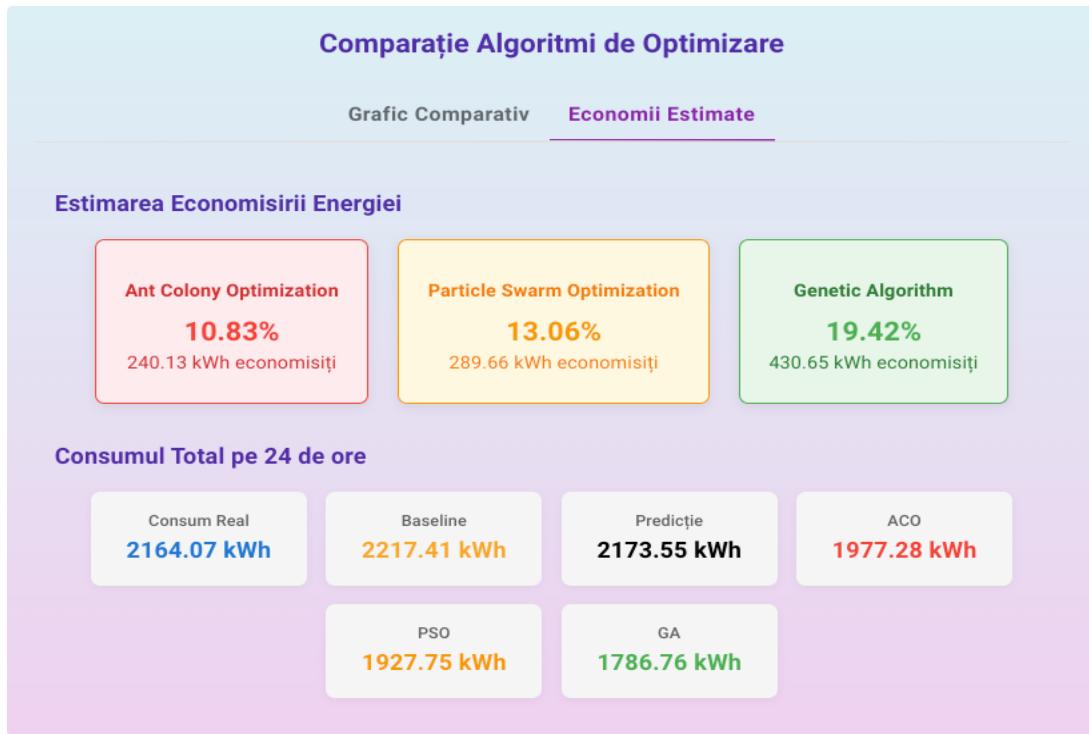


Figura 6.15 Economiile estimate din interfața grafică

Chiar dacă valorile numerice prezentate pot indica o reducere mai mare a consumului prin intermediul unui anumit algoritm, acestea nu reflectă întotdeauna calitatea sau eficiența reală a optimizării. De exemplu, un algoritm care pare să scadă consumul total poate, totodată, să determine variații bruse în profilul de consum, ceea ce poate afecta negativ funcționarea sistemului HVAC. Astfel, aceste numere au în principal un rol informativ și comparativ, însă interpretarea lor corectă necesită o coerentă între analiza comportamentului grafic și evaluarea stabilității soluției propuse.

Prin urmare, fiecare algoritm de optimizare are particularitățile și limitările sale, niciunul nu este universal valabil sau perfect. Spre exemplu, algoritmul genetic s-a remarcat prin cea mai mare economisire de energie, ceea ce, la prima vedere, pare ideal. Totuși, PSO a demonstrat o distribuție orară mai echilibrată și o stabilitate superioară. Așadar, alegerea unui algoritm optim nu ar trebui să se bazeze pe valoarea totală economisită, ci și pe modul în care acesta reușește să gestioneze consumul între orele de vârf și cele de minim, pentru a evita consumul ineficient într-un anumit interval orar.

În final, acest capitol a abordat procesul de testare și validare a aplicației, concentrându-se pe evaluarea critică a metodelor utilizate și a performanțelor obținute. Procesul a început cu selectarea și prelucrarea setului de date, unde au fost identificate caracteristicile esențiale pentru clădirile inteligente, continuând cu o evaluarea comparativă a modelelor de predicție, atât la nivel orar, cât și zilnic.

Rezultatele obținute au arătat diferențe semnificative între cele trei modele de predicție analizate. În cadrul predicțiilor orare, modelul Random Forest a prezentat o acuratețe mare, remarcându-se prin stabilitate și un echilibru eficient între complexitate și performanță. Totuși, modelul LSTM a fost superior în captarea dinamicii temporale pe termen lung, care reprezintă un aspect esențial în contextul predicțiilor zilnice. Din acest motiv, LSTM a fost selectat ca bază pentru etapele ulterioare de optimizare energetică. De asemenea, LSTM a prezentat rezultate superioare în zonele cu variație

constantă a consumului, reușind să identifice atât tendințele sezoniere, cât și comportamentele recurente ale clădirilor non-rezidențiale.

