MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

APLICAȚIE PENTRU PREDICȚIA PREȚURILOR DIN DOMENIUL IMOBILIAR

LUCRARE DE LICENŢĂ

Absolvent: **Larisa-Elena DUD**

Coordonator științific: **Prof. dr. ing. Gheorghe SEBESTYEN-PAL**

2023

Cuprins

Capitolul 1. Introducere

1.1. Contextul proiectului

Piața imobiliară reprezintă un sector de interes major, care joacă un rol esențial în viața de zi cu zi a oamenilor și în economie. Achiziționarea sau vânzarea unei proprietăți reprezintă deseori o decizie importantă și financiar semnificativă pentru indivizi și companii. În acest context, evaluarea cât mai corectă a proprietăților și întelegerea pieței imobiliare sunt esențiale pentru a asigura o tranzacție eficientă și echitabilă.

Domeniul imobiliar se caracterizează prin complexitate și diversitate, deoarece implicațiile financiare și juridice sunt considerabile. Evaluarea proprietăților imobiliare presupune un proces care implică analiza unei game largi de factori, cum ar fi: localizarea, suprafața, calitatea construcției, facilitățile și infrastructura din jur, precum și tendintele pieței. Este esențial ca aceste evaluări să fie cât mai precise și obiective pentru a reflecta valoarea reală a proprietăților și pentru a asigura tranzacții echitabile între cumpărători și vânzători.

În trecut, estimarea prețurilor imobiliarelor s-a bazat în mare parte pe experiența agenților imobiliari sau a evaluatorilor specializați. Cu toate acestea, evoluția tehnologică și dezvoltarea inteligenței artificiale au oferit noi oportunitati și solutii în domeniul imobiliar.

Implementarea acestui proiect se concentrează pe utilizarea inteligenței artificiale pentru a dezvolta o aplicație în domeniul imobiliar prin care utilizatorilor li se vor furniza prețuri estimate ale proprietăților imobiliare, dar și alte informații relevante și actuale despre piața imobiliară.

Astfel, acest proiect reprezintă o soluție inovatoare în domeniul imobiliar, oferind beneficii semnificative pentru ambele parți implicate în tranzacțiile imobiliare și contribuind la imbunătățirea experienței utilizatorilor în acest sector dinamic, ajutându-i să ia decizii informate și să identifice oportunități avantajoase.

1.2. Motivația alegerii lucrării de licență

Motivația alegerii acestei teme pentru lucrarea de licență se bazează pe importanța și relevanța sectorului imobiliar în economie și viața cotidiană, tranzacțiile imobiliare reprezentând adesea cele mai semnificative achiziții financiare pe care indivizii le fac în viața lor, fie că este vorba despre achiziționarea unei locuințe sau investiții in proprietăți comerciale.

Dezvoltarea unei aplicații care poate estima prețurile imobilelor aduce multiple beneficii atât pentru cumpărători cât și pentru vânzători. Pentru cumpărători, o astfel de aplicație oferă o sursă de informații obiective și ajută la evaluarea corectă a valorii unei proprietăți în funcție de caracteristicile sale, aceștia având posibilitatea să ia decizii informate și să negocieze in mod eficient prețul. Pentru vânzători, aplicația poate reprezenta un instrument de evaluare a prețului corect al proprietății lor, oferindu-le posibilitatea de a obține un profit adecvat și de a-și maximiza rezultatele financiare.

Această aplicație poate servi și ca un instrument de sprijin în realizarea evaluărilor pentru agenții imobiliari și experții în evaluarea proprietăților. Aceștia pot

utiliza datele și predicțiile generate de aplicație pentru a obține o perspectivă mai cuprinzătoare asupra pieței imobiliare și pentru a oferi clienților estimări mai precise și credibile ale valorii proprietăților.

In concluzie, alegerea acestei teme pentru dezvoltarea unei aplicații de estimare a prețurilor imobilelor este motivată de dorința de a aprofunda cunostințele în domeniul inteligenței artificiale și de a cerceta domeniul imobiliar. Prin această inițiativă, contribuim la transformarea digitală a pieței imobiliare și obținem competențe valoroase pentru viitoarele noastre cariere în acest domeniu.

1.3. Structura lucrării

Documentul de față este structurat în opt capitole ample, fiecare abordând diferite aspecte importante legate de proiectul descris anterior și fiind împărțit în mai multe subcapitole.

În primul capitol, se prezintă și se descrie contextul actual în ceea ce privește partea tehnologică a domeniului imobiliar, precum motivația alegerii acestei teme și se realizează o scurtă introducere în ramura specifică a tehnologiei aplicate în domeniul imobiliar.

În al doilea capitol, se definesc obiectivele pe care sistemul prezentat își propune să le atingă în cadrul proiectului, precum și descrierea modului în care aceste obiective vor fi realizate. Totodată, se prezintă tipurile de utilizatori și cerințele funcționale și non-funcționale ale sistemului dezvoltat.

În cadrul capitolului trei, se efectuează o parcurgere a resurselor și a lucrărilor de specialitate realizate anterior, complementare temei abordate în domeniul imobiliar. Acest capitol oferă o imagine de ansamblu asupra cercetărilor și rezultatelor obținute în domeniul tehnologiilor aplicate în piața imobiliară.

În al patrulea capitol, se descrie designul general al soluției propuse, prezentând conceptele teoretice implementate și fundamentul teoretic care susține proiectul. De asemenea, se dezvoltă aspectele abstracte și teoretice relevante pentru înțelegerea arhitecturii software a proiectului.

În capitolul cinci, se documentează detaliile și implementarea aplicației în cadrul proiectului, asigurând astfel dezvoltarea și întreținerea ulterioare a acestuia. Sunt prezentate scheme reprezentative ale aplicației, precum arhitectura folosită și componentele implicate în cadrul sistemului dezvoltat.

Capitolul șase se concentrează pe partea de testare și validare a sistemului dezvoltat în cadrul proiectului, punând în evidență rezultatele obținute și relevanța acestora în contextul problemei propuse în piața imobiliară.

În al șaptelea capitol, se oferă un ghid de utilizare a sistemului dezvoltat pentru utilizatori, împreună cu procesul de instalare al aplicației. Se prezintă cerințele hardware și software necesare pentru o utilizare eficientă a sistemului.

Ultimul capitol al lucrării conține concluziile extrase din studiul prezentat, oferind, totodată, sugestii pentru posibile dezvoltări ulterioare ale aplicației. Aceasta permite sintetizarea rezultatelor și aportului personal adus în domeniul imobiliar, evidențiind contribuțiile și inovațiile aduse de proiect.

Capitolul 2. Obiectivele proiectului

2.1. Obiectivul principal

Obiectivul principal al proiectului constă în dezvoltarea unui model de predicție a prețurilor proprietăților imobiliare, care va fi integrat și gestionat într-o aplicație web complexă în care administratorul actualizează datele și reantrenează modelul, iar utilizatorii beneficiază de evaluarea automată a proprietăților pe baza unor caracteristici specifice. Scopul principal al proiectului este de a furniza o aplicație suport pentru procesul decizional privind achiziționarea sau vânzarea unei proprietăți. Pentru a atinge acest scop, trebuie îndeplinite următoarele obiective:

2.2. Obiective specifice

1. *Dezvoltarea unui model de predicție a prețului*

Unul dintre obiectivele principale ale proiectului este crearea unui model de inteligență artificială care să efectueze predicții precise ale prețurilor proprietăților imobiliare. Acest model va fi antrenat pe un set de date extins, care conține caracteristici relevante pentru evaluarea prețurilor, cat si pe un set de date actuale colectate manual de pe Internet. Modelul va fi proiectat astfel încât să utilizeze algoritmul regresiei liniare pentru efectuarea predicției. Scopul final este de a obține un model cu o acuratețe cat mai buna pentru a furniza utilizatorilor estimări cât mai precise.

2. *Colectarea de date actuale de antrenament*

Un alt obiectiv principal este colectarea si actualizarea setului de date folosit pentru antrenarea modelului. Pentru a obține predicții mai precise, este necesară extragerea datelor actuale de antrenament de pe diverse site-uri de anunțuri imobiliare. Aceste date actualizate vor contribui la îmbunătățirea acurateței predicțiilor și la reflectarea tendințelor actuale din piața imobiliară.

3. *Implementarea funcționalității de estimare a prețului*

O altă componentă importantă a proiectului este implementarea unei funcționalități care să permită utilizatorilor să introducă caracteristicile propriilor proprietăți și să primească o estimare a prețului acestora. Aceasta functionalitate va folosi modelul construit si va oferi utilizatorilor informațiile necesare pentru a evalua valoarea proprietăților lor și a lua decizii informate în privința tranzacțiilor imobiliare.

4. *Actualizarea constantă a datelor*

Pentru a menține precizia predicțiilor, este esențială actualizarea constantă a datelor folosite în cadrul aplicației. Aceasta implică monitorizarea și colectarea permanentă a informațiilor legate de prețurile și caracteristicile proprietăților imobiliare, astfel încât utilizatorii să beneficieze de estimări actualizate și relevante. Aceast lucru se poate face manual de catre o persoana autorizata, si la extragerea anunturilor se va tine cont de data de publicare pentru a asigura ca anunturile sunt actuale si nu exista deja in setul de date

5. *Realizarea unui sistem de înregistrare și autentificare*

Utilizatorii aplicației vor putea să își înregistreze un cont în aplicație și să se autentifice. De asemenea, se vor putea diferentia utilizatorii dupa rol in momentul autentificarii, ceea ce va permite limitarea accesului utilizatoriilor de tip client la anumite actiuni atribuite administratorului aplicatiei. Pentru început utilizatorii care sunt autentificați vor avea, in plus fata de cei ce nu sunt, capacitatea de a vizualiza statisticile care sunt prezentate in obiectivul secundar 3. În vreme ce, administratorul va avea acces la toate paginile si functionalitățile aplicației.

6. *Vizualizarea și filtrarea anunțurilor imobiliare*

Utilizatorii vor avea posibilitatea de a vizualiza anunțurile imobiliare actuale și de a naviga către pagina care conține anunțul original pentru mai multe informații. În plus, acestia vor putea aplica diferite filtre asupra anunțurilor pentru a le vizualiza strict doar pe cele care îndeplinesc caracteristicile căutate.

7. *Vizualizarea statisticilor relevante*

Administratorul și utilizatorii care dețin un cont și sunt autentificați vor putea să vizualizeze diferite reprezentări grafice cu referire la diverse aspecte ale pieței imobiliare și grafice care reflectă acuratețea modelui de predicție. Această funcționalitate va oferi utilizatorilor o perspectivă mai amplă asupra pieței imobiliare, dar ii va și ajuta să își dea seama cât de bună este predicția făcută de model.

8. *Implementare unei interfete intuitive*

Unul dintre cele mai importante obiective secundare este implementarea unei interfețe bine structurate, aceasta trebuie să fie cât mai intuitivă pentru a oferi o experiență prietenoasă utilizatorului. In urma definirii obiectivelor anterioare , interfata utilizator va trebui să conțină următoarele cate o pagina pentru fiecare dintre următoarele funcționalități: înregistrare, logarea în aplicție, vizualizarea și filtrarea anunțurilor, estimarea prețurilor, vizualizarea statisticilor si administrator.

Capitolul 3. Studiu bibliografic

Acest capitol reprezintă studiul bibliografic realizat pentru dezvoltarea acestei aplicatii, concentrându-se pe tema esențială a predicției prețurilor imobiliare. În cadrul acestei capitol, ne aprofundăm într-o multitudine de metode și modele folosite în predictia prețurilor proprietăților, de la abordări statistice tradiționale la algoritmi de învățare automată de ultimă oră. Analiza se extinde si la tehnicile de colectare a datelor de pe Internet, dezvăluind impactul transformator al web scraping și data mining în domeniul achiziției de date imobiliare.

De asemenea, vom examinăm aplicațiile existente, oferind perspective asupra modului în care aceste modele predictive și strategii de colectare a datelor sunt traduse în soluții practice și vom evidenția cu ce vine in plus aplicația noastră. Prin această investigație, cititorii vor obține o imagine de ansamblu a metodologiilor, provocărilor și aplicațiilor în evoluție care definesc domeniul predicției prețurilor anunturilor imobiliare.

În articolul [5] este prezentată o soluție pentru un model de predicție al prețului în domeniul imobiliar, acesta oferă o explorare cuprinzătoare a tehnicilor de clasificare și regresie liniară. Studiul s-a concentrat pe predicția prețurilor imobiliare folosind un set de date care cuprinde 79 de caracteristici pentru 1460 de case vândute în Ames, Iowa, între 2006 și 2010. Acest set de date bogat a permis explorarea atât a tehnicilor de regresie, cât și a celor de clasificare. Pentru a gestiona complexitatea setului de date, au fost aplicate tehnici de regularizare. Modelele de clasificare, cum ar fi Multinomial Naive Bayes și Multinomial Logistic Regression, au oferit o precizie de aproximativ 50%. Cu toate acestea, Support Vector Machine Classification (SVC) cu un nucleu liniar le-a depășit semnificativ, obținând o precizie de 63%. Alegerea finală, clasificarea aleatorie a pădurilor, a atins o precizie de 67%. În domeniul modelelor de regresie, regresia liniară a servit drept linie de bază, urmată de tehnicile de regularizare. Regresia liniară cu regularizare Lasso a realizat un RMSE redus, depășind valoarea de bază. Regresia vectorială de suport (SVR) cu un nucleu gaussian s-a dovedit a fi cea mai eficientă, depășind modelul de bază cu un RMSE mai mic de 0,5271. În timp ce modelele de clasificare oferă perspective unice, modelele de regresie, în special SVR cu un nucleu gaussian, demonstrează îmbunătățiri promițătoare în prezicerea prețurilor caselor. Aceste constatări servesc ca un punct de referință valoros pentru îmbunătățirea modelelor de predicție a prețurilor imobiliare,

Articolul [6] explorează metode de predicție a prețurilor imobiliare, concentrându-se în primul rând pe tehnicile de regresie și de stimulare. Regresia liniară servește ca model de bază pentru înțelegerea relației dintre variabile. Analiza de regresie multiplă este folosită pentru a examina asocierile dintre mai multe variabile, în timp ce regresia Lasso ajută la regularizare pentru a preveni supraadaptarea. Algoritmul Gradient Boosting, inclusiv variante precum XGBoost, AdaBoost și Gentle Boost, se remarcă ca o abordare puternică de învățare automată atât pentru sarcinile de regresie, cât și de clasificare. Acești algoritmi de stimulare îmbunătățesc acuratețea modelului predictiv prin combinarea rezultatelor de la clasificatoare „slabe”, făcându-le valoroase în predicția prețurilor imobiliare.

În articolul [7] se prezintă dezvoltarea unui model de predicție a prețurilor imobiliare folosind setul de date privind locuințele din Boston de la UCI Machine Learning Repository. Acest set de date conține informații despre 506 intrări legate de

case din suburbiile Bostonului, cu 14 caracteristici. Setul de date este apoi împărțit într-un set de date de antrenament de 70% și un set de date de testare de 30% pentru a facilita învățarea supravegheată. Articolul evidențiază importanța pașilor de pre-procesare pentru a permite învățarea eficientă a modelului. Valorile numerice sunt normalizate, iar valorile categoriale sunt codificate. Scalare este aplicată pentru a se asigura că caracteristicile sunt la o scară consecventă pentru performanța îmbunătățită a modelului. Pentru faza de dezvoltare a modelului se folosește algoritmul de pădure aleatoare, utilizând RandomForestClassifier de la Scikit-learn. Abordarea de învățare prin ansamblu ale pădurii aleatorii, combinată cu reglarea parametrilor, oferă un model capabil să prezică prețurile caselor cu o diferență de ±5, arătându-și eficacitatea în sarcinile de predicție a prețurilor imobiliare.