Analiza nu s-a limitat doar la comparații cantitative între modele, ci a inclus și o evaluarea critică a modului în care fiecare algoritm a reacționat în contextul specific studiat. Spre exemplu, MLP-ul a fost eficient în situații cu date mai statice, dar a întâmpinat dificultăți în captarea corelațiilor temporale. În schimb, Random Forest a demonstrat robustețe, însă nu a reușit să generalizeze la fel de bine atunci când datele au devenit mai dinamice. În ceea ce privește modelul LSTM, acesta a fost bun în modelarea dependințelor temporale pe termen lung, însă este mai sensibil la variațiile bruște și poate ajusta prea mult datele de antrenament. Aceste aspecte necesită calibrare atentă și resurse computaționale ridicate pentru a asigura performanțe stabile.

În etapa următoare, accentul a fost pus pe evaluarea algoritmilor de optimizare energetică. Fiecare metodă a fost testată utilizând profilul de consum furnizat de modelul LSTM, iar performanța lor a fost evaluată în ceea ce privește eficiența redistribuirii consumului, cu accent pe intervalele de vârf. Analiza graficelor a arătat diferențe notabile între comportamentele algoritmilor. ACO a demonstrat o stabilitate consistentă, PSO a atins rapid convergență, iar GA s-a remarcat prin cel mai bun raport de economie energetică (pe exemplul studiat). Însă, rezultatele au demonstrat că o economie energetică mai mare nu înseamnă neapărat o soluție mai bună, deoarece variațiile bruște pot afecta confortul și stabilitatea sistemului HVAC.

În cadrul acestui capitol a fost analizată și o metodă clasică de optimizare, precum MPC. Deși metoda este des întâlnită, rezultatele obținute aici au arătat anumite limitări. Ajustarea factorului de penalizare  $\gamma$  nu a generat modificări relevante în soluția finală, fapt susținut de forma curbei de consum care a rămas aproape neschimbată. Acest lucru arată că modelul MPC nu a reacționat eficient la penalizarea consumului în intervale de vârf. Astfel, metoda MPC s-a dovedit insuficient de flexibilă pentru a furniza soluții adaptabile într-un context real de HVAC pentru clădiri. Din acest motiv, s-a ales excluderea acestui model în favoarea metodelor inspirate din natură, precum ACO, GA sau PSO, care pot oferi o explorare mai extinsă și o căutare globală în spațiul soluțiilor.

Astfel, procesul de testare și validare a evidențiat faptul că nu există o singură soluție universală, ci o combinație de metode care, împreună, pot oferi rezultate eficiente. Alegerea arhitecturii LSTM pentru predicțiile zilnice, împreună cu integrarea algoritmilor ACO, GA și PSO pentru optimizare, reprezintă o abordare echilibrată și bine fundamentată, susținută de rezultate obiective. Prin urmare, această etapă a reprezentat o validare a deciziilor anterioare și a oferit o bază solidă pentru integrarea eficientă a componentelor în cadrul aplicației.

Capitolul ce urmează va fi destinat manualului de instalare și utilizare, prezentându-se cerințele aplicației, pașii de instalare și instrucțiunile necesare pentru utilizarea eficientă a platformei.

# Capitolul 7. Manual de instalare și utilizare

Acest capitol reprezintă un ghid practic pentru instalarea și utilizarea aplicației propuse. Vor fi prezentate în mod concret resursele necesare și etapele pentru instalarea și configurarea corectă atât a componentelor backend, cât și a celor frontend.

În partea a doua, accentul va cădea pe experiența utilizatorului final, prin modul în care interacționează cu aplicația. Instrucțiunile vor fi structurate astfel încât să poată fi urmate cu ușurință și de persoane fără cunoștințe tehnice avansate, asigurând o abordare accesibilă și practică a procesului de instalare și utilizare.

## 7.1. Instalarea aplicației

În această secțiune se vor prezenta componentele software esențiale pentru instalarea aplicației, atât pe partea de backend, dezvoltată în Python utilizând Flask, cât și pentru frontend, realizat cu React. Totodată, vor fi specificate cerințele hardware minime necesare pentru a asigura funcționarea optimă a aplicației într-un mediu local.

### 7.1.1. Cerințele software

Pentru instalarea și rularea aplicației în mediul local, este necesară pregătirea unui set specific de instrumente și pachete software, atât pentru componenta de backend, cât și pentru frontend.

Prin urmare, backend-ul aplicației utilizează limbajul Python, având la bază framework-ul Flask. În ceea ce privește mediul de dezvoltare, se recomandă utilizarea *Pycharm*, datorită eficienței sporite oferite în gestionarea proiectelor Python. De asemenea, este indicată folosirea versiunii *Python 3.11*, împreună cu *IDE-ul Pycharm* sau, alternativ, orice alt mediu compatibil cu dezvoltarea aplicațiilor Python.

Pentru gestionarea și stocarea datelor, s-a ales utilizarea *PostgreSQL*, o soluție recunoscută pentru fiabilitatea și performanța sa în domeniul bazelor de date relaționale. Se recomandă utilizarea versiunii *16.9*, aceasta asigurând un echilibru optim între compatibilitate și stabilitate. De asemenea, baza de date va fi instalată local și va comunica cu aplicația prin intermediul bibliotecii specializate *psycopg2*, inclusă în fișierul de dependințe al proiectului.

În final, interfața aplicației este dezvoltată utilizând React, un framework modern cunoscut pentru eficiența sa în construirea interfețelor de utilizator. Pentru compilarea și rularea acestuia, este necesară instalarea pachetului *Node.js*, cu versiunea *20.17*, care integrează și *npm* (Node Package Manager), cu versiunea *10.8*, esențial pentru gestionarea dependințelor proiectului. Se recomandă utilizarea unui editor de cod precum *Visual Studio Code*, apreciat pentru suportul său extins în dezvoltarea aplicațiilor frontend și pentru gama largă de extensii dedicate React și TypeScript.

### 7.1.2. Cerințele hardware

Pentru a asigura o funcționare optimă a aplicației, este de preferat ca utilizatorul să utilizeze un calculator cu o configurație hardware adecvată. În cadrul implementării acestui proiect, toate etapele de dezvoltare, testare, antrenare a modelelor și rularea aplicației, au fost realizate pe un sistem Apple echipat cu procesor *M3 Pro* și *16 GB RAM*, ceea ce a permis o experiență de lucru rapidă și eficientă. Cu toate acestea,

aplicația a fost concepută astfel încât să poată funcționa și pe sisteme cu resurse hardware mai modeste, fără a compromite funcționalitatea sa de bază.

În ceea ce privește procesorul, este suficient un model *Dual-Core* sau *Quad-Core* de cel puțin *2.5 GHz* sau mai rapid. Având în vedere că aplicația utilizează componente grafice și vizualizări de date, este recomandată o placă video dedicată cu *4 GB* de memorie, pentru a asigura o experiență fluentă în interfață.

Minimul recomandat de memorie RAM este de *8 GB*, dar pentru o experiență fluidă, mai ales când se rulează simultan backend-ul, frontend-ul și baza de date, ar fi indicat să se disponă de cel puțin *16 GB*. De asemenea, este esențial ca utilizatorul să disponă de cel puțin *10 GB* liberi pe disc, pentru a putea descărca aplicațiile local, pentru a instala toate dependințele necesare și pentru a salva datele generate pe parcursul utilizării.

Nu în ultimul rând, aplicația funcționează bine pe principalele platforme moderne, cum ar fi *Windows 10* sau versiuni mai recente, *mac OS*, precum și distribuții Linux. Totodată, utilizatorul trebuie să dețină drepturi administrative pentru a putea instala software-ul necesare și pentru configurarea conexiunii cu baza de date PostgreSQL. Astfel, lipsa acestor permișioni ar împiedica desfășurarea corectă a procesului de instalare și funcționare a aplicației.

#### 7.1.3. Configurare backend

Subcapitolul privind instalarea backend-ului prezintă etapele esențiale pentru obținerea, configurarea și inițializarea componentei server a aplicației. Codul sursă al backend-ului este pus la dispoziția utilizatorilor pe platforma *GitHub*, în repository-ul dedicat: <https://github.com/daniboar/EnergyEfficiencyBenchmarkingHVAC>. Pentru a reuși instalarea, se recomandă clonarea locală a repository-ului, utilizând comanda „*git clone*” sau, alternativ, descărcarea arhivei *.zip*.

După obținerea codului, următorul pas este instalarea tuturor pachetelor necesare rulării aplicației. Acest lucru se realizează cu ajutorul fișierului *requirements.txt*, care conține toate bibliotecile utilizate în proiect. Pentru instalare, se deschide un terminal în directorul principal al aplicației și se execută comanda „*pip install -r requirements.txt*”. De adăugat că, această comandă trebuie să fie rulată într-un mediu virtual activat, pentru a asigura izolarea dependințelor specifice proiectului și a preveni eventuale conflicte la nivelul sistemului.

Pentru configurarea conexiunii la baza de date, este necesar să se editeze fișierul de configurare numit *config.py*. În acest fișier trebuie introduse informațiile pentru conectare, cum ar fi adresa serverului, portul, numele bazei de date, numele de utilizator și parola. După stabilirea conexiunii, urmează inițializarea tabelelor bazei de date prin utilizarea unui fișier denumit *tables.sql*, care conține scripturile de creare a tabelelor, elaborate conform modelelor backend-ului.

În final, aplicația poate fi inițializată în două moduri principale. Prima opțiune implică rularea manuală a fișierului principal (*/main/app.py*) din terminal, utilizând comanda „*python app.py*”. Alternativ, se poate folosi direct din IDE-ul PyCharm, unde pornirea se face prin apăsarea butonului de rulare situat în colțul din dreapta sus , asigurându-se că fișierul activ este *main/app.py*. Astfel, odată pornit, serverul Flask devine accesibil local la adresa <http://localhost:5000>.

#### 7.1.4. Configurare frontend

Pentru a instala și rula interfața aplicației, utilizatorul trebuie să obțină codul sursă aferent componentei frontend, acesta fiind disponibil pe platforma GitHub: <https://github.com/daniboar/EnergyEfficiencyBenchmarkingHVAC-fe>. Accesarea codului se poate face fie prin utilizarea comenzi „git clone”, fie prin descărcarea directă a arhivei .zip, în funcție de preferințele utilizatorului.

După obținerea codului sursă, proiectul trebuie deschis într-un editor de cod, cum ar fi Visual Studio Code. Pe urmă, trebuie accesat directorul de frontend și deschis terminalul pentru a executa comanda „npm install”, care va instala toate pachetele și dependințele necesare pentru funcționarea corectă a aplicației.