În articolul [8] se vorbeste despre colectarea datelor de pe Internet, mai exact web scraping. Web scraping este o tehnică de extragere eficientă a datelor de pe site-uri web, oferind mai multe metode precum expresii regulate, Beautiful Soup sau lxml în acest scop. Automatizează colectarea datelor, facilitând obținerea de informații structurate de pe web pentru diverse aplicații, inclusiv cercetarea de piață și analiza datelor. În acest articol se mai evidențiază si faptul că web scraping-ul joacă un rol crucial în business intelligence prin automatizarea colectării datelor. Ajută în sarcini precum urmărirea prețurilor concurenților, monitorizarea tendințelor pieței, chiar si a pieței imobiliare. Web scraping oferă numeroase avantaje, inclusiv colectarea de date eficientă și rentabilă în comparație cu metodele manuale. Oferă întreținere redusă, dă putere companiilor să obțină informații precise și în timp util, accelerând procesele de luare a deciziilor și obținând un avantaj competitiv.

În articolul [9] avem un studiul al carui obiectiv principal este de a prezice în mod eficient prețurile imobiliare din Taiwan. Pentru a realiza acest lucru, sunt aplicate și comparate două modele predictive, și anume Back Propagation Neural Network (BPNN) și Support Vector Regression (SVR). Studiul începe cu selecția variabilelor relevante pe baza cercetărilor anterioare și utilizează atât selecția în trepte, cât și metodele de încercare și eroare pentru a alege cele mai potrivite variabile. Constatările arată că SVR, în special atunci când a utilizat metoda de selecție a variabilelor de încercare și eroare, a avut cel mai bine rezultate cu o eroare procentuală medie absolută (MAPE) de 4,466% și un coeficient de determinare (R^2) de 0,8540. În special, rata de rediscount, masa monetară și prețul din ultima lună au apărut ca variabile comune atât pentru modelele BPNN, cât și pentru SVR.Această cercetare este semnificativă, deoarece abordează provocarea de a prezice prețurile imobiliare, care sunt cruciale pentru înțelegerea tendințelor economice și a bulelor. Modelele tradiționale de regresie au limitări în furnizarea de predicții precise, în timp ce rețelele neuronale precum BPNN și SVR oferă mai multă flexibilitate și o mai bună gestionare a datelor neliniare. În acest context, SVR se evidențiază ca metodă promițătoare datorită soluției sale optime unice, care previne suprainstalarea, făcându-l potrivit pentru predicția prețurilor imobiliare. Procesul de selecție a variabilelor din studiu evidențiază importanța ratei de rediscount, a masei monetare și a prețului lunii precedente ca factori cheie în prezicerea eficientă a prețurilor imobiliare în Taiwan.

În articolul [10] autorii prezintă o soluție pentru predicția prețului din domeniul imobiliar folosindu-se și de web scraping. Articolul se concentrează, în primul rând pe conceptul de web scraping, care este procesul de extragere sistematică a informațiilor de pe World Wide Web. Acesta explică faptul că web scraping se realizează prin crawler-uri web, programe de calculator sau script-uri automate care navighează pe web într-o manieră structurată pentru a localiza date specifice și a le compila pentru utilizare ulterioară. De asemnea, se elaborează diverse tehnici și instrumente folosite în web

scraping, cum ar fi Beautiful Soup, Selenium și Screen-scraper, subliniind rolul lor în transformarea datelor web nestructurate, de obicei în format HTML, în date structurate care pot fi stocate și analizate eficient. În plus, articolul abordează pe scurt importanța web scraping în analiza datelor imobiliare. Acesta subliniază faptul că web scraping este esențial în colectarea de date valoroase pentru predicția prețurilor imobiliare, mai ales atunci când informațiile legate de proprietate sunt dispersate pe diverse platforme web. Această tehnologie permite crearea de seturi de date cuprinzătoare prin agregarea datelor despre prețurile proprietăților, locații, tipologii și multe altele din diverse surse online.

În articolul [11] se vorbeste despre o aplicația Zillow.com ca exemplu de succes în industria imobiliară. Autorii evidențiază importanța satisfacerii așteptărilor utilizatorilor pentru experiențe rapide, intuitive și informative. Articolul subliniază că o aplicație de evaluare a proprietăților imobiliare ușor de utilizat, precum Zillow, poate avea un impact semnificativ asupra succesului unui site web, făcându-l relevant nu numai în domeniul imobiliar, ci și în toate industriile.

O aplicație foarte cunoscută din domeniul imobiliar, este Imobiliare.ro, aceasta este o platformă imobiliară online proeminentă din România care oferă o gamă largă de servicii pentru persoanele fizice care doresc să cumpere, să vândă sau să închirieze proprietăți. Utilizatorii pot naviga prin listele de proprietăți, pot vizualiza descrieri detaliate, imagini de înaltă calitate și chiar pot lua legătura cu agenții imobiliari și proprietarii de proprietăți direct prin intermediul platformei. Imobiliare.ro facilitează căutarea proprietăților în diverse regiuni ale României, făcându-l un instrument convenabil atât pentru cumpărători, cât și pentru vânzători din industria imobiliară. Cu toate acestea, aceasta aplicație nu oferă funcționalitatea de autoevaluare a unei proprietăți precum Zillow din Statele Unite.

În urma acestui studiu bibliografic, cercetarea noastră a aprofundat în diferite tehnici de ultimă oră pentru dezvoltarea unui model precis de predicție a proprietăților. Am descoperit importanța utilizării metodologiilor de scraping a datelor pentru a colecta date cuprinzătoare și în timp real despre proprietăți, un aspect critic în îmbunătățirea preciziei modelului nostru de predicție. În plus, analiza noastră a cuprins o evaluare aprofundată a mai multor aplicații existente pe piața imobiliară.

Examinând peisajul aplicațiilor imobiliare din România, am observat că multe platforme excelează în furnizarea de listări și conectarea cumpărătorilor cu vânzătorii. Cu toate acestea, proiectul nostru iese în evidență prin introducerea unei caracteristici inovatoare, mai exact capacitatea utilizatorilor de a-și evalua instantaneu și independent proprietățile. Această adăugare unică nu numai că simplifică procesul de luare a deciziilor pentru cumpărătorii și vânzătorii de proprietăți, dar și simplifică experiența generală a utilizatorului pe piața imobiliară din România.

Capitolul 4. Analiză și fundamentare Teoretică

În acest capitol se vor prezenta principiile și conceptele teoretice care pun bazele proiectării și implementării aplicației menite să îndeplinească obiectivele propuse. În plus, se dorește motivarea alegerilor făcute în ceea ce privește algoritmi utilizați și familiarizarea cititorului cu funcționalitățile soluției.

4.1. Modelul de predicție a prețului

4.1.1. Seturile de date utilizate

Sistemul creat lucreaza cu doua seturi de date, un set de date de dimensiune mare care conține anunturi mai vechi și un set de date actual construit manual prin colectarea de anunturi de pe Internet. Setul de date folosit are o foarte mare importanță în dezvoltarea si evaluarea unui model inteligent, motiv pentru care am folosit mai multe tipuri de date pentru a pune în evidență comportamentul modelului in funcție de diversitatea datelor.

Un set de date [1] contine caracteristicile unor proprietati de vanzare din Bucuresti. Cu toate ca aceste date vechi furnizeaza doar o perspectiva istorica asupra preturilor, datorita faptului ca setul de date este destul de mare, continand pana la 8321 de anunturi putem obtine un model mult mai bun de predictie a pretului, chiar daca acesta nu este in concordanta cu preturile pietei actuale.

Cel de-al doilea set de date este creat manual prin colectarea de anunturi imobiliare din Cluj-Napoca de pe site-ul Blitz Imobiliare. Acest set de date are dimensiuni mai mici, pana la momentul actual colectandu-se 493 de anunturi, ne asteptam ca acest aspect sa influenteze in mod negativ acuratețea predicției, dar ne va ajuta să oferim o imagine mai actuală asupra pieței imobiliare din aceasta zonă.

4.1.2. Preprocesarea datelor

Preprocesarea datelor este un pas foarte important in antrenarea modelelor de predictie, care implica operații de curățare, transformare si organizare pentru a face datele potrivite pentru sarciniile de analiza sau invatare automata. Astfel, ne vom asigura ca vom antrena modelul de predictie a pretului pe un set de date exacte, consecvente si cu un format adecvat modelelor predictive.

Unul dintre aspectele importante cand vine vorba de analiza seturilor de date il reprezinta provenineta acestora, de aceea un prim pas in procesul de preprocesare a datelor este acela de a ne asigura ca datele sunt relevante pentru analiza pretului din domeniul imobiliar, ca acestea provin din surse sigure si ca sunt suficient de multe pentru o predicție cât mai bună. Putem spune că seturile de date folosite pentru dezvoltarea acestui proiect indeplinesc aceste criterii având în vedere faptul ca setul de date cu anunțuri din București este preluat de pe un site certificat și pare sa conțină anunțuri realiste, iar setul de date cu anunțuri din Cluj-Napoca a fost construit colectând date de pe un site dedicat publicării acetui tip de anunțuri.

Al doilea pas presupune gestionarea duplicatelor, a valorilor lipsă si reducerea setului de date. Pentru ștergerea înregistrarilor duplicate vom identifica și elimina instanțele identice din setul de date, care pot apărea din erori de colectare sau integrare a datelor. În ceea ce privește valorile lipsă, putem alege să eliminam inregistrarile cu

date lipsa, sa inlocuim aceste valori cu medii, mediane sau sa folosim tehnici avansate precum imputarea datelor lipsa. Pentru acest pas folosim functii speciale pentru analiza datelor, iar in cazul coloanelor irelevante care au foarte multe valori lipsa le vom elimina, iar pentru coloanele relevante cu valori null vom atribui valoarea 0 pentru acestea. Reducerea setului de date se va face identificând si eliminând coloanele care nu contribuie semnificativ la puterea de predicție a modelului

În urma analizării datelor pentru Cluj-Napoca observăm în Figura 4-1 că unele coloane relevante conțin niste valori de null, mai exact: Nr. Balcoane, Locuri de parcare, pentru acestea vom inlocui valoare null cu valoarea 0 , iar coloana Facilități o vom sterge deoarece nu conține nicio valoare. Pentru a reduce setul de date vom elimina coloanele urmatoare: tip compartiment, tip imobil, dotări, tip finisaj, materiale contrucție, modalitate vânzare, data anunt, link anunt, acestea nefiind relevante pentru fromare modelului de predicție.

```
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Nr. camere:             457 non-null    object
 1   Suprafață utilă         457 non-null    float64
 2   Nr. băi:                457 non-null    int64
 3   Nr. balcoane:           343 non-null    float64
 4   Etaj:                   457 non-null    int64
 5   Locuri de parcare:      312 non-null    float64
 6   Boxă la subsol:         457 non-null    object
 7   Tip compartimentare:    457 non-null    object
 8   Vechime apartament:     457 non-null    object
 9   Tip imobil:             457 non-null    object
 10  Dotări:                 457 non-null    object
 11  Anul construcției:      457 non-null    int64
 12  Tip finisaj:            457 non-null    object
 13  Materiale construcție:  379 non-null    object
 14  Modalitate vânzare:     407 non-null    object
 15  Facilități:             0 non-null      float64
 16  Data anunt              457 non-null    object
 17  Cartier                 457 non-null    object
 18  Pret/mp                 457 non-null    int64
 19  Link Anunt              140 non-null    object
```

Figura 4-1: Numărul valorilor non-null și tipul fiecărei coloane din setul de date pentru Cluj-Napoca

Pentru setul de date din București observăm în Figura 4-2 că intâlnim aceeași situație. Vom inlocui valorile null de pe coloanele: garages_count, bathrooms_count, kitchens_count, parking_lots_count, balconies_count cu valoarea 0 , iar dintre coloana useful_surface si built_surface vom alege coloana built_surface pentru că are mai putin valori lipsa. După înlocuirea valorilor null de pe coloanele relevante vom elimina coloanele care nu au un rol important in predictia pretului, acestea fiind: id, location, partitioning, build_surface, level, seller_type, type, comfort, real_estate_type, heigh_regim. Pentru a nu influența predicția in mod negativ si deoarece sunt in număr mi, valorile null din coloana useful_surface le vom elimina.

```
RangeIndex: 8320 entries, 0 to 8319
Data columns (total 21 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   8320 non-null   int64
 1   location             8320 non-null   object
 2   location_area        8320 non-null   object
 3   seller_type          8320 non-null   object
 4   type                 8320 non-null   object
 5   partitioning         8146 non-null   object
 6   comfort              8065 non-null   object
 7   price                8320 non-null   int64
 8   rooms_count          8319 non-null   float64
 9   useful_surface       8246 non-null   float64
 10  built_surface        7399 non-null   float64
 11  construction_year    8180 non-null   float64
 12  real_estate_type     8281 non-null   object
 13  height_regime        8167 non-null   object
 14  level                8319 non-null   object
 15  max_level            8118 non-null   float64
 16  kitchens_count       7157 non-null   float64
 17  bathrooms_count      8002 non-null   float64
 18  garages_count        500 non-null    float64
 19  parking_lots_count   2009 non-null   float64
 20  balconies_count      6218 non-null   float64
```

Figura 4-2: Numărul valorilor non-null și tipul fiecărei coloane din setul de date pentru București

Un alt pas in preprocesarea datelor il constituie codificarea valorilor categorice. In cazul in care avem variabile categorice precum “cartier”, “boxa subsol”, adica variabile care influenteaza valoarea pretului dar nu au valori numerice, trebuie sa le codificam intr-o forma numerica pentru a putea fi folosite in regresia liniara.

In urma analizei vizuale Figura 4-4 si nu numai vom transpune valori precum “Garsonira” pentru coloana “nr. Camere” in valoarea 1 si pentru coloana “Boxa subsol” valorii “Nu” ii vom atribui 0, iar valorii “Da” ii vom atribui 1.

Nr. camere:	uprafață ut	Nr. băi:	r. balcoan	Etaj:	uri de parc	Boxă la subsol:
3	76	1	2	4	3	Nu
2	58	1		2	1	Nu
2	44	1	1	1		Nu
3	70	2	1	3		Nu
2	54	1	1	4	1	Nu
3	66	2	1	2	1	Nu
3	67	1	1	1	1	Nu
2	57	1	1	3	1	Nu
Garsonieră	33	1		4		Nu

Figura 4-3: Înregistrări din setul de date pentru Cluj-Napoca

Ultimul pas il constituie împărțirea datelor in setul de antrenare si cel de testarea. Setul de antrenare e constituit din 80% din setul de date si va fi folosit pentru construirea modelului, in timp ce setul de testare e constituit din 20% din setul de date si va fi folosit pentru a evalua performanta acestuia.

4.1.3. Algoritmi de codificare a locațiilor

In domeniul imobiliar , zona in care este situata o proprietate dintr-un anunt are un impact foarte mare asupra pretului, de aceea avem nevoie de un algoritm cat mai bun pentru a include si coloana destinata locatiei sub forma unui cod in modelul de predictie.

Un prim algoritm care a fost incercat se bazeaza pe atribuirea codului unei locatii pe baza ordonarii crescatoare a locatiilor in functie de pret. Mai intai se grupeaza toate anunturile dupa locatie si se realizeaza media pretului pe unitatea de suprafata pentru fiecare locatie. Dupa calcularea mediilor se ordoneaza locatiile crescator in functie de pretul mediu si li se vor atribui coduri de la 1 pentru locatia cu cel mai mic pret mediu pana la numarul total de locatii pentru locatia cu cel mai mare pret mediu.

Al doilea algoritm incepe tot cu gruparea inregistrarilor dupa locatie si calcularea pretului mediu pe unitatea de suprafata pentru fiecare locatie. Apoi se va calcula media pretului mediu pe fiecare locatie si se va folosi in calcularea codului impartind valoarea medie pe locatie la media generala. Prin utilizarea acestui algoritm de codificare, obținem o reprezentare numerică a locației care conține informații sugestive despre costul imobiliar în diferitele cartiere.

Deși primul algoritm poate fi mai simplu în comparație cu al doilea, in construirea modelului l-am ales pe cel de al doilea, deoarece are avantajul că păstrează informații despre relația de ordine dintre prețurile medii ale locațiilor. Deci, într-un model de regresie liniară, algoritmul poate ajuta la capturarea variațiilor de preț și a subtilităților între locații, ceea ce poate duce la predicții mai precise ale prețurilor imobiliare.