Odată ce instalarea s-a încheiat, aplicația poate fi lansată local prin comanda „npm start”, ceea ce va deschide automat interfața în browser la adresa <http://localhost:3000>, astfel utilizatorul va avea acces complet la funcționalitățile disponibile. Este esențial ca backend-ul să fie deja pornit în paralel, întrucât interfața comunică direct cu serverul Flask pentru obținerea datelor. Prin urmare, neglijarea pornirii backend-ului va împiedica funcționarea corectă a aplicației de frontend.

## 7.2. Utilizarea aplicației

Aplicația este structurată în trei pagini principale: pagina principală de dashboard, pagina cu predicțiile și o pagină destinată algoritmilor de optimizare, fiecare având roluri distincte în cadrul platformei. Interfața se remarcă printr-o reacție bună la comenzi utilizatorului, fiind proiectată pentru a oferi o experiență prietenoasă și eficientă.

#### 7.2.1. Pagina de Dashboard

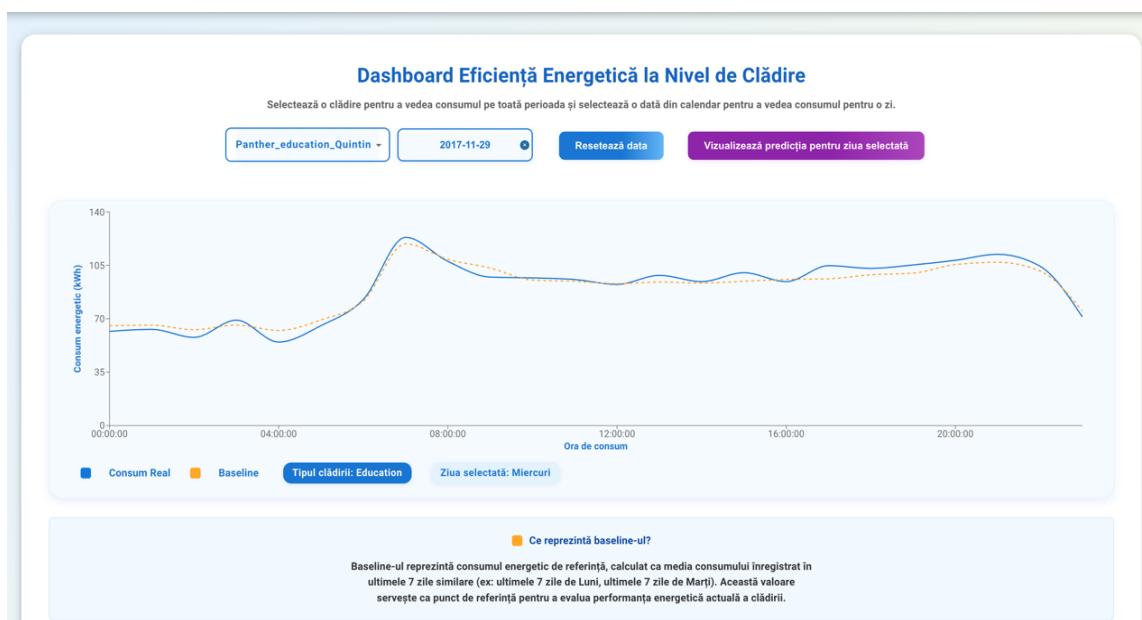


Figura 7.1 Reprezentarea funcționalităților de pe pagina Dashboard

Aici, utilizatorul poate selecta o clădire și poate compara consumul actual cu valorile de referință. Prin selectarea unei date din calendar, sistemul afișează un grafic detaliat pentru ziua respectivă, astfel ajutând la analiza consumului orar într-un mod clar și rapid. Ce se mai poate vedea pe pagina aceasta sunt detalii informative despre

valoarea de referință (*baseline-ul*) și informații teoretice despre clădirea aleasă. Dacă se apasă pe butonul „Vizualizează predicția pentru ziua selectată” ne va redirecționa pe pagina următoare legată de predicția consumului energetic.

### 7.2.2. Pagina Prediction

În această pagină, utilizatorul poate vizualiza metricile de performanță ale modelului de predicție, cat și 3 grafice comparative. Primul grafic reprezintă valorile estimate de modelul LSTM, alături de baseline și consumul real (*Figura 7.2*), iar cel de-al doilea și al treilea reprezintă grafice ale variabilelor meteorologice relevante și ale profilului de consum pentru ziua respectivă, raportat la valorile de referință (*Figura 7.3*). Astfel, utilizatorul beneficiază de o perspectivă detaliată asupra comportamentului energetic al clădirii pentru selectată.



**Figura 7.2 Reprezentarea metricilor și a graficului comparativ între valorile estimate, cele de referință și valorile reale**



**Figura 7.3 Graficele cu variabilele meteorologice și cu profilul de consum reportat la baseline**

Dacă utilizatorul apasă pe butonul „Vezi Optimizatori”, atunci îl va redirecționa pe ultima pagină responsabilă de analiza algoritmilor care optimizează consumul. Altfel, dacă apasă pe butonul „Înapoi la Dashboard”, îl va întoarce la pagina principală.