4.1.4. Utilizarea regresiei liniare pentru predictia pretului

Regresia liniară este o tehnică de modelare statistică utilizată pentru a prezice valoarea unei variabile pe baza uneia sau mai multor variabile. Variabila care se doreste sa fie prezisa se numeste variabila dependenta, iar celelalte variabile utilizate in predictie se numesc variabile independente. Aceasta metoda estimeaza coeficientii ecuatiei liniare, implicand una sau mai multe variabile independente. Coeficienții din ecuația de regresie indică direcția și magnitudinea impactului fiecărei variabile independente asupra variabilei dependente.

Ecuația unui model de regresie liniară simplă este dată de:

$$\mathbf{Y = \beta 0 + \beta 1 X + \varepsilon},$$

unde Y este variabila dependentă, X este variabila independentă, β0 și β1 sunt coeficienții de regresie, iar ε este termenul de eroare.

Regresia liniara este o alegere buna pentru implementarea modelului de predictie a pretului, deoarece ofera o formula matematica usor de interpretat si asigura antrenarea rapida a modelului. Într-o aplicație specifică în industria imobiliară, regresia liniară este utilizată pentru a realiza predicții privind prețurile proprietăților. Aici, variabila dependentă este prețul proprietății, iar variabilele independente sunt reprezentate de diverse caracteristici ale proprietății care ar putea influența prețul.

Pentru setul de date din Cluj-Napoca, variabilele independente sunt urmatoarele: numărul de camere, suprafața, numărul de băi, numărul de balcoane, etajul, numărul de locuri de parcare, existența unei boxe în subsol, codificarea cartierului și vechimea clădirii. Ecuatia acestui model va arata in felul urmator:

$$\text{Preț proprietate} = \beta 0 + \beta 1 \times \text{Nr. Camere} + \beta 2 \times \text{Suprafața} + \ldots + \varepsilon$$

Coeficienții reflectă impactul fiecărei variabile independente asupra prețului proprietății. De exemplu, coeficientul asociat numărului de camere ar putea arăta cât de mult crește prețul pe măsură ce numărul de camere crește. Prin ajustarea coeficienților

în funcție de datele disponibile, modelul poate prezice prețurile estimative ale proprietăților pe baza caracteristicilor lor.

4.1.5. Utilizarea Random Forest pentru predictia pretului

Random Forest (Pădurea Aleatoare) este o tehnică de învățare automată utilizată pentru atât probleme de regresie, cât și de clasificare. Acesta este un tip de algoritm ensemble, ceea ce înseamnă că combină mai multe modele mai simple pentru a obține un rezultat mai robust și mai precis.

Random Forest este compus din mai multe arbori de decizie. Fiecare arbore de decizie este construit pe un set diferit de date, ales aleatoriu din setul de antrenare. Fiecare arbore de decizie produce o predicție independentă, iar rezultatul final al Random Forest este o medie (în cazul regresiei) sau o majoritate (în cazul clasificării) a predicțiilor individuale ale arborilor. O caracteristică cheie a Random Forest este că poate gestiona atât caracteristicile continue, cât și cele categorice fără a necesita preprocesare extensivă.

În cazul predicției prețurilor în imobiliare, Random Forest poate fi folosit în mod similar cu regresia liniară pentru a estima prețurile proprietăților. Fiecare arbore de decizie din Random Forest va lua în considerare diverse caracteristici ale proprietăților și va prezice prețurile individuale, iar rezultatul final va fi o medie a acestor predicții.

Caracteristici ale proprietăților (variabile independente) precum numărul de camere, suprafața utilă, numărul de băi, etajul, existența balcoanelor, vechimea clădirii etc. pot fi folosite pentru construirea Random Forest. Coeficienții din ecuația regresiei liniare sunt înlocuiți aici de importanța caracteristicilor determinate de Random Forest.

Astfel, Random Forest poate să ofere o predictibilitate mai bună decât un singur model de regresie liniară, deoarece abordează mai bine variabilitatea din date prin utilizarea unor multiple modele.

4.1.6. Utilizarea clasificatorului Support Vector Machines pentru predictia pretului

Support Vector Regression (SVR) este o variantă a algoritmului Support Vector Machine (SVM) adaptată pentru probleme de regresie. În loc să încerce să separare clasele, cum face SVM în clasificare, SVR încearcă să găsească o linie (pentru regresie liniară) sau o curbă (pentru regresie neliniară) care se potrivește cel mai bine cu datele de antrenare și minimizează erorile de predicție.

SVR implică găsirea unei linii sau a unei curbe care maximizează distanța (marginea) dintre punctele de antrenare și linia/curba de regresie. Marginea este definită de două puncte de suport, care sunt cele mai apropiate puncte de linie/curbă. Aceste puncte de suport sunt cele care influențează direct construirea liniei/curbei de regresie.

4.1.7. Compararea algoritmilor de predictie

Regresia liniara este simpla si eficienta din punct de vedere computational, insa funcționează bine atunci cand exista o relație liniară intre caracteristicile de intrare si variabila tinta, ceea ce in domeniul imobiliar nu se prea întampla. De asemenea, regresia liniara nu gestioneaza bine valorile aberante prezente cu preponderență in setul de date colectat.

Random Forest este foarte flexibil, funționeaza bine chiar daca relația dintre caracteristici si preț este una neliniara si gestioneaza cu succes si valorile aberante. Ca si dezavantaje, acest algoritm poate fii mai greu de interpretat.

SVR este eficient atunci cand sunt implicate date de dimensiuni mari si poate gestiona relații neliniare prin utilizarea funțiilor nucleu. Ca si dezavantaje, este mai putin interpretabil si necesita o reglare atenta a hiperparametrilor.

4.1.8. Algoritmul de detectare a anomaliilor

Detecția anomaliilor este procesul de identificare a punctelor de date sau a observațiilor care se abat semnificativ de la comportamentul general sau normal al unui set de date. Aceste puncte, numite "anomalii" pot fi cauzate de erori ale datelor, evenimente rare sau neașteptate, sau pot semnala situații de interes în sine. Acest proces are scopul de a identifica aceste anomalii pentru a putea investiga și lua măsuri corespunzătoare în funcție de contextul problemei.

Algoritmii de detecție a anomaliilor implică în general următorii pași:

- **Definirea normalității**: Este necesar să se stabilească ce este considerat "normal" în setul de date. Acest lucru poate fi realizat prin analiza distribuției datelor sau a comportamentului obișnuit.
- **Alegerea metodei**: Există numeroase metode de detectare a anomaliilor, cum ar fi bazate pe statistici, distanțe, modele matematice sau modele de învățare automată. Fiecare metodă are propriile sale avantaje și limitări.

- **Calcularea deviației**: Algoritmul măsoară cât de mult se abate fiecare punct de date de la normalitate. Această deviație poate fi calculată folosind metrici precum distanțe euclidiene, diferențe de medie, deviație standard sau alți indicatori specifici.
- **Stabilirea pragului**: Un prag este stabilit pentru a delimita între anomaliile și datele normale. Punctele care depășesc acest prag sunt considerate a fi anomalii.
- **Identificarea anomaliilor**: Punctele de date care trec peste pragul sunt identificate și marcate ca anomalii. Acestea pot fi considerate pentru ulterioră investigație.
- **Interpretarea rezultatelor**: Rezultatele obținute sunt analizate pentru a înțelege natura și cauza anomaliilor. Uneori, o anomalie poate fi justificată de circumstanțe speciale sau erori în date.
- **Acțiuni următoare**: În funcție de context, se pot lua măsuri pentru a rezolva sau investiga cauzele anomaliei. Aceste măsuri pot varia de la simpla corecție a datelor până la acțiuni precum stergerea acetsora.

4.1.9. Principalii pași în construirea unui model de predicție

În acest subcapitol, vom evidenția principalii pași care stau la baza construirii modelului de predicție. Fluxul de lucru în vederea dezvoltării modelului se poate observa în Figura 4-4.

Pentru început vom avea nevoie de un set de date care conține anunțuri imobiliare, pe care vom aplica diverși pași de preprocesare, pentru a ne asigura ca avem date relevante pentru efectuarea predicție.

Apoi se va realiza construirea efectivă a modelului, acest pas cuprinde alegerea algoritmilor de invățare folosiți. Dupa construirea modelului, acesta va fi antrenat pe un set de date dedicat antrenării.

În cele din urmă, dupa ce avem un model antrenat se vor realiza predicții pe date noi, astfel încât se va putea determina acuratețea predicțiilor pe date pe care modelul nu le cunoaște.

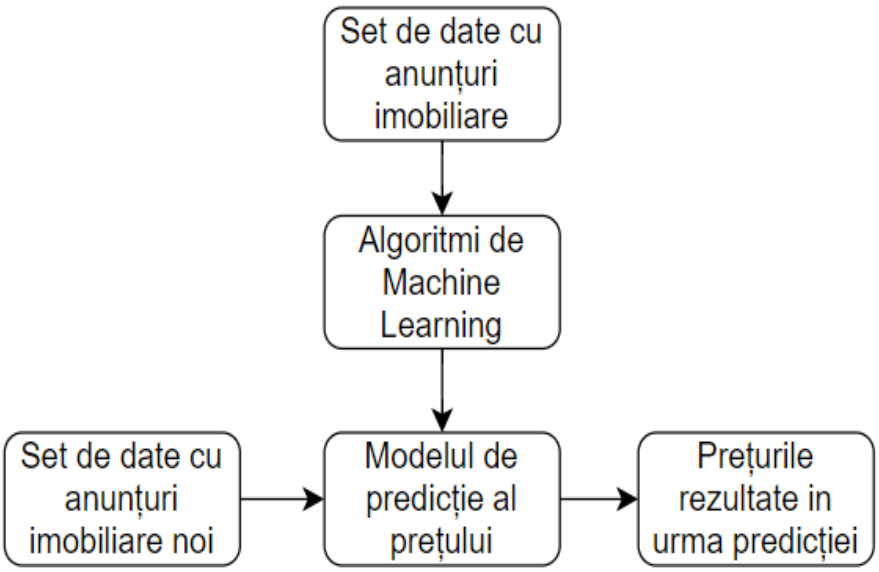


Figura 4-4: Fluxul de lucru pentru antrenarea modelului de predicție

4.2. Colectarea datelor de pe internet

În era informației digitale, colectarea datelor de pe internet a devenit esențială pentru cercetători, analiști și dezvoltatori de software. O metodă eficientă de extragere a datelor din paginile web este data scraping, o tehnică care implică extragerea automată a datelor din paginile web pentru analiză ulterioară. Una dintre cele mai utilizate abordări în data scraping implică utilizarea driverului Chrome.

Am ales să colectăm date de pe site-ul "Blitz Imobiliare". Acest site este cunoscut pentru a oferi informații detaliate despre proprietăți disponibile, inclusiv caracteristici precum numărul de camere, suprafață, etaj, cartier, preț etc. De asemenea, site-ul se actualizează frecvent, asigurând astfel date relevante pentru analiză.

Prin alegerea unei surse precum "Blitz Imobiliare", ne asigurăm că avem acces la date de calitate și actualizate, oferindu-ne o bază solidă pentru analiză. Utilizând tehnica de data scraping cu driverul Chrome, putem colecta eficient datele relevante și le putem transforma în informații valoroase pentru analiza noastră în contextul regresiei liniare aplicate pe piața imobiliară.

Pasii pentru procesul de colcetare al anunturilor sunt urmatorii:

- Instalarea Driverului Chrome: Descarcă și instalează driverul Chrome corespunzător versiunii browserului tău Chrome.
- Selectarea Elementelor: Identifică elementele HTML pe care dorești să le extragi din pagina web. Acestea pot fi selecționate folosind selecția CSS sau XPath.

- Configurarea în Python: Utilizează o bibliotecă precum Selenium în Python pentru a interacționa cu driverul Chrome. Încarcă pagina, găsește elementele dorite și extrage-le.
- Preluarea Datelor: După identificarea elementelor, preia valorile lor (text, linkuri, imagini etc.) și transformă-le în formatul dorit (CSV, JSON etc.).
- Iterație și Navigare: Pentru paginile web cu mai multe pagini sau cu conținut generat dinamic, va trebui să iterezi prin pagini și să simulezi acțiuni precum scroll sau click pentru a prelua toate datele dorite.

4.3. Cazurile de utilizare ale aplicației

4.3.1. Cazuri de utilizare pentru utilizatorul de tip client

În cadrul acestei aplicații, un utilizator de tip client poate face următoarele acțiuni:

- Dacă utilizatorul nu are deja un cont, acesta poate să se înregistreze introducând numele, prenumele, adresa de email si o parola puternică.
- Dacă utilizatorul deține deja un cont in aplicație acesta poate sa se autentifice introducând adresa de email si parola.
- Atât utilizatorii autentificați cât si cei neautentificati pot sa vizualizeze anunturile colectate.
- Atât utilizatorii autentificați cat si cei neautentificati pot sa filtreze annunturile dupa urmatoarele criterii; numar de camere, cartier si compartimentare.
- Utilizatorii autentificati pot sa isi evalueze o proprietate, prin introducerea caracteristicilor specifice acesteia, rezultatul fiind pretul prezis de model
- Utilizatorii autentificati pot vizualiza diferite statistici referitoare la anunturile imobiliare si la modelul de predictie.

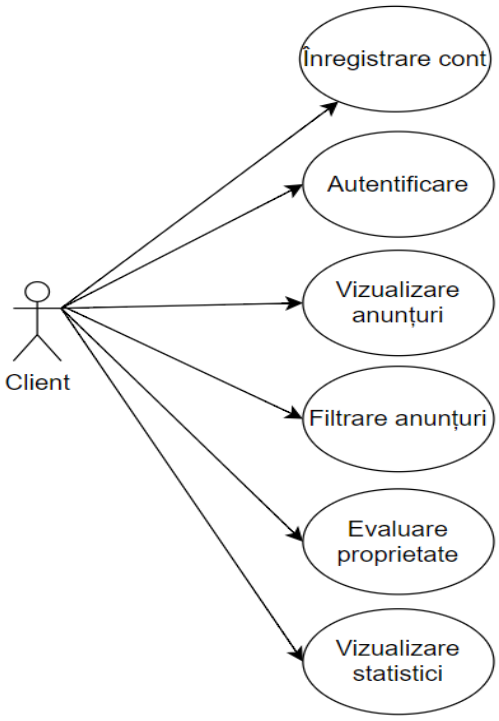


Figura 4-5: Diagrama cazurilor de utilizare ale clientului

4.3.2. Cazuri de utilizare pentru utilizatorul de tip administrator

Administratorul va avea deja un cont inregistrat in aplicatie si in urma autentificarii va avea acces la actiunile specifice acestui tip de utilizator.

Un administrator poate sa faca toate actiunile pe care un client le poate face, mai putin actiunea de inregistrare cont.