### 7.2.3. Pagina Optimization

În această pagină se folosesc predicțiile generate anterior, iar aplicația realizează optimizarea consumului prin intermediul algoritmilor ACO, GA și PSO. Aici utilizatorul poate vedea un grafic comparativ cu toate consumurile relevante în aplicație, împreună cu estimările privind economiile de energie, atât în procente, cât și exprimate în Kwh (vezi Figura 6.15). Astfel, utilizatorul poate evalua cu ușurință eficiența fiecărui algoritm analizat.

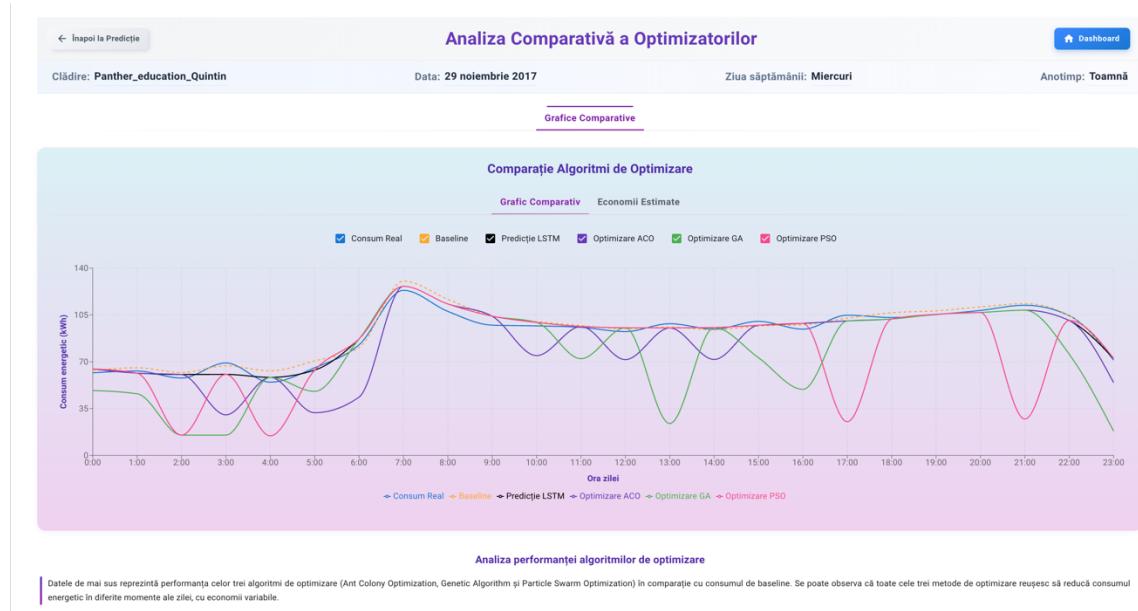


Figura 7.4 Funcționalitatea paginii Optimization

În final, dacă utilizatorul apasă pe butonul „Dashboard” îl aduce pe pagina inițială, iar dacă apasă pe butonul „Înapoi la Predictie” îl redirecționează pe pagina de predicții.

### 7.2.4. Navigarea între pagini

Navigarea se realizează prin intermediul unui meniu fix situat în partea superioară a paginii. De asemenea, interfața răspunde rapid la comenzi, iar elementele sunt organizate eficient, asigurând un parcurs clar al utilizatorului de la selecția clădirii, la predicție și apoi la optimizare. Astfel, experiența de utilizare rămâne una intuitivă și prietenosoasă.

Prin urmare, acest capitol a reprezentat un ghid detaliat pentru instalarea și utilizarea aplicației, oferind instrucțiuni clare atât pentru configurarea software-ului, cât și pentru navigarea în aplicație din perspectiva utilizatorului. Astfel, informațiile prezentate permit oricărui utilizator să ruleze și să testeze aplicația local, fără a avea dificultăți majore. În continuare, documentația va aborda ultimul capitol esențial, concentrându-se asupra concluziilor generale ale proiectului.

## Capitolul 8. Concluzii

Acest capitol prezintă o sinteză a parcursului proiectului, subliniind încă o dată obiectivul dezvoltării unui sistem integrat, capabil să realizeze prognoza și optimizarea consumului energetic în clădiri non-rezidențiale, utilizând tehnici avansate de învățare automată. Tema a fost abordată în contextul procesului de digitalizare și optimizare a infrastructurii energetice, în special în sectorul HVAC, recunoscut ca unul dintre cei mai mari consumatori de energie din clădirile comerciale și instituționale. Astfel, proiectul răspunde unei provocări actuale, propunând soluții inovatoare pentru reducerea risipei energetice și optimizarea costurilor operaționale.

### 8.1. Contribuții și realizări

Lucrarea a avut ca scop dezvoltarea unui instrument avansat pentru analiza consumului de energie, identificarea unor tipare relevante utilizând modele cum ar fi Random Forest, MLP și LSTM, precum și aplicarea rezultatelor acestor predicții pentru optimizarea consumului zilnic. Sistemul a fost testat și validat pe un set de date real, provenit din clădiri non-rezidențiale din statul Florida (identificate prin prefixul *Panther*), iar integrarea datelor meteorologice asociate a contribuit la creșterea acurateței modelului.

În urma evaluării performanțelor obținute, modelul LSTM s-a remarcat drept cea mai potrivită opțiune pentru prognoza zilnică, reușind să găsească un echilibru optim între precizie și stabilitate. Acest model a constituit baza pentru elaborarea profilului energetic orar, asupra căruia au fost ulterior aplicati trei algoritmi de optimizare, precum Ant Colony Optimization, Particle Swart Optimization și Genetic Algorithm. Fiecare algoritm a fost analizat din perspectiva eficienței în reducerea consumului global și în redistribuirea sarcinii energetice în afara intervalelor de vârf, urmărind obținerea unor rezultate cât mai productive.

Legat de contribuțiile efective ale acestei lucrări, ele derivă clar din obiectivele stabilite în *Capitolul 2*. Mai exact, s-a realizat o analiză critică a metodelor existente prezentate în literatura de specialitate, adresând atât tehniciile de predicție, cât și pe cele de optimizare. Acest demers a fost detaliat în *Capitolul 3*, unde s-au examinat cu atenție metodele cel mai frecvent utilizate în cercetare și în industrie, fapt ce a stat la baza selecției ulterioare a algoritmilor implementați. Analiza a inclus atât factori importanți legați de benchmark-ul energetic, de metode de învățare automată dedicate predicției, cât și de strategii de optimizare inspirate din procese naturale.

În *Capitolul 5* a fost abordată în mod riguros dezvoltarea și proiectarea sistemului, inclusiv detalii precise despre modul de calcul al baseline-ului, precum și arhitectura fiecărui model de predicție. Această secțiune continuă cu prezentarea implementării fiecărui algoritm de optimizare, pornind de la profilul energetic obținut, urmând apoi analize comparative și calcule relevante privind economiile potențiale. Totodată, în acest capitol este documentată realizarea interfeței grafice de tip dashboard, integrarea API-urilor REST pentru asigurarea comunicării între backend și frontend, precum și salvarea rezultatelor în baza de date PostgreSQL. Astfel, capitolul oferă o privire largă asupra etapelor esențiale ale realizării sistemului.

Capitolul 6 a reprezentat atât prezentarea setului de date, procesele de feature engineering, cât și testarea și validarea tuturor componentelor esențiale aplicației.

Modelele de predicție au fost evaluate atât din perspectivă numerică, folosind metriki precum MSE, MAE, SMAPE și  $R^2$ , cât și vizuală, prin intermediu graficelor care compară predicțiile cu valorile reale. De asemenea, a fost analizat în detaliu comportamentul fiecărui algoritm de optimizare, evidențiindu-se atât avantajele, cât și limitările acestora, cu scopul de a oferi o imagine clară asupra performanței generale a proiectului.

Una dintre realizările esențiale ale acestei lucrări constă în combinarea și integrarea mai multor componente diverse într-o singură aplicație funcțională, accesibilă prin intermediul unei interfețe web (prezentată în *Capitolul 7*). Utilizatorul poate astfel vizualiza datele privind consumul energetic al unei clădiri și poate genera predicții pentru consumul din anumite zile. În plus, are posibilitatea de a analiza în mod direct impactul strategiilor de optimizare asupra consumului total de energie, ajutând astfel la evaluarea măsurilor propuse pentru optimizarea energetică.

## **8.2. Limitări ale sistemului propus**

Chiar dacă sistemul analizat a reușit să atingă anumite obiective relevante, trebuie discutat și despre limitările acestuia, pentru a înțelege mai bine contextul în care poate fi aplicat. Una dintre ele ține de dependența semnificativă de date istorice care trebuie să fie atât complete, cât și corecte. Pentru ca algoritmii să ofere rezultate relevante, este esențial să existe suficiente informații din trecut pentru fiecare zi analizată, cum ar fi același tip de zi, același interval orar și aceleași condiții. În lipsa acestor date, predicțiile generate de sistem pot deveni inconsistent sau chiar irelevante. Un alt aspect de luat în calcul este existența unui *lag* în colectarea și procesarea datelor de consum. Fără o infrastructură *IoT* performantă, capabilă să transmită informațiile în timp real, sistemul poate funcționa doar în regim offline sau cu întârziere.

Un alt punct vulnerabil ține de modul în care sistemul face față evenimentelor neprevăzute, cum ar fi întreruperile de curent, defecțiunile senzorilor sau modificările bruse de comportament ale utilizatorilor. Fără mecanisme avansate pentru detectarea și corectarea automată a anomaliei, există riscul ca sistemul să producă rezultate eronate sau neconforme cu realitatea. De asemenea, în configurația sa actuală, sistemul este dependent de procesări care implică un cost computațional moderat spre ridicat, în special pentru antrenarea modelelor sau rularea algoritmilor evolutivi. Acest lucru îl face mai greu de utilizat pentru infrastructuri cu resurse limitate.

Nu în ultimul rând, implementarea și calibrarea acestui tip de sistem reprezintă un proces destul de complex. Selectarea caracteristicilor relevante, precum și ajustarea hiperparametrilor, solicită un nivel ridicat de expertiză tehnică. Astfel, în lipsa personalului specializat, numeroase organizații pot întâmpina dificultăți în aplicarea practică a acestei soluții.