În plus, acesta mai poate face urmatoarele actiuni:

- Administratorul poate sa actualizeze seturile de date ce contin anunturi colectate de pe Internet.
- Administratorul poate reantrena modelul de predictie

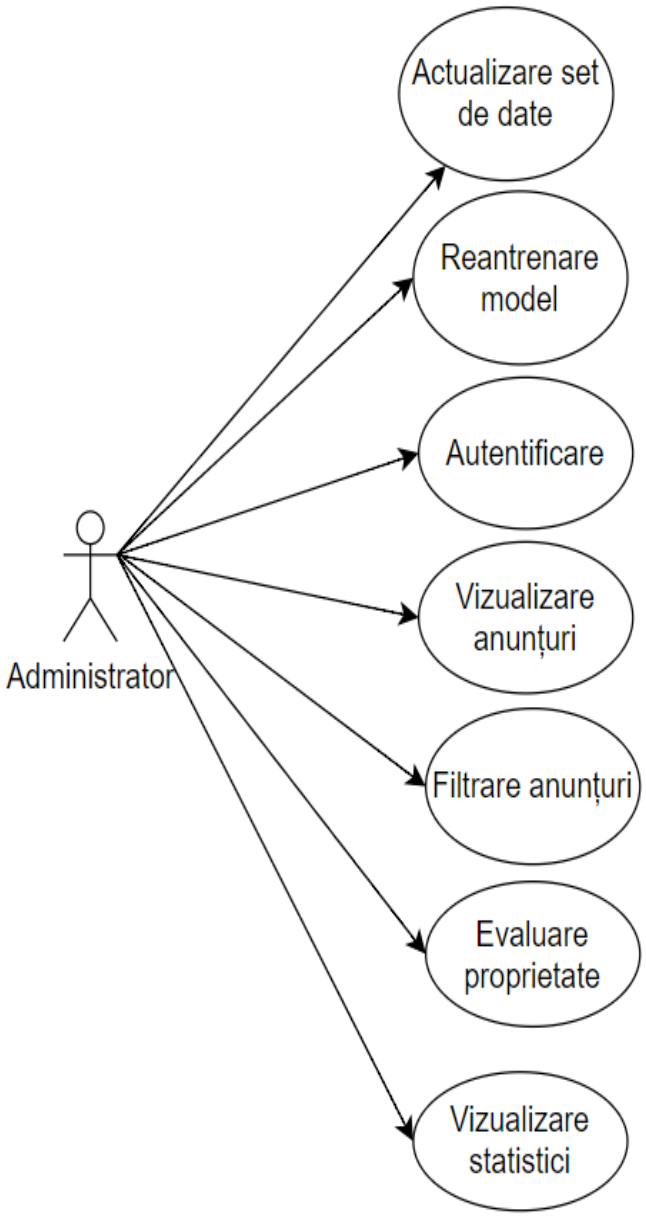


Figura 4-6: Diagrama cazurilor de utilizare ale administratorului

4.4. Medii de programare

Un aspect esential în realizarea aplicatiei il constituie alegerea limbajului de programare, deoarece fiecare limbaj e diferit si ofera diferite lucruri si acesta sta la bazele înțelegerii cat mai rapide a proiectului. Astfel, alegerile facute au fost unele strategice in ceea ce priveste toate componentele aplicației. Pentru partea de inteligenta artificiala si web scraping, am optat pentru Python, un limbaj recunoscut pentru versatilitate si eficienta in acest domeniu. Totodata, Python vine cu o gama larga de librarii specializate si o sintaxa concisa, cee ace faciliteaza implementarea de algoritmi complecsi si exploatarea solutiilor de invatare automata.

În ceea ce privește gestionarea utilizatorilor, am ales sa folosesc Java in combinație cu framework-ul Spring. Java este cunoscut pentru fiabilitate si securitatea sa, iar Spring furnizeaza un cadru de lucru solid pentru dezvoltarea aplicatiilor web scalabile si robuste. Aceasta alegere asigura o gestionare eficienta a utilizatorilor si un mediu sigur pentru interactiunea cu platforma.

Pentru partea de frontend, am ales sa folosesc limbajul JavaScript impreuna cu biblioteca React. JavaScript este un limbaj de programare destinate dezvoltarilor de interfete utilizator, iar acesta folosit impreuna cu React ofera o dezvoltare modulara si eficienta pentru construirea interfetelor web creand astfel experiente de utilizare intuitive. Tot la partea de frontend am folosit CSS si SCSS pentru stilizarea interfetei.

In ceea ce priveste partea de server a componentei inteligente, am utilizat framework-ul Flask in Python. Flask ofera un mediu usor de utilizat si flexibil pentru gestionarea serviciilor web.

Fiecare limbaj si tehnologie selectate au fost evaluate minutios pentru a se potrivi cerintelor fiecarei componente in parte si pentru a asigura o dezvoltare si intretinere usoara pe termen lung.

Capitolul 5. Proiectare de detaliu și implementare

In capitolul acesta, vom documenta detaliat aplicatia dezvoltata cu scopul de a contribui la o dezvoltare ulterioara si intrentinere mai eficienta. Viitorii dezvoltatori vor putea identifica si intelege modulele principale ale aplicatiei mai usor . Se vor prezenta schema generala a aplicatie, diagrame de clase si implementarea metodelor importate pentru a concretiza modul de functionare al aplicatiei. De asemenea, se vor explica si alegerile tehnologice impreuna cu motivele din spatele acestora.

5.1. Arhitectura generală a aplicației

Solutia implementata pentru acest proiect reprezinta un sistem compus din trei aplicatii, fiecare avand responsabilitatii specifice si contribuind la functionarea generala a aplicatiei. Scopul alegerii unei arhitecturi modulare a fost acela de a separa functionalitatile aplicatiei asigurand astfel o dezvoltare ulterioara si intretinere mult mai optima, cat si pentru a acoperii cat mai multe cunostinte acumulate .

Prima aplicatie se ocupa cu implemntarea interfetei web reprezentand partea vizibila a aplicatiei. Celelalte doua functioneaza ca servere si au roluri specifice in cadrul sistemului.

Aplicatia dezvoltata in Java se ocupa cu gestionarea utilizatorilor, mai exact cu implementarea functionalitatilor de inregistrare si autentificare. Aceasta stocheaza datele despre utilizatori intr-o baza de date MySql.

Aplicatia dezvoltata in Python reprezinta componenta inteligenta a sistemului, ea fiind responsabila cu predictia preturilor si colectarea anunturilor de pe Internet.

Aceste trei aplicații funcționează într-un mod integrat și interacționează între ele pentru a oferi o experiență completă și funcțională utilizatorilor. Astfel, prin separarea responsabilităților și utilizarea tehnologiilor și limbajelor de programare adecvate, această arhitectură permite o dezvoltare modulară și scalabilă a aplicației,

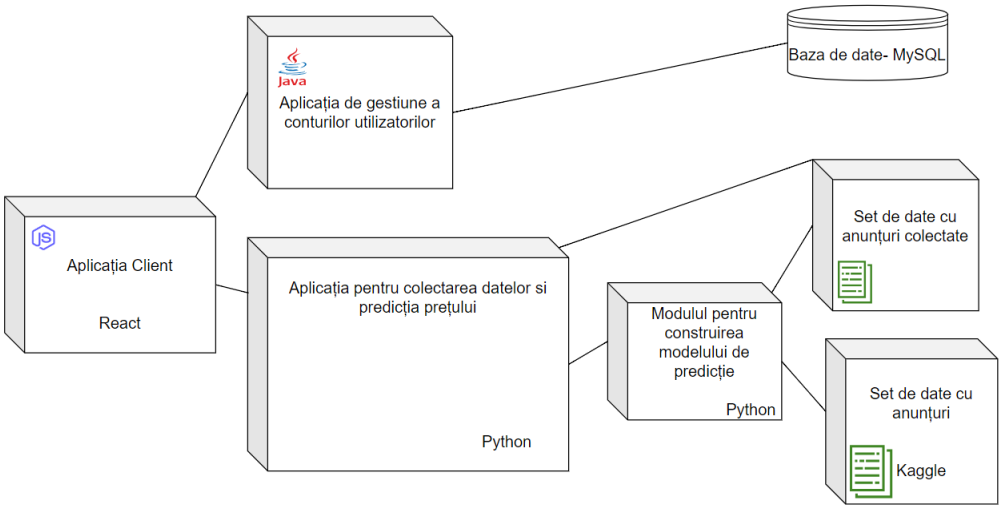


Figura 5-1: Arhitectura generala a aplicației

5.2. Structura programului

5.2.1. Diagrame de pachete

Diagrama de pachete pentru aplicatia Web este prezentată in Figură 5-2.

Pachetul hooks este responsabil pentru gestionarea hook-urilor personalizate. Hooks-urile in React sunt niste functii speciale care permit componentelor functionale sa aiba stari interne si sa utilizeze functionalitati specifice clasei, fara a fi nevoie sa fie convertite in componente bazate pe clase. Un hook personalizat, pe de alta parte, este un hook creat de dezvoltatori din mai multe hooks predefinite.

Pachetul constants contine fisierele care contin constante utilizate in intreaga aplicatie. Constantele joaca un rol important in furnizarea de valori fixe care raman consistente in diferitele parti ale codului. Separarea acestora intr-un pachet dedicat, le face mai usor de gestionat si actualizat atunci cand este necesar.

Pachetul pages contine pachete individuale corespunzatoare fiecarei pagini a aplicatiei, in interiorul fiecarui pachet de pagina se afla componenta paginii, care defineste structura si functionalitatea acelei pagini, precum si fisierul pentru stilizare asociat.

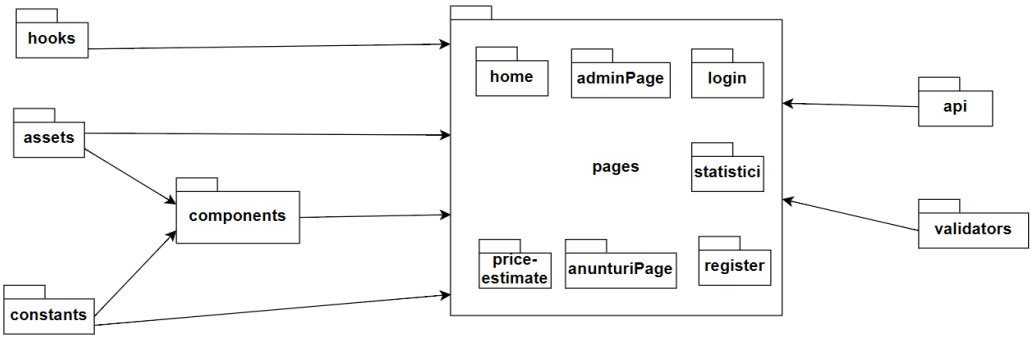
Pachetul api contine fisierele responsabile pentru configurarea comunicarii dintre frontend si backend.

Pachetul assets cuprinde diverse resurse utilizate in cadrul aplicatiei pentru a oferi continut vizual, acestea pot fii: imagini, fonturi, videoclipuri, sunete, etc.

Pachetul components cuprinde diverse componente reutilizabile si de uz general in celelalte componente ale aplicatiei.

Pachetul validators contine diversi validatori responsabili cu validrea datelor introduse de utilizator. Acești validatori efectuează verificări și aplică reguli sau constrângeri specifice asupra introducerii utilizatorului, asigurând integritatea și acuratețea datelor.

Toate aceste pachete se afla in pachetul src, care reprezinta pachetul radacina si include componenta principala a aplicatiei.

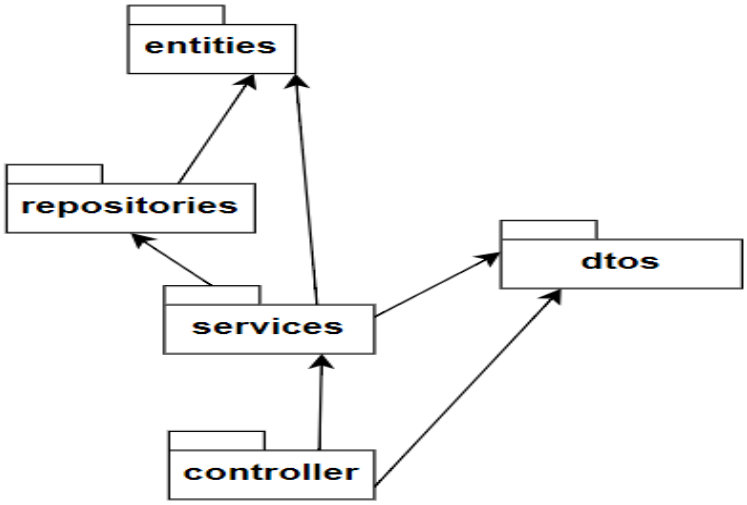


Figură 5-2: Diagrama de pachete a aplicației Web

Diagrama de pachete pentru aplicatia de gestiune a utilizatorilor

Pachetul PricePrediction este pachetul radacina si cuprinde totalitatea pachetelor si clasa principala a aplicatie.

Pachetul entities contine clasele destinate reprezentarii obiectelor care vor fi stocate in baza de date.
Pachetul repositores contine clasele care implementeaza operatiile pe baza de date.
Pachetul service contine clasele care implementeaza functionalitatiile logice ale aplicatiei.
Pachetul controller contine controllerul aplicatiei , cel care implementeaza metodele care vor fi apelate atunci cand frontend-ul va trimite o cerere.



Figură 5-3: Diagrama de pachete a aplicației de gestiune a utilizatorilor

5.2.2. Diagrame de clase

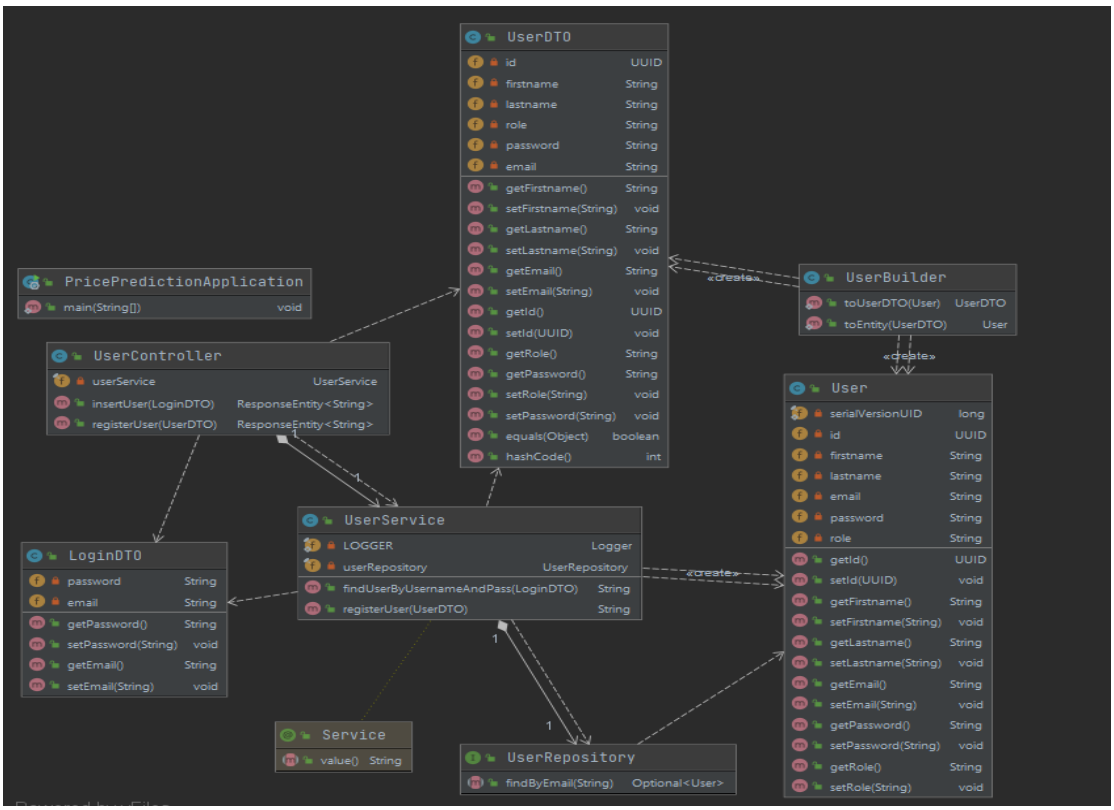


Figura 5-4: Diagrama de clase a aplicației pentru gestiunea utilizatorilor

5.3. Modulul pentru construirea modelului de predicție a prețului

Construirea modelului de predicție a fost realizată în Jupyter Notebook pentru o mai bună vizualizare a rezultatelor. Pentru fiecare secțiune de cod am scris câte o descriere sugestivă pentru a fi ușor de identificat scopul acesteia.

La început am importat toate librariile necesare implementării.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.svm import SVR
        from sklearn.metrics import mean_absolute_error, mean_squared_error
        from sklearn.model_selection import ShuffleSplit
        from sklearn.model_selection import cross_val_score
        from sklearn.linear_model import LinearRegression
        from sklearn.model_selection import train_test_split
        import datetime
        import os
        import json
```

Figura 5-5: Librăriile necesare implementării modelului de predicție

5.3.1. Metoda de citire a datelor

Aceasta metoda primeste ca si parametru un path catre un fisier csv de unde va citii datele si le va returna. În ultima linie am apelat această metodă.

```
def load_data(csv_path):
    return pd.read_csv(csv_path)

announces = load_data("house_offers.csv")
```

Figura 5-6. Metoda de citire a datelor

5.3.2. Metodele de preprocesare a datelor

Prima metodă trateaza duplicatele, coloanele irelevante și valorile null.

Prima linie de cod elimină rândurile duplicate din DataFrame-ul "announces" folosind metoda drop_duplicates.

Urmatoarele linii adaugă o nouă coloană, numită cum apare în partea dreaptă între ghilimele, la DataFrame. Valorile din această coloană sunt luate din coloana existentă cu același nume. Dacă există valori lipsă (NaN) în coloana existentă, acestea sunt înlocuite cu zero (0) folosind metoda fillna(0).

Apoi vom face o copie a datelor modificate până acum, dupa care eliminăm anumite coloane din DataFrame-ul "data_announces". Coloanele specificate în lista [...] sunt eliminate din DataFrame pentru a reduce dimensiunea și a exclude datele care nu sunt relevante pentru analiza ulterioară.

Ultima linie elimină rândurile care conțin cel puțin o valoare lipsă (NaN)pentru a ne asigura că nu mai avem deloc valori NaN.

Eliminarea duplicatelor, gestionarea valorilor null si eliminare coloanelor irelevante

```
announces.drop_duplicates
announces['garages_count'] = announces['garages_count'].fillna(0)
announces['bathrooms_count'] = announces['bathrooms_count'].fillna(0)
announces['kitchens_count'] = announces['kitchens_count'].fillna(0)
announces['parking_lots_count'] = announces['parking_lots_count'].fillna(0)
announces['balconies_count'] = announces['balconies_count'].fillna(0)

data_announces=announces
data_announces= data_announces.drop(columns=['id','location' ,'partitioning','built_surface', 'level', 'seller_typ
data_announces=data_announces.dropna()
```

Figura 5-7: Secțiunea de cod pentru eliminarea duplicatelor, coloanelor irelevante și a valorilor null.

Următoarea secțiune de cod adaugă două coloane noi la setul de date, price_per_unit care reprezintă prețul pe unitatea de suprafață, îl obținem împărțind prețul total la suprafața utilă. Urmatoarea coloana adaugată este building_years pe care o calculăm scăzând din anul curent anul în care a fost construit imobilul.

După calculul noilor caracteristici formăm două seturi de date , primul se opține prin stergerea coloanei construction_year folosind metoda drop(), iar cel de-al doilea se obține eliminând și coloana price pe lângă cea menționată anterior.

```
data_announces['price_per_unit'] = data_announces['price']/data_announces['useful_surface']
data_announces['building_years']= datetime.datetime.now().year - data_announces['construction_year']
data_with_full_price=data_announces.drop(columns=['construction_year' ], axis = 1)
data_announces= data_announces.drop(columns=['price','construction_year' ], axis = 1)
```

Figura 5-8: Secțiune de cod pentru adăugarea de noi caracteristici relevante predicției

Metoda de separarea a datelor dintr-un DataFrame în două seturi distincte, unul pentru antrenare și celălalt pentru testare.

X și y sunt convențional folosite pentru a reprezenta caracteristicile (variabilele independente) și variabila dependentă din setul de date. În cazul nostru, X reprezintă caracteristicile, iar y reprezintă variabila țintă.

test_size=0.2 specifică proporția datelor pe care dorim să le folosim pentru testare, în acest caz, 20% și 80% vor rămâne pentru antrenare. Funcția train_test_split din biblioteca sklearn efectuează efectiv împărțirea datelor în cele două seturi. Datele de antrenare sunt stocate în variabilele X_train și y_train, în timp ce datele de testare sunt stocate în variabilele X_test și y_test.

```
X = np.c_[np.ones(X.shape[0]), X]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
```

Figura 5-9: Secțiunea de cod pentru împarțirea datelor, în date de testare și antrenare

5.3.3. Codificarea locațiilor

Codul de mai jos are rolul de a efectua o codificare a locațiilor în funcție de prețul mediu pe metru pătrat în fiecare cartier.

Începem prin calcularea mediei prețului pe metru pătrat (Pret/mp) pe toate datele noastre și o stocăm în variabila "overall_mean_price". Pentru calculul mediei ne vom folosi de funcția mean()

Apoi, grupăm datele noastre în funcție de coloana "Cartier" și calculăm valoarea medie a prețului pe metru pătrat pentru fiecare cartier în parte. Rezultatele sunt stocate într-un nou DataFrame numit "grouped".

Calculăm media valorilor din grupul "grouped" și o stocăm în variabila "media_mediilor". Aceasta este valoarea medie a prețului pe metru pătrat în toate cartierele.

Pentru a codifica locațiile, calculăm raportul dintre prețul pe metru pătrat al fiecărui cartier și valoarea medie globală. Acest raport este stocat într-o nouă coloană numită "cod_locatie" în DataFrame-ul "grouped".

Pentru a introduce codificarea înapoi în DataFrame-ul original, folosim funcția merge() pentru a combina datele din "grouped" cu datele din "data_announces" pe baza coloanei "Cartier". Astfel, fiecare rând din "data_announces" va avea acum o valoare de codificare asociată cu cartierul din care face parte.

Sortăm datele din "data_cod_locatie" în funcție de codificare în ordine crescătoare.

Salvăm rezultatele codificării într-un fișier JSON numit "mapare_locatii.json" pentru a putea fi utilizate ulterior.

Pentru a vizualiza codificarea, creăm un grafic de bare orizontal care prezintă prețul mediu pe metru pătrat în funcție de fiecare cartier. Aceasta ne permite să observăm cum sunt distribuite valorile codificate în ordine crescătoare.

În final, afișăm DataFrame-ul "grouped" sortat în funcție de codificare pentru a vedea cum au fost codificate cartierele în funcție de prețul lor pe metru pătrat. Coloana "cod_locatie" este convertită în tipul de date float pentru a fi utilizată corespunzător.

```
overall_mean_price = data_announces['Pret/mp'].mean()
print("Media pretului/mp pe toate datele",overall_mean_price )

grouped = data_announces.groupby('Cartier')['Pret/mp'].mean().reset_index()
media_mediilor = grouped['Pret/mp'].mean()

print("Media pretului pe fiecare locatie",grouped)
grouped['cod_locatie'] = grouped['Pret/mp'] / media_mediilor

data_cod_locatie = data_announces.merge(grouped[['Cartier', 'cod_locatie']], on='Cartier', how='left')
data_cod_locatie = data_cod_locatie.sort_values('cod_locatie', ascending=True)
grouped[['Cartier', 'cod_locatie']].to_json('mapare_locatii.json', orient='records')

grouped_sorted = grouped.sort_values('Pret/mp')

plt.barh(grouped_sorted['Cartier'], grouped_sorted['Pret/mp'])
plt.xlabel('Prețul Mediu pe Metru Pătrat')
plt.ylabel('Cartier')
plt.title('Prețul Mediu pe Metru Pătrat în funcție de Cartier (Ordonate Crescător)')
plt.show()

print(grouped.sort_values('cod_locatie', ascending=True))
grouped['cod_locatie'] = grouped['cod_locatie'].astype(float)
```

Figura 5-10: Secțiunea pentru codificarea locațiilor

5.3.4. Metoda de antrenare a modelului

Această metodă conține antrenarea celor trei modele de predicție folosindu-se de trei algoritmi diferiți.

Variabila model_cleaned este o instanță a modelului LinearRegression din scikit-learn. Parametrul fit_intercept=True specifică faptul ca modelul trebuie sa estimeze interceptul în timpul antrenamentului.Variabila model_rf este o instanță a modelului RandomForestRegressor din scikit-learn. Parametrul n_estimators speficiă numărul de arbori de decizie din pădurea aleatoare și parametrul random_state setează valoarea seeed-ului aleator.

```
model_cleaned = LinearRegression(fit_intercept=True)

model_rf = RandomForestRegressor(n_estimators=100, random_state=42)

svm = SVR(kernel='rbf', C=1e3, gamma=0.00001)



model_cleaned.fit(X_train_cleaned, y_train_cleaned)
model_rf.fit(X_train_cleaned, y_train_cleaned)
svm.fit(X_train_cleaned, y_train_cleaned)
```

Figura 5-11. Codul pentru antrenarea celor trei modele de predicție

Această secțiune de cod se referă la utilizarea modelelor antrenate (Regresia Liniară, Random Forest și Regresia cu Vectori de Suport) pentru a face predicții pe datele de testare curățate.

- model_cleaned este modelul de regresie liniară antrenat pe datele de antrenare curățate.
- X_test_cleaned conține caracteristicile de intrare (variabile independente) ale datelor de testare curățate.
- model_cleaned.predict(X_test_cleaned) folosește modelul de regresie liniară pentru a face predicții pe baza datelor de testare curățate și le stochează în variabila y_pred_cleaned. Aceste predicții reprezintă prețurile estimate pe metru pătrat pentru datele de testare.

Urmatoarele doua linii reprezintă predicția pentru celelelate doua modele.

```
y_pred_cleaned= model_cleaned.predict(X_test_cleaned)
y_pred_rf= model_rf.predict(X_test_cleaned)
y_pred_svm= svm.predict(X_test_cleaned)
```

Figura 5-12: Secțiunea de cod care efectuează predicția folosind cele trei modele

5.3.5. Metoda pentru calcularea perfomantei modelului

Codul din Figura 5-13 se referă la evaluarea performanței celor trei modele pe setul de date de testare curățate și calcularea scorurilor pentru fiecare model

diff_cleaned, diff_cleaned_rf, diff_cleaned_svm - Aceste linii de cod calculează diferența absolută între valorile reale (y_test) și valorile prezise de fiecare model (y_pred, y_pred_rf și y_pred_svm) pentru setul de date de testare curățate.

Următoarele trei linii calculează acuratețea fiecărui model bazat pe diferențele calaculate anterior. Se compară diferența absolută cu un prag specific, care este calculat ca fiind 15% din valorile prezise ale acelui model.

Acuratețea este calculată ca fracțiunea de exemple de testare pentru care diferența absolută este mai mică sau egală cu pragul, împărțită la numărul total de exemple de testare.

Apoi se va afișa scorul fiecărui model. Scorul reprezintă procentul de predicții corecte făcute de fiecare model pe datele de testare și este o măsură a calității modelelor. Cu cât scorul este mai mare, cu atât modelul este mai precis în estimările sale.

```
diff_cleaned = np.abs(y_test - y_pred)
diff_cleaned_rf = np.abs(y_test - y_pred_rf)
diff_cleaned_svm = np.abs(y_test - y_pred_svm)

accuracy = np.mean(diff_cleaned <= (y_pred*0.15))
accuracy_rf = np.mean(diff_cleaned_rf <= (y_pred_rf*0.15))
accuracy_svm = np.mean(diff_cleaned_svm <= (y_pred_svm*0.15))

print("Scorul pe setul de date de testare curățat:", accuracy)
print("Scorul pe setul de date de testare curățat rf", accuracy_rf)
print("Scorul pe setul de date de testare curățat svr", accuracy_svm)
```

Figura 5-13: Secțiunea de cod care calculează acuratețea celor trei modele de predicție

5.3.6. Metoda pentru detectia si eliminarea anomaliilor

Main întâi antrenăm modelul folosin functia predict().

diff = np.abs(y_test - y_pred) - Aici, se calculează diferența absolută între valorile reale "y_test" și valorile prezise "y_pred". Acest lucru ne oferă o măsură a discrepanței între predicție și realitate.

Se creează un nou DataFrame numit "diff_df" care conține trei coloane: "Y_test" cu valorile reale, "Y_pred" cu valorile prezise și Diferenta cu diferența absolută dintre ele. Acest DataFrame va fi folosit pentru analiza ulterioară.

Se stabilește un prag (threshold) pentru a identifica anomalii. În acest caz, pragul este setat la 100, ceea ce înseamnă că orice diferență absolută mai mare de 100 va fi considerată o anomalie.

Se creează un nou DataFrame numit "anomalies" care conține doar acele rânduri din "diff_df" pentru care diferența absolută este mai mare decât pragul definit anterior. Aceste rânduri reprezintă anomalii.

Se calculează media tuturor diferențelor absolute din "diff_df", oferind o măsură a diferenței medii între predicții și valorile reale.

exageret_anomalies = anomalies[anomalies['Diferenta'] > 2 * mean_diff]- Aceasta linie de cod creează un nou DataFrame numit "exageret_anomalies" care conține doar acele anomalii pentru care diferența absolută este mai mare de două ori media diferențelor. Acestea sunt considerate anomalii exagerate.

Se obțin indexurile acestor anomalii exagerate sub formă de listă pentru a identifica rândurile corespunzătoare în setul de date original.

La final se creează un nou DataFrame numit "anomaly_table" care conține doar rândurile din setul de date original "data_announces3" care corespund indexurilor anomalilor exagerate. Acesta conține informații detaliate despre aceste anomalii.

```
y_pred = model.predict(X_test)
diff = np.abs(y_test - y_pred)
diff_df = pd.DataFrame({'Y_test': y_test, 'Y_pred': y_pred, 'Diferenta': diff})
print(diff_df)


threshold = 100
anomalies = diff_df[diff_df['Diferenta'] > threshold]
print("Anomaliile sunt:")
print(anomalies)
mean_diff = np.mean(diff_df['Diferenta'])
print(f"Media diferențelor este {mean_diff:.2f}")
print("Anomaliile sunt:")
exageret_anomalies=anomalies[anomalies['Diferenta'] > 2*mean_diff]
print(exageret_anomalies)
#
anomaly_indexes = exageret_anomalies.index.tolist()
anomaly_table = data_announces3.loc[anomaly_indexes]
print(anomaly_table)
```

Figura 5-14: Secțiunea de cod pentru determinarea anomaliilor din setul de date

Dupa detectarea anomaliilor vom elimina acei indecși din seturile de date de antrenare și testare folosindu-ne de expresiile de mai jos.

np.arange(len(X_train)): Aceasta creează un vector de indecși de la 0 la lungimea setului de date de antrenare "X_train". Acest vector conține toate indecșii de la 0 la len(X_train) - 1.

np.isin(): Aceasta este o funcție NumPy care compară fiecare element din primul argument (în acest caz, vectorul de indecși creat mai sus) cu elementele din al doilea argument (în acest caz, "anomaly_indexes"). Rezultatul este un vector de valori booleane care indică dacă fiecare index din "X_train" este sau nu în lista de indecși a anomaliilor.

"~": Acest operator inversează valorile booleane din vectorul rezultat de la np.isin(), astfel încât indecșii care corespund anomaliilor să devină False și indecșii care nu corespund anomaliilor să devină True.