## **8.3. Direcții viitoare de dezvoltare**

Deși sistemul prezentat reprezintă o bază solidă pentru estimarea și optimizarea consumului energetic în clădiri non-rezidențiale, există încă multiple direcții de dezvoltare care pot fi abordate pentru extinderea funcționalității și pentru adaptarea soluției la diferite contexte operaționale. O primă direcție semnificativă ar fi extinderea orizontului de predicție dincolo de intervalul zilnic, către perioade mai mari, precum săptămână, lună sau chiar un an. O astfel de abordare ar permite o planificare energetică mai strategică, ajutând la gestionarea eficientă a resurselor prin adaptarea la sezonalitate și la tendințele de consum pe termen lung.

Integrarea datelor furnizate de senzorii *IoT* instalati in cladire aduce un nivel semnificativ de detaliu privind consumul energetic, atat la nivel de echipament, cat si pe zone diferite. Aceasta granularitate ridicata va oferi modelelor de predictie o baza de antrenare mai solidă, care va reduce considerabil incertitudinile si va creste precizia rezultatelor obtinute. De asemenea, existenta acestor date detaliate permite implementarea unor strategii avansate de control automat pentru cladirile inteligente. Acest lucru faciliteaza adaptarea rapidă a functionarii acestora in functie de orice abateri identificate fara de profilul optim de consum energetic.

O directie de cercetare importantă constă în testarea și validarea sistemului în medii geografice și climatice diferite față de setul inițial de date din Florida. Acest lucru este fundamental pentru a determina în ce măsură soluția poate fi generalizată și adaptată la particularitățile regionale. În plus, se pot explora și arhitecturile hibride, care combină avantajele algoritmilor tradiționali cu cele ale rețelelor neuronale de tip *Transformer*. Această abordare ar putea aduce îmbunătățiri semnificative în ceea ce privește modelarea și învățarea contextului dinamic al consumului de energie.

Aceste directii nu vizează doar îmbunătățirea performanței tehnice, ci se axează în principal pe identificarea sistemului ca un instrument sustenabil și scalabil pentru medii reale. În contextul complexității energetice a cladirilor inteligente, este esențial ca soluțiile propuse să fie robuste, adaptabile și să fie cât mai explicate, astfel încât să poată răspunde eficient provocărilor practice.

## Bibliografie

- [1] E. Wang, N. Alp, J. Shi, C. Wing, X. Zhang și H. Chen, „Multi-criteria building energy performance benchmarking through variable clustering based compromise TOPSIS with objective entropy weighting,” *Energy*, vol. 125, pp. 197-210, 2017.
- [2] Z. Li, Y. Han și P. Xu, „Methods for benchmarking building energy consumption against its past or intended performance: An overview,” *Applied Energy*, vol. 124, pp. 325-334, 2014.
- [3] Z. Du, X. Jin, X. Fang și B. Fan, „A dual-benchmark based energy analysis method to evaluate control strategies for building HVAC systems,” *Applied Energy*, vol. 183, pp. 700-714, 2016.
- [4] Y. Bichiou și M. Krarti, „Optimization of envelope and HVAC systems selection for residential buildings,” *Energy and Buildings*, vol. 43, pp. 3373-3382, 2011.
- [5] M. Yalcintas și U. A. Ozturk, „An energy benchmarking model based on artificial neural network method US Commercial Buildings Energy Consumption Survey (CBECS) database),” *Wiley InterScience*, vol. 31, pp. 412-421, 2007.
- [6] L. Wang, S. Greenberg, J. Fiegel, A. Rubalcava, S. Earni, X. Pang, R. Yin, S. Woodworth și J. Hernandez-Maldonado, „Monitoring-based HVAC commissioning of an existing office building for energy efficiency,” *Applied Energy*, vol. 102, pp. 1382-1390, 2013.
- [7] D. Singh, M. Arshad, B. Tyagi și G. Kalia, „Predictive Maintenance Strategies for HVAC Systems: Leveraging MPC, Dynamic Energy Performance Analysis, and ML Classification Models,” *IRE Journals*, vol. 7, 2023.
- [8] A. Capozzoli, F. Lauro și I. Khan, „Fault detection analysis using data mining techniques for a cluster of smart office buildings,” *Expert Systems with Applications*, vol. 42, pp. 4324-4338, 2015.
- [9] S. Taheri, A. Ahmadi, B. Mohammadi-Ivatloo și S. Asadi, „Fault detection diagnostic for HVAC systems via deep learning algorithms,” *Energy & Buildings*, vol. 250, 2021.
- [10] P. W. Tien, S. Wei, J. Darkwa, C. Wood și J. K. Calautit, „Machine Learning and Deep Learning Methods for Enhancing Building Energy Efficiency and Indoor Environmental Quality – A Review,” *Energy and AI*, vol. 10, 2022.
- [11] S. M. Namburu, M. S. Azam, J. Luo, K. Choi și K. R. Pattipati, „Data-Driven Modeling, Fault Diagnosis and Optimal Sensor Selection for HVAC Chillers,” *IEEE Transactions on Automation Science and Engineering*, vol. 4, 2007.
- [12] M. Gaur, S. Makonim, I. V. Bajic și A. Majumdar, „Performance Evaluation of Techniques for Identifying Abnormal Energy Consumption in Buildings,” *IEEE Access*, vol. 7, 2019.