```
X_train_cleaned = X_train[~np.isin(np.arange(len(X_train)), anomaly_indexes)]
y_train_cleaned = y_train[~np.isin(np.arange(len(y_train)), anomaly_indexes)]

X_test_cleaned = X_test[~np.isin(np.arange(len(X_test)), anomaly_indexes)]
y_test_cleaned = y_test[~np.isin(np.arange(len(y_test)), anomaly_indexes)]
```

Figura 5-15: Secțiunea de cod destinată eliminării anomaliilor din seturile de antrenare și testare.

5.3.7. Clasa pentru generarea graficelor

Deoarece am generat destul de multe grafice, mai jos am explicat modul de implementare a unui grafic care compară valorile reale ale prețului pe metru pătrat (Y_test) cu valorile prezise ale prețului pe metru pătrat (Y_pred) pentru un set de date.

Creăm un DataFrame numit "results_df_price" care conține două coloane: "Y_test" care reprezintă valorile reale ale prețului pe metru pătrat și "Y_pred" care reprezintă valorile prezise ale prețului pe metru pătrat.

Sortăm DataFrame-ul "results_df_price" în funcție de coloana "Y_test" în ordine crescătoare, astfel încât valorile să fie ordonate în funcție de prețul real al metrului pătrat.

Resetăm indexul DataFrame-ului pentru a avea un nou index numeric care să reflecte ordinea sortată a valorilor.

Începem să creăm graficul folosind Matplotlib. Specificăm dimensiunile figurii folosind "plt.figure(figsize=(10, 5))".

Folosim funcția "plt.plot" pentru a trasa două linii pe grafic: una pentru valorile reale ("Valori reale" în roșu) și alta pentru valorile prezise ("Valori prezise" în verde, cu opacitate redusă).

Adăugăm o linie orizontală la nivelul mediei valorilor reale, evidențiată în gri cu un stil de linie punctată ("plt.axhline(...)"). Aceasta ne ajută să vedem cum se compară valorile prezise cu valoarea medie a celor reale.

Etichetăm axele X și Y pentru a indica ce reprezintă fiecare și adăugăm o legendă pentru a diferenția valorile reale de cele prezise și activăm grila pentru a face graficul mai ușor de citit.

Specificăm directorul în care dorim să salvăm graficul și numele fișierului ("nume_fisier") în care dorim să salvăm imaginea. Calculăm calea absolută a fișierului folosind "os.path.abspath(os.path.join(director, nume_fisier))" și salvăm imaginea în calea specificată cu "plt.savefig(ruta_absoluta)".

La final afișăm graficul cu "plt.show()".

```
import matplotlib.pyplot as plt

results_df_price = pd.DataFrame({'Y_test': y_test, 'Y_pred': y_pred})

results_df_price.sort_values(by=['Y_test'], inplace=True)

results_df_price = results_df_price.reset_index(drop=True)

print("Price (in Euro)")

plt.figure(figsize=(10, 5))
plt.plot(results_df_price['Y_test'], label='Valori reale', color='red')
plt.plot(results_df_price['Y_pred'], label='Valori prezise', alpha=0.5, color='green')
plt.axhline(y=results_df_price['Y_test'].mean(), color='gray', linestyle='--', label='Valoare medie')
plt.xlabel('Exemple ordonate cresator dupa pret/mp')
plt.ylabel('Preț/mp (in Euro)')
plt.legend()
plt.grid(True)
director = 'C:\\Users\\Larisa\\Desktop\\licenta\\price-prediction\\src\\assets'
nume_fisier = 'grafic_valoriPreziseDupaValoriReale.png'
ruta_absoluta = os.path.abspath(os.path.join(director, nume_fisier))
plt.savefig(ruta_absoluta)
plt.show()
```

Figura 5-16: Secțiunea de cod pentru reprezentarea grafică a valorilor prezise in raport cu cele reale

5.4. Modulul de colectare a datelor de pe internet

Functia de colectare a datelor de pe Internet a fost proiectata in asa fel incat sa se poata colecta atat anunturi pentru Bucuresti , cat si anunturi pentru Cluj.

Functia primeste ca parametrul orasul pentru care se doreste colectarea. Acest parametru este transmis prin ruta din frontend cu numele de city si de aceea se apeleaza functia de get a argumentelor pentru a lua valoarea acestuia. In functie de oras vom initializa variabilele care au valori diferite pentru cele doua orase, acestea sunt: existing_data – datele deja existente in excel, output_filename – numele fisierului in care se vor adauga datele colectate, url – url-ul care se acceseaza pentru colectarea anunturilor si zona- denumirea tagului care indica locatia din care este anuntul.

Dupa aceste atribuiri, vom extrage cea mai recenta data de publicare a unui anunt care se afla deja in setul de date, utilizand functia max().

In variabila PATH de declara calea catre fisierul executabil al driverului Chrome utilizat pentru web scraping. Apoi in variabila driver se va creea o instanta a driverului Chrome utilizand calea specificata in variabila PATH.

Prin intermediul comenzii driver.get(url) se va accesa pagina aferenta link-ului trimis ca parametru.

Pentru a evita anumite restrictii aplicate de site vom folosi variabila count pentru a contoriza cate anunturi au fost colectate.

In urma analizei codului sursa al site-ului cu anunturi am observat ca avem o lista initiala cu toate anunturile si fiecare anunt se regaseste in elemente care au clasa card, prin urmare extragem toate aceste elemenete cu functia find_elements si le stocam in variabila cards.

Informațiile extrase din anunțuri sunt stocate într-un dicționar denumit anunt.

Se verifică data anunțului extrasa și se compara cu data cea mai recentă din datele existente. Dacă data anunțului este mai recentă, acesta este adăugat la lista anunturi.După ce toate anunțurile au fost procesate, datele noi sunt stocate într-un nou DataFrame (df2).

Noile date sunt concatenate cu datele existente pentru a forma un DataFrame final (df_final). Datele finale sunt salvate într-un fișier Excel corespunzător orașului (output_filename) pentru a actualiza baza de date.La final, driverul WebDriver este închis pentru a elibera resursele.

```
for li in soup.find_all('li'):
    for strong in li.find_all('strong'):
        column_name = strong.text.strip()
        column_value = ''
        if strong.next_sibling is not None:
            column_value = strong.next_sibling.strip()
        numeric_value = re.search(r'\d+[\.,]?\d*', column_value)
        if numeric_value:
            column_value = numeric_value.group(0).replace(',', '.')
            print(column_value)
        anunt[column_name] = column_value
```

Figura 5-17: Codul pentru extragerea caracteristicilor unui anunț de pe site

5.5. Aplicația Web

5.5.1. Comunicarea între Backend si Frontend

Un aspect fundamental în dezvoltarea aplicațiilor web îl reprezintă comunicarea dintre Frontend și Backend. În cadrul acestui proiect, comunicare se realizează prin intermediul protocolului de transfer de date HTTP. Pentru schimbul de informații între client și server, Frontend-ul poate face cereri HTTP, Backend-ul primește cererile și le procesează iar după procesare transmite un raspuns HTTP către frontend care conține informațiile solicitate sau un status care indica dacă cererea s-a executat cu succes sau a intervenit o eroare.

În fișierul **httpConfig.js** se configurează și se creează instante Axios pentru efectuarea cererilor HTTP.

```
3   const javaEndpoint = "http://localhost:8085";
4   const pythonEndpoint = "http://localhost:5000";
5
6   export const javaHttp = axios.create({
7     baseURL: javaEndpoint,
8     headers: {
9       Accept: "application/json",
10      "Content-Type": "application/json",
11    },
12  });
13
14  export const pythonHttp = axios.create({
15    baseURL: pythonEndpoint,
16    headers: {
17      Accept: "application/json",
18      "Content-Type": "application/json",
19    },
20  });
```

Figura 5-18:

Mai întai se definesc adresele de baza pentru servere ținând cont de portul pe care acestea rulează:

- '**javaEndpoint**' reprezintă adresa de baza pentru serverul Java care rulează pe portul 8085 (http://localhost:8085)
- '**pythonEndpoint**' reprezintă adresa de bază pentru serverul Python care rulează pe portul 5000 (http://localhost:5000)

Apoi se creează doua instanțe separate de obiecte Axios 'javaHttp' și 'pythonHttp' pentru fiecare server utilizând metoda 'axios.create()', aceste instanțe sunt configurate pentru a efectua cereri HTTP către două servere distincte, metoda create() are doi parametrii: baseUrl reprezintă URL-ul de bază adică adresele definite anterior iar al doilea parametru, headers e folosit pentru a specifica formatul de date care se acceptă și tipul de conținut trimis către server, în cazul nostru configurăm cererile ca să trimită și să accepte date cu format JSON.

Istanțele de obiecte Axios sunt exportate și folosite ulterior în fișierul httpUtils la construirea unor funcții generice care au ca parametrii un URL și alți parametrii opționali și care vor fi folosite în intreaga aplicație pentru realizarea cererilor HTTP de tip GET, POST, PUT, PATCH si DELETE către cele două servere.

Aceste funcții sunt denumite sugestiv pentru a putea face diferența dintre serverul către care se va face cererea și scopul lor este de a abstractiza detalii precum gestionarea anteturilor și a tipului de conținut.

5.5.2. Componenta de meniu

Componenta **Menu** reprezintă bara de meniu a aplicației, aceasta asigura vizibilitatea elementelor de meniu in functie de diferite conditii precum tipul de utilizator si starea de logare a acestuia.

La inceputul componentei se extrag variabilele de stare de tip bool care ne indica daca utilizatorul este logat (isLoggedIn) respectiv daca acesta este administratorul aplicatiei (isAdmin). Pentru ca aceste variabile trebuie sa aiba o valoare persistenta in toate componentele aplicatiei, acestea s-au definit in contextul componentei principale (AppContext) , iar pentru a le putea accesa si modifia in componenta Menu se foloseste hook-ul useContext pentru a accesa variabilele din contextul principal.

Functia onClickLogout din aceasta componenta este apelata cand utilizatorul se delogheaza, respectiv apasa pe optiunea Logout din meniu. Aceasta functie seteaza variabila isLoggedIn si isAdmin la false, efectuand astfel delogarea.

Partea de return a componentei cuprinde ceea ce va fi randat in aplicatie, mai exact in tag-ul header se afla toate componentele meniului, care sunt afisate in functie de starea de autentificare si tipul utilizatorului folosindu-se randearea conditionata.

Daca utilizatorul normal sau de tip administrator nu este logat, in meniu vor aparea doar optiunile Login, Register, Anunturi, Evaluarea proprietate si Logo-ul aplicatiei, care reprezinta link-ul catre pagina principala a aplicatiei.

Daca utilizatorul este logat si nu este de tip admin atunci pentru acesta vor aparea urmatoarele optiuni: Logo-ul, Anunturi, Statistici, Evaluare proprietate, si Logout. Iar daca utilizatorul este de tip admin si este logat, pe langa cele mentionate anterior ii va aparea optiunea Admin.

Fiecare optiune din meniu este reprezentata prin intermediul componentelor NavItem din biblioteca reactstrap , respectiv Link din biblioteca react-router-dom, in interiorul acestor tag-uri se trece continutul care dorim sa fie afisat ca optiune in meniu iar in tag-ul de Link la atributul to se va pune URL-ul catre care se va naviga in comentul apasarii optiunii.

5.5.3. Componenta de login

O componenta importanta a aplicatiei o reprezinta componenta de Login prin intermediul careia utilizatorul in urma completarii datelor necesare se poate autentifica in contul sau.

Starile interne principale ale componentei sunt:

- email – stare folosita pentru stocarea valorii emailului introdus de utilizator in form
- password – stare folosita pentru stocarea parolei introduse de utilizator in form
- wrongDataMessage – este o variabila de stare care contine mesajul de eroare care va fi afisat daca utilizatorul introduce credentiale gresite.
- navigate – este o variabila care ne ajuta sa redirectionam utilizatorul catre o alta pagina in urma unei actiuni

Functiile onEmailChanged si onPasswordChanged primesc ca si parametru valoarea introdusa de utilizator in campul email, respectiv password si sunt apelate de fiecare data cand valoarea acestora se schimba, in interiorul acestor functii se seteaza variabilele de stare corespunzatoare.

Functia onLoginClicked se apeleaza in momentul in care utilizatorula apasa butonul de Login. Aceasta functie este una asincrona datorita cererii de tip POST trimisa catre backend la ruta /login, ca si parametru al cererii vom trimite un obiect format din credentialele introduse de utilizator. Daca rezultatul returnat este diferit de null atunci verificam care este rolul utilizatorului, in cazul in care acesta este de tip admin vom seta variabila de context isAdmin cu valoarea true, iar apoi indiferent de tipul utilizatorului vom seta variabila isLoggedIn la tru si vom naviga catre pagina de home a aplicatiei. Daca in baza de date nu este gasit un utilizator cu credentialele respective, cererea va returna un status de eroare si vom afisa un mesaj sugestiv.

Componenta vizuala este reprezentata de un form cu doua input-uri, un buton de Login care va fi activ doar in momentul in care sunt introduse valori in ambele campuri, mesajul de eroare in cazul unei autentificari esuate, si un link catre pagina de inregistrare in cazul in care utilizatorul nu are un cont.

5.5.4. Componenta de Inregistrare cont

Componenta Register permite utilizatorului sa se inregistreze prin completarea unui formular. Prin intermediul acestei componente se pot inregistra doar utilizatori de tip clienti, utilizatorul de tip administrator fiind unul singur adaugat manual in baza de date.

Variabilele de stare ale acestei componente sunt:

- formData – aceasta variabila stocheaza datele introduse de utilizator in formular si are structura unui obiect cu campurile corespunzatoare input-urilor din formular(firstname, lastname, email, password, confirmPassword).
- errorMessage – e o variabila care stocheaza mesajele de eroare legate de toate campurile formularului, pentru campul email avem mesajul de eroare care va aparea atunci cand utilizatorul introduce un format gresit de email, pentru campurile password si confirmPassword avem mesajul de eroare in cazul in care parola nu este suficient de complexa si nu respecta formatul cerut, si un al mesaj de eroare este destinat introducerii gresite a parolei confirmate
- passwordConfirmed, validEmail si registrationSuccess – reprezinta variabile de stare de tip bool care vor avea valoarea true in cazul in care avem parola initiala se potriveste cu cea confirmata, avem un email valid, respectiv inregistrarea s-a efectuat cu succes, iar in caz contrar acestea vor avea valoarea false.

Functia onInputChange este apelata de fiecare data cand valoarea unui camp din formular se schimba si are rolul de a actualiza starea variabilei formData si de a valida valorile introduse.

Functia handleFormSubmit este apelata atunci cand utilizatorul apasa pe butonul Register si are scopul de a trimite catre backend datele introduse de acesta in formular prin intermediul unei cereri de tip POST catre ruta /register. Daca toate campurile au fost valide si cererea s-a realizat cu succes atunci variabila registrationSucces va fi setata la true.

Functia validateInput valideaza valorile introduse de utilizator in formular si seteaza mesajul de eroare specific fiecare validari esuate, si tot in functie de rezultatul validarii seteaza si valoarea starilor passwordConfirmed si validEmail.

Componenta returneaza un formular cu urmatoarele campuri : nume, prenume, email, parola si confimare parola si un buton de Register care va fi inactiv cat timp valorile din campuri nu sunt valide. Mesajele de eroare se vor afisa sub campul carora le corespund , iar daca inregistrarea s-a efectuat cu succes se va afisa un mesaj sugestiv.

5.5.5. Pagina de vizualizare a anunturilor

Pagina destinata anunturilor ne permite sa le vizualizam caracteristicile, sa navigam catre site-ul de pe care provine anuntul respectuv si sa aplicam diferite filtre pentru a vizualiza doar anunturile care ne intereseaza. Implementarea si reprezentarea vizuala a acestor functionalitati se regasesc in componenta Anunturi.

Starile interne ale acestei componente sunt urmatoarele:

- excelData – aceasta variabila e folosita pentru a stoca lista de anunturi afisata in interfata
- filters – este o variabila de tip obiect care va contine valorile filtrelor dorite, acest obiect contine urmatoarele etichete: nrCamere, compartiment, cartier, mai exact atributele dupa care se poate efectua filtrarea.

- filteredData – aceasta variabila stocheaza anunturile care satisfac filtrele selectate

Dupa declararea variabilelor de stare, am implementat extragerea anunturilor din fisierul de pe disc in interiorul unui useEffect, deoarece pentru a avea versiunea finala a datelor stocate ne dorim ca la fiecare rerandare a paginii de anunturi sa se actualizeze datele in cazul in care acest lucru s-a intamplat in fisierul excel. Pentru extragerea propriu-zisa se realizeaza o cerere de tip GET catre serverul de python cu ruta /read-excel. Daca aceasta cerere s-a executat cu succes rezultatele sunt stocate in starea excelData si se seteaza si starea variabilei filteredData tot cu acest rezultat, iar daca cererea esueaza se va afisa eroarea in consola.

Urmatoarea functie, handleFilterChange se va apela de fiecare data cand se modifica o valoare indiferent de campul in care aceasta se modifica, in interiorul functiei setam valoarea variabilei de stare cu noile valori.

Ultimul useEffect contine filtrarea anunturilor si se va apela de fiecare data cand se schimba filtrele, adica variabila filters, respectiv setul de anunturi, variabila excelData.

Aceasta componenta va returna conditionat in functie de numarul de anunturi, 3 campuri cu etichete sugestive destinate valorilor filtrelor si un tabel cu anunturile care va avea toate coloanele care se regasesc si in excelul de pe disc, ultima coloana din acest tabel va fi reprezentata de un buton care ne va deschide o noua pagina catre link-ul anuntului de unde a fost extras pentru mai multe detalii despre acesta.

```
7   const Anunturi = () => {
8     const [excelData, setExcelData] = useState([]);
9     const [filters, setFilters] = useState({
10      nrCamere: "",
11      compartiment: "",
12      cartier: "",
13    });
14    const [filteredData, setFilteredData] = useState([]);
15
16    useEffect(() => {
17      const fetchData = async () => {
18        axios
19          .get("/read-excel", {
20            baseURL: "http://localhost:5000",
21          })
22          .then((response) => {
23            const data = JSON.parse(response.data);
24            setExcelData(data);
25            setFilteredData(data);
26            console.log(data);
27          })
28          .catch((error) => {
29            console.error(error);
30          });
31      };
32
33      fetchData();
34    }, []);
```

Figura 5-19: Variabilele de stare alea componentei Anunțuri și useEffectul pentru citirea datelor

5.5.6. Componenta administratorului

Ca si administrator putem face urmatoarele lucruri: sa actualizam setul de date destinat anunturilor imobiliare din Cluj cat si din Bucuresti si sa reantrenam modelul de predictie a pretului manual. Functionalitatile mentionate anterior se regasesc implementate in componenta AdminPage.

Componenta AdminPage este o functie care returneaza interfata vizuala a paginii de administrare sub format JSX, pentru asigurarea functionalitatilor dorite am avut nevoie de trei variabile de stare declarate folosin useState:

- isLoading – este o variabila de tip bool si reprezinta starea de incarcare care va fi false cat timp nu se colecteaza noi anunturi, respective true in timpul colectarii, aceasta variabila ne ajuta sa reprezentam vizual perioada in care se colecteaza anunturile.
- nrAnunturiColectate – este o variabila de tip int si reprezinta numarul de inregistrari colectate in urma actiunii de actualizare a setului de date
- city – este o variabila de tip string , initializata cu valoarea cluj, care ne ajuta sa identificam pentru ce oras se doreste actualizarea setului de date.

Functia handleExtraction este apela atunci cand utilizatorul apasa pe butonul Actualalizeaza pentru a colecta anunturi din orasul selectat. In interiorul functie se va construi URL-ul pentru cerere in functie de valoarea variabile city. Daca setul de date a fost actualizat cu succes atunci se va afisa numarul de anunturi colectate primit ca raspuns de la server, iar daca nu , se va afisa un mesaj de eroare in cosola.

Functia onCityChange este apelata de fiecare data cand orasul selectat se schimba si in interiorul ei se realizeaza setarea variabile de stare city la valoarea selectata in interfata.

Aceasta componenta returneaza un div in care se gaseste un selectBox pentru selectarea orasului cu optiunile Bucuresti sau Cluj si un buton pentru actualizarea setului de date. Cat timp are loc colectarea datelor se va afisa textul :Loading... , iar la finalul colectarii se va afisa cate anunturi au fost colectate.

La final aceasta componenta este exportata pentru a putea fi folosita si in alte parti , mai exact la crearea rutelor aplicatiei.

```
function AdminPage() {
  const [isLoading, setIsLoading] = useState(false);
  const [nrAnunturiColectate, setNrAnunturiColectate] = useState(0);
  const [city, setCity] = useState("cluj");

  const handleExtraction = () => {
    setIsLoading(true);
    setNrAnunturiColectate(0);

    const url =
      city === "cluj" ? "/extract?city=cluj" : "/extract?city=bucuresti";
    getRequestPy(url)
      .then((response) => {
        const { message, num_data } = response.data;
        setIsLoading(false);
        setNrAnunturiColectate(num_data);
        console.log(message);
      })
      .catch((error) => {
        setIsLoading(false);
        console.error(error);
      });
  };

  const onCityChange = (event) => {
    setCity(event.target.value);
  };

```

Figura 5-20: Funcția care inițializează extracția datelor în componenta administratorului

5.5.7. Comoponenta de statistici

Componenta Statistici returneaza un meniu cu toate reprezentarile grafice generate, si in functie de optiuniea pe care o vom selecta ni se va afisa o poza cu graficul respectiv si o scurta descriere.

Mai intai am importat toate imaginile cu grafice din folderul assets, dupa care am declarat o constanta globala de tip obiect care are cate o eticheta sugestiva pentru fiecare grafic disponibil, ca si valoare pentru aceste etichete avem tot un obiect cu atributele: image care reprezinta imaginea propriu-zisa cu graficul si description, descrierea care va fi afisata impreuna cu acesta.

Ca si variabile de stare interne aceasta componenta are doar starea selectedOption folosita pentru a identifica optiunea selectata din meniu. Functia handleOptionSelected se apeleaza de fiecare data cand se selecteaza o alta optiune si are rolul de a seta variabila seletedOption.

Functia renderGraph se ocupa cu randarea oricarui grafic in interfata. Aceasta verifica daca avem vreo optiune selectata, daca nu avem se va afisa un mesaj sugestiv, iar daca avem vom extrage imaginea si descriere in doua constante locale din obiectul global care le contine si vom returna un div compus din o imagine si un paragraf cu descrierere.

5.5.8. Componenta pentru estimarea pretului

Componenta PriceEstimate consta intr-un formular in care utilizatorul introduce caracteristicile proprietatii sale, iar apoi folosindu-ne de modelele de predictie create ii vom prezice un pret.

In partea de sus a componentei functionale se declara variabilele de stare necesare estimării pretului, acestea sunt : (price, priceRF, priceSVM) cate o variabila pentru fiecare pret prezis de fiecare model diferit, variabila care stocheaza cartierele din JSON (neighborhoods), și pentru datele introduse de utilizator (formData).

Functia handleChange este o funcție pentru a actualiza starea formData atunci când utilizatorul completează câmpurile de intrare. Ea este apelată atunci când valorile din câmpurile de intrare sunt modificate.

Functia handleSubmit este apelată atunci când utilizatorul trimite formularul. Ea face o cerere POST către serverul backend (la adresa http://localhost:5000) cu datele introduse de utilizator în formData.Când serverul răspunde cu rezultatele estimărilor de preț, valorile sunt actualizate în starea componentei.

Efectul useEffect este folosit pentru a încărca datele din mapare_locatii.json în starea neighborhoods la încărcarea inițială a paginii.

Componenta este compusa dintr-un formular care contine pentru fiecare caracteristică (număr de camere, suprafață utilă etc.) câmpuri de intrare pentru ca utilizatorii să introducă valorile.

Câmpul "Cartier" este reprezentat de un meniu select, unde utilizatorii pot alege unul din cartierele încărcate din neighborhoods.

Pentru ca utilizatorul sa trimita formularul se foloseste un buton Estimare Pret, cand acesta este apasat se afișează rezultatele estimărilor de preț pentru fiecare model, precum și o medie și preț total estimat.

```
31  const handleSubmit = (e) => {
32    e.preventDefault();
33
34    axios
35      .post("/", formData, {
36        baseURL: "http://localhost:5000",
37      })
38      .then((response) => {
39        setPrice(response.data.price);
40        setPriceRF(response.data.price_rf);
41        setPriceSVM(response.data.price_svm);
42      })
43      .catch((error) => {
44        console.error(error);
45      });
46  };
47
48  useEffect(() => {
49    setNeighborhoods(locatii);
50  }, []);
51
```

Figura 5-21: Funcția pentru estimarea prețului

5.5.9. Componenta principală

La inceputul componentei principale sunt importate fișierele necesare pentru funcționarea componentei App. Menu este o componentă pentru meniu, Route și

Routes sunt componente oferite de React Router pentru gestionarea rutelor și navigarea între pagini, iar celelalte importuri sunt pentru diverse funcționalități și hook-uri.

Apoi este creat un context numit AppContext. Acest context va fi folosit pentru a partaja informații între diferite componente din aplicație.

Componenta App reprezinta nucleul aplicatiei Aici sunt definite stările isLoggedIn (pentru verificarea autentificării utilizatorului) și isAdmin (pentru a verifica dacă utilizatorul este administrator). De asemenea, se inițializează variabilele de stare userId și setUserId utilizând hook-ul usePersistentState, care pare să fie definit în altă parte a codului.

Componenta utilizează contextul AppContext.Provider pentru a furniza aceste informații altor componente din aplicație. În interiorul acestei componente, se randează meniul (Menu) și rutele aplicației (appRoutes) folosind componente Routes și Route. Acestea din urmă sunt gestionate de către React Router pentru a direcționa utilizatorii către paginile corespunzătoare.

5.5.10. Alte componente importante

In cadrul pachetului constants am creat fisierul **PagesUrl** care contine toate rutele disponibile in aplicatie, deoarece acestea pot aparea in mai multe componente ale aplicatiei declarandu-le intr-un singur loc centralizat vom asigura coerenta si usurinta in gestionarea acestora . Rutele sunt definite in obiectul PAGES_URL sub forma de perechi cheie-valoare , cheia reprezinta un identificator unic pentru o anumita ruta, in timp ce valoarea reprezinta ruta efectiva care se va afisa in browser. De exemplu, identificatorul Admin este folosit pentru ruta catre pagina administratorului si este setata la /admin.

Componenta **usePersistentState** din pachetul hooks reprezinta un hook personalizat care are ca si scop pastrarea unei stari intre sesiuni si reincarcari ale aplicatiei, mai exact a fost implementata pentru pastrarea state-urilor legate de statusul de login al utilizatorilor, astfel oferindu-le acestora o experienta coerenta fara delogari involuntare.

Aceasta functie are doi parametrii, storageKey care reprezinta cheia la care se gaseste valoarea starii dorite in localStorage, respectiv initialState care reprezinta valoarea initiala a variabilei de stare.

Mai intai vom declara o variabila de stare interna persistentState folosind useState() si ii vom oferii ca si valoare initiala parametrul primit de functie.

Iar apoi prin intermediul unui useEffect care se va executa o singura data la incarcarea initiala a componentei vom verifica valorile stocate in localStorage la cheia storageKey. Daca exista o valoarea salvata in localStorage, aceasta este preluata si se verifica daca are valoarea true sau false , daca da, atunci se va transforma intr-un boolean, altfel, se pastreaza ca si sir de caractere. Daca exista o valoare recuperata din localStorage atunci starea interna va fi setata cu acea valoare.

In ultima parte a codului se defineste o functie setState care va inlocui functia obisnuita de setare a straii , aceasta va actualiza starea interna persistentState dar va si salva valoarea acesteia in localStorage la cheia respectiva.

Acest hook va returna un array format din doua elemente: persistentState- starea curenta persistenta si setState- functia pentru actualizarea starii.

5.6. Aplicatia de gestiune a utilizatoriilor

5.6.1. Descrierea claselor si metodelor

5.6.1.1. Clasa User

Clasa User- reprezinta entitatea de baza si incapsuleaza diverse atribute care definesc un utilizator. Aceste atribute sunt : un identificator unic de tip UUID(id), prenumele (firstname), numele de familie (lastname), adresa de e-mail (email), parola (pasword) si rolul (role) de tip string. Dupa declarearea atributelor, clasa User prezintă un constructor care inițializează obiectul cu valorile furnizate, iar ca si metode avem doar getter si setter pentru fiecare atribut.

5.6.1.2. Interfata UserRepository

Interfata UserRepository extinde JpaRepository oferit de frameworkul Spring pentru a facilita interactiunea dintre aplicatie si baza de date. Aceasta mosteneste operatiile predefinite pentru inregistrarile de tip user, cum ar fi: salvarea, preluarea, actualizarea și ștergerea datelor utilizatorului.

5.6.1.3. Clasa UserBuilder

Clasa UserBuilder nu are atribute servind ca o clasa de utilitate. Clasa are două metode esențiale: toUserDTO() și toEntity(). Metoda toUserDTO() convertește un obiect User într-un obiect UserDTO corespunzător, facilitând transferul de date și straturile de abstractizare în cadrul aplicației. În schimb, metoda toEntity() permite conversia unui obiect UserDTO înapoi într-un obiect User.

5.6.1.4. Clasa UserDTO

Clasa UserDTO extinde RepresentatiomModel si contine toate atributele clasei user, dar si metodele de get si set, in plus avem un constructor fara parametrul id, aceasta clasa suprascrie metodele hashCode si equals, metoda equals verifica daca un obect de tip object este egal cu un obiect de tip user, iar metoda hash returneaza codul hash al obiectului.

5.6.1.5. Clasa LoginDTO

Clasa LoginDTO este folosita pentru autentificarea si include doar atributele necesare procesului de conectare, si anume password si email. In aceasta clasa se regaseste un constructor si metodele getter si setter pentru fiecare atribut.

5.6.1.6. Clasa UserController

Clasa UserController are un atribut final de tip UserService care faciliteaza interactiunea intre controller si serviciile corecpunzatoaare. Clasa contine doua metode , cea de login si register, in interiorul metodelor se apeleaza metodele din service corespunzatoare pentru autentificare, respectiv inregistrare si se retuneaza un ResponseEntity adecvat. Ambele metode au adnotari de @PostMapping cu numele rutei permitand aplicatiei sa gestioneze cererile POST primite si sa execute logica asociata.

5.6.1.7. Clasa UserService

Clasa UserService se foloseste de un obiect de tip UserRepository pentru a implementa logica din spatele metodelor de login si register.

5.6.2. Persistenta datelor

În cadrul aplicației, se utilizează baza de date MySQL pentru a asigura persistența datelor. Acest aspect este deosebit de important, deoarece permite stocarea și accesarea informațiilor relevante pentru utilizatori într-un mod sigur și fiabil.

Prin intermediul framework-ului Spring și a bibliotecii Hibernate, aplicația este configurată pentru a comunica cu baza de date MySQL. Acest lucru implică definirea unei conexiuni la baza de date și a unui set de proprietăți, precum utilizatorul și parola necesare pentru autentificarea la serverul MySQL. Aceste proprietati se regasesc in fisierul application.properties si trebuie schimbate in functie de credentialele proprii.