- [13] B. Si, Z. Tian, X. Jin, X. Zhou, P. Tang și X. Shi, „Performance indices and evaluation of algorithms in building energy efficient design optimization,” *Energy*, vol. 114, pp. 100-112, 2016.
- [14] V. Valkiloroaya, Q. P. Ha și B. Samali, „Energy-efficient HVAC systems: Simulation-empirical modelling and gradient optimization,” *Automation in Construction*, vol. 31, pp. 176-185, 2013.
- [15] L. Yu, Di Xie, C. Huang, T. Jiang și Y. Zou, „Energy Optimization of HVAC Systems in Commercial Buildings Considering Indoor Air Quality Management,” *IEEE Transactions on Smart Grid*, vol. XX, 2018.
- [16] Z. Yang și B. Becerik-Gerber, „The coupled effects of personalized occupancy profile based HVAC schedules and room reassignment on building energy use,” *Energy and Buildings*, vol. 78, pp. 113-122, 2014.
- [17] S. Ley, J. L. Mathieu și R. K. Jain, „Performance of Existing Methods in Baseline Demand Response From Commercial Building HVAC Fans,” *Journal of Engineering for Sustainable Buildings and Cities*, vol. 2, pp. 1-12, 2021.
- [18] Z. Yang, A. Ghahramani și B. Becerik-Gerber, „Building occupancy diversity and HVAC (heating, ventilation, and air conditioning) system energy efficiency,” *Energy*, vol. 109, pp. 641-649, 2016.
- [19] X. Gao și A. Malkawi, „A new methodology for building energy performance benchmarking: An approach based on intelligent clustering algorithm,” *Energy and Buildings*, vol. 84, pp. 607-616, 2014.
- [20] M. Zekic-Susac, R. Scitovski și A. Has, „Cluster analysis and artificial neural networks in predicting energy efficiency of public buildings as a cost-saving approach,” *Croatian Review of Economic, Business and Social Statistics*, vol. 4, nr. 2, pp. 57-66, 2018.
- [21] T. G. Nikolaou, D. S. Kolokotsa, G. S. Stavrakakis și I. D. Skias, „On the Application of Clustering Techniques for Office Buildings’ Energy and Thermal Comfort Classification,” *IEEE Transactions on Smart Grid*, vol. 3, 2012.
- [22] J. Q. Wang, Yu Du, J. Wang, „LSTM based long-term energy consumption prediction with periodicity,” *Energy*, vol. 197, 2020.
- [23] S. Afzal, B. M. Ziapour, A. Shokfri, H. Shakibi și B. Sobhani, „Building energy consumption prediction using multilayer perceptron neural network-assisted models; comparison of different optimization algorithms,” *Energy*, vol. 282, 2023.
- [24] Z. Wang, Y. Wang, R. Zeng, R. S. Srinivasan și S. Ahrentzen, „Random Forest based hourly building energy prediction,” *Energy & Buildings*, vol. 171, pp. 11-25, 2018.
- [25] E. Mocanu, M. Gibescu, P. H. Nguyen și W. Kling, „Benchmarking algorithms for resource allocation in smart buildings,” *IEEE Eindhoven PowerTech*, 2015.
- [26] R. Z. Homod, „Analysis and optimization of HVAC control systems based on energy and performance considerations for smart buildings,” *Renewable Energy*, vol. 126, pp. 49-64, 2018.
- [27] B. Grillone, S. Danov, A. Sumper, J. Cipriano și G. Mor, „A review of deterministic and data-driven methods to quantify energy efficiency savings

- and to predict retrofitting scenarios in buildings," *Renewable and Sustainable Energy Reviews*, vol. 131, 2020.
- [28] K. Bamdad, M. E. Cholette, L. Guan și J. Bell, „Ant colony algorithm for building energy optimization problems and comparison with benchmark algorithms,” *Energy and Buildings*, vol. 154, pp. 404-414, 2017.
  - [29] M. Ala'raj, M. Radi, M. F. Abbod, M. Majdalawieh și M. Parodi, „Data-driven based HVAC optimization approaches: A Systematic Literature Review,” *Journal of Building Engineering*, vol. 46, 2022.
  - [30] V. Kirubakaran, C. Sahu, T. K. Radhakrishnan și N. Sivakumaran, „Energy efficient model based algorithm for control of building HVAC systems,” *Ecotoxicology and Environmental Safety*, vol. 121, pp. 236-243, 2015.
  - [31] A. Garces-Jimenez, J-M. Gomez-Pulido, N. Gallego-Salvador și A-J. Garcia-Tejedor, „Genetic and Swarg Algorithms for Optimizing the Control of Building HVAC Systems Using Real Data: A comparative Study,” *Mathematics*, vol. 9, 2021.
  - [32] C. Miller, A. Kathirgamanathan, B. Picchetti, P. Arjunan, J. Y. Park, Z. Nagy, P. Raftery, B. W. Hobson, Z. Shi și F. Meggers, „The Building Data Genome Project 2, energy meter data from the ASHRAE Great Energy Predictor III competition,” *Scientific Data*, vol. 7, 2020.
  - [33] A. Doherty, „Hark,” 29 April 2021. [Interactiv]. Available: <https://harksys.com/blog/what-are-energy-baselines-enb/>.
  - [34] W. Koehrsen, 27 December 2017. [Interactiv]. Available: <https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d>.