```
##############################################
### DATABASE CONNECTIVITY CONFIGURATIONS ###
##############################################
database.ip = ${MYSQL_IP:localhost}
database.port = ${MYSQL_PORT:3306}
database.user = ${MYSQL_USER:root}
database.password = ${MYSQL_PASSWORD:root}
database.name = ${MYSQL_DBNAME:users-management}
```

Figura 5-22: Datele de conectare la baza de date

Datorita faptului ca aplicatia se ocupa doar cu gestiunea utilizatorilor, in baza de date vom avea un singur tabel, acesta este tabelul users care are urmatoarele coloane :id, firstname, lastname, email, password si role, coloana id reprezentand cheia primară.

Capitolul 6. Testare și validare

În acest capitol, vom aprofunda toate metodele de testare și validare care au fost aplicate atât pe parcurs cât și la finalul implementării, pentru a asigura cea mai bună acuratețe și eficacitate a aplicației noastre.

6.1. Testarea și validarea modelelor de predicție

Testarea modelului de predicție a prețului antrenat pe setul de date din București s-a efectuat atât pe parcursul dezvoltării modelului cât și pe rezultatul final.

La inceputul procesului de dezvoltare, am optat să lucrez cu seturi de date de dimensiuni reduse, create intern, pentru a ne a asigura că modelul nostru nu este afectat de eventuale anomalii sau date nereprezentative. Această etapă inițială ne-a permis să identificăm și să corectăm orice probleme potențiale în stadiile incipiente ale dezvoltării, asigurând astfel o temelie solidă pentru modelul nostru.

Pentru testarea modelului, am aplicat principiul fundamental al testării pe date necunoscute modelului, adică pe date cu care acesta nu s-a antrenat anterior. Pentru a realiza acest lucru, am împărțit setul de date inițial în două părți distincte: una destinată antrenării și cealaltă pentru testare. Partea de testare conținea date complet noi și neutilizate în procesul de antrenare, asigurând astfel că modelul nostru este capabil să generalizeze corect și să facă predicții precise pe date reale, necunoscute în prealabil.

În procesul de dezvoltare a modelului de predicție, am luat în considerare variații în ceea ce privește caracteristicile utilizate. Inițial, am antrenat modelul fără a lua în considerare locația ca o caracteristică. Cu toate acestea, ulterior, am decis să includ locația ca o variabilă importantă, deoarece am observat că poate avea un impact semnificativ asupra prețului proprietății.

Pentru a valida și evalua corectitudinea rezultatelor obținute, am implementat o metodă eficientă de vizualizare. Inițial, am generat un tabel care a afișat valorile reale ale proprietăților imobiliare alături de valorile prezise de către model. Acest tabel ne-a oferit o perspectivă directă și comparativă asupra performanței modelului, permițându-ne să observăm diferențele dintre estimările acestuia și realitatea din teren. Această abordare ne-a ajutat să identificăm eventuale discrepanțe.

În Figura 6-1 se poate observa tabelul generat în urma efectuării predicției de catre modelul antrenat folosind regresia liniară, inaintea eliminării anomaliilor. Pe coloana din stanga se afla valorile reale ale prețului pe unitatea de suprata, iar în partea dreaptă se află valorile prezise.

	Y_test	Y_pred
2827	1244.186047	1319.245234
309	1078.431373	1108.684211
3070	4162.698413	4217.873159
587	2090.909091	1920.392834
5768	2812.500000	2600.694363
...	...	...
1707	2493.750000	2180.310124
3143	2362.509434	2569.734251
683	1320.000000	1317.308947
1886	1469.016393	1733.923040
770	1795.774648	1721.624110

Figura 6-1: Tabel cu valorile reale si cele prezise în urma predicției cu modelul de regresie liniară

Pentru a valida mai precis rezultatele predicției am creat o metodă de calcul a acurateții modelului. Metoda de calcul a acurateții se bazează pe evaluarea preciziei modelului în predicția prețurilor imobiliare și presupune compararea valorilor prezise cu cele reale. Această metoda constă în calculul diferenței absolute dintre prețurile reale ale proprietăților imobiliare și prețurile prezise de către model. Aceste diferențe absolute reprezintă discrepanța dintre predicțiile modelului și datele reale. În esență, se evaluează dacă diferențele absolute dintre valorile prezise și cele reale sunt mai mici sau egale cu 15% din valorile prezise. Astfel, fiecare diferență este comparată cu 15% din valoarea prezisă corespunzătoare. Dacă o diferență se încadrează în această limită de toleranță, este considerată acceptabilă, și acest lucru contribuie la creșterea scorului de acuratețe. Acest prag reprezintă o diferență acceptabilă între prețurile prezise și cele reale, pe care considerăm că modelul ar trebui să o respecte. Scorul final este obținut calculând media tuturor evaluărilor făcute pentru diferențele dintre valorile prezise și cele reale. Cu cât un scor este mai mare, cu atât modelul este mai precis în prezicerea valorilor target. Primul scor a fost de 86% precizie in predicția prețului, pentru modelul dezvoltat folosind regresia liniară.

După obținerea scorului pe setul de date de testare, am efectuat o analiză detaliată asupra valorilor prezise de către model și a diferențelor dintre acestea și valorile reale cunoscute. În urma acestei analize, am observat că în setul de date pot exista valori aberante care au avut un impact negativ asupra preciziei modelului nostru. Am identificat aceste predicții care prezentau o discrepanță semnificativă față de valorile reale și am luat măsuri pentru a le elimina din setul de date. Această acțiune a avut scopul de a îmbunătăți performanța generală a modelului nostru, asigurându-ne că acesta se bazează pe date fiabile și coerente, fără influențe negative generate de valori aberante.

Următoarea etapă a procesului de testare a constat în reantrenarea modelului după eliminarea anomaliilor identificate anterior. După aceasta, am refăcut predicțiile folosind setul de date curățat, fără acele valori aberante. Rezultatele obținute au arătat o îmbunătățire semnificativă a performanței modelului de predicție, noul scor fiind de 89% precizie in predicția prețului.

O altă metodă eficientă pentru validarea rezultatelor obținute în cadrul modelului de predicție a fost vizualizarea grafică a prețului prezis în comparație cu cel

real. Această abordare ne oferă o imagine clară și intuitivă a performanței modelului, permițându-ne să evaluăm cu ușurință discrepanțele dintre valorile prezise și cele reale. Prin plasarea acestor valori pe un grafic, am putut observa modelele și tendințele din setul de date, precum și orice deviații semnificative. În Figura se poate observa reprezentarea grafică a rezultatelor modelului de predicție dezvoltat folosind regresia liniară pe un set de date fară anomalii.

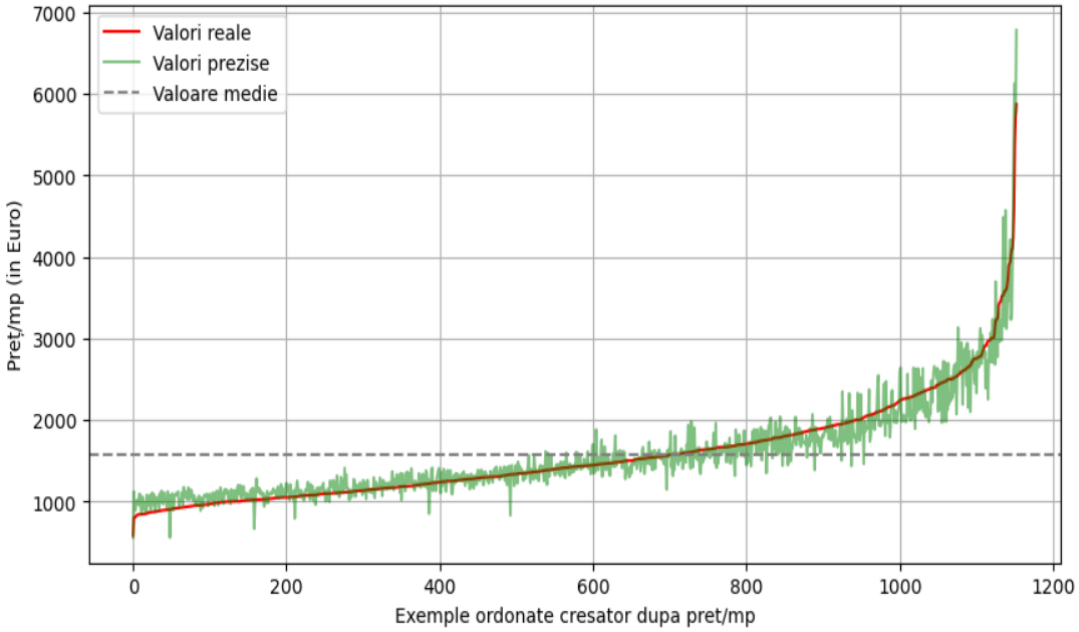


Figura 6-2: Reprezentarea grafică a valorilor prezise ordonate crescător dupa valorile reale pentru modelul de predicție dezvoltat folosind regresia liniară pe un set de date fară anomalii

Pentru a extinde testarea modelului, am realizat și predicția prețului întreg al proprietăților, pentru a analiza comportamentul modelului. Aceasta modalitate de testare a fost una eficientă deoare rezultatele modelului s-au dovedit a fi mai bune în cazul predicției prețului total. Acest lucru se poate observa in Figura care constă in reprezentarea grafică a predicției prețului întreg folosind modelul dezvoltat cu regresia liniară pe setul de date fără anomalii.

Din analiza celor două grafice, putem observa că precizia modelului este mai mare în cazul predicției prețului total comparativ cu predicția prețului pe unitate. Cu toate acestea, trebuie să luăm în considerare contextul și scopul utilizării acestor predicții. În general, o abordare mai logică ar fi să efectuăm predicția prețului pe unitate, deoarece acesta oferă o informație mai detaliată și specifică pentru potențialii cumpărători sau investitori în imobiliare.

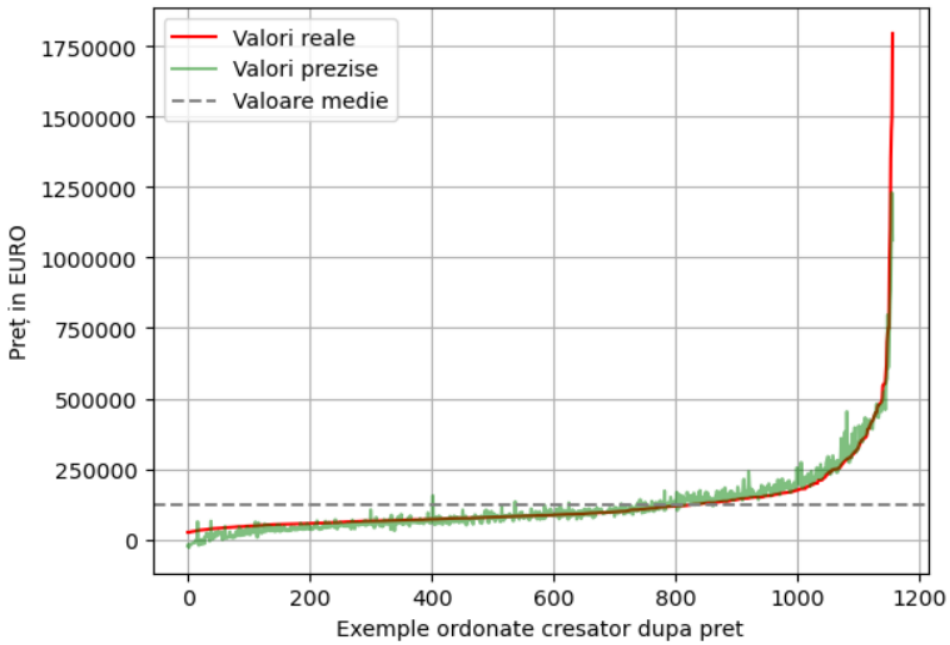


Figura 6-3: Reprezentarea grafică a valorilor prezise ordonate crescător dupa valorile reale obținute în urma predicției prețului toatal pentru modelul cu regresia liniară pe un set de date fară anomalii

Pentru validarea vizuală a predicției prețului total am realizat și o altă modalitate de reprezentare grafică a rezultatelor predicției. În figura de mai jos se punctele reprezintă perechi dintre valoarea prezisa a pretului si cea reală, iar cu cât aceste puncte sunt mai apropiate de linia de referință cu atât predicția modelului este mai precisă.

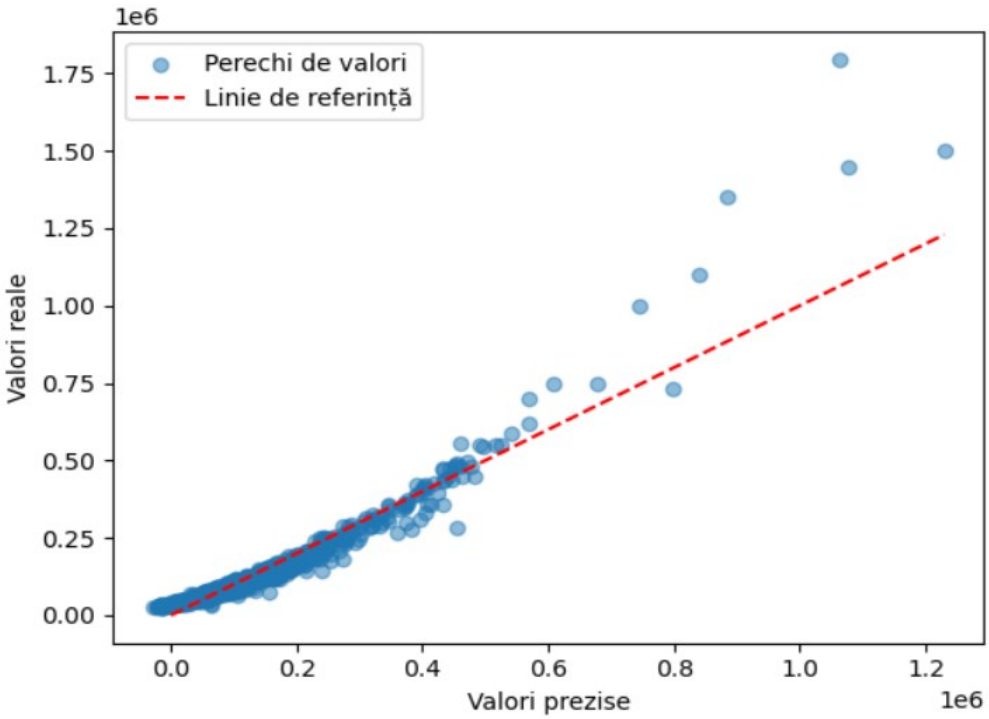


Figura 6-4: Reprezentarea grafică sub formă de perechi a valorilor prezise si a celor reale obținute în urma predicției prețului total folosind modelul cu regresia liniară

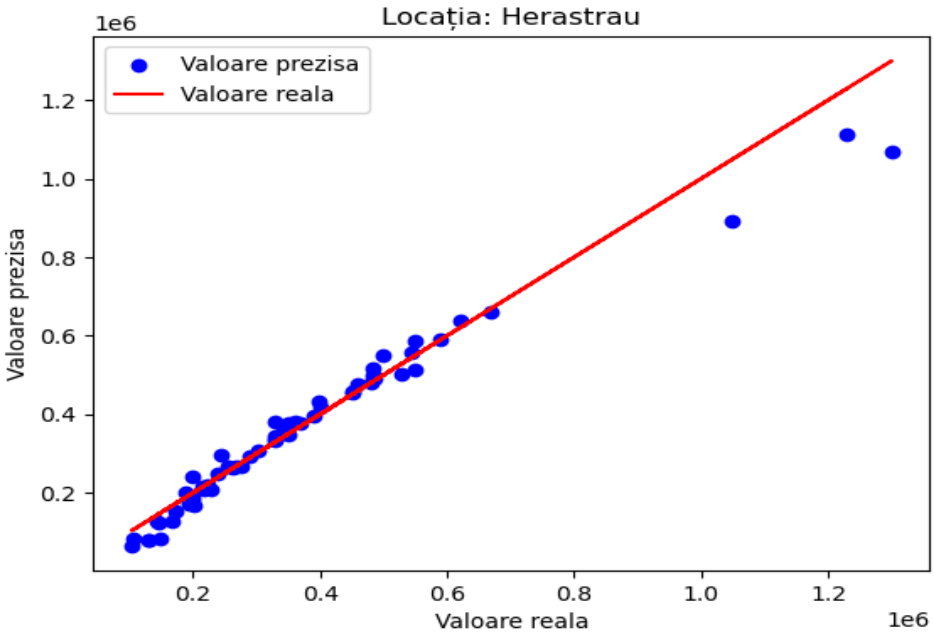


Figura 6-5: Reprezentarea grafică sub formă de perechi a valorilor prezise si a celor reale obținute în urma predicției prețului total pe date cu locația Herastrau folosind modelul cu regresia liniară

Pe măsură ce am obținut un model de predicție cu o acuratețe semnificativă, am decis să explorăm și să dezvoltăm încă două modele alternative, bazate pe algoritmi diferiți, cum ar fi Random Forest si regresie cu vectori de suport. Acest demers a avut drept scop evaluarea și compararea performanței multiplelor soluții pentru a ne asigura că alegem cea mai eficientă și robustă opțiune. Testarea acestor modele alternative a fost esențială pentru a verifica dacă există posibilitatea de îmbunătățire a predicțiilor noastre prin folosirea altor algoritmi.

În Figura 6-6 se poate observa reprezentarea grafică a prețurilor prezise în raport cu cele reale pentru modelul dezvoltat folosind Random Forest. Putem observa o diferență semnificativă comparativ cu modelul dezvoltat folosind regresia liniară.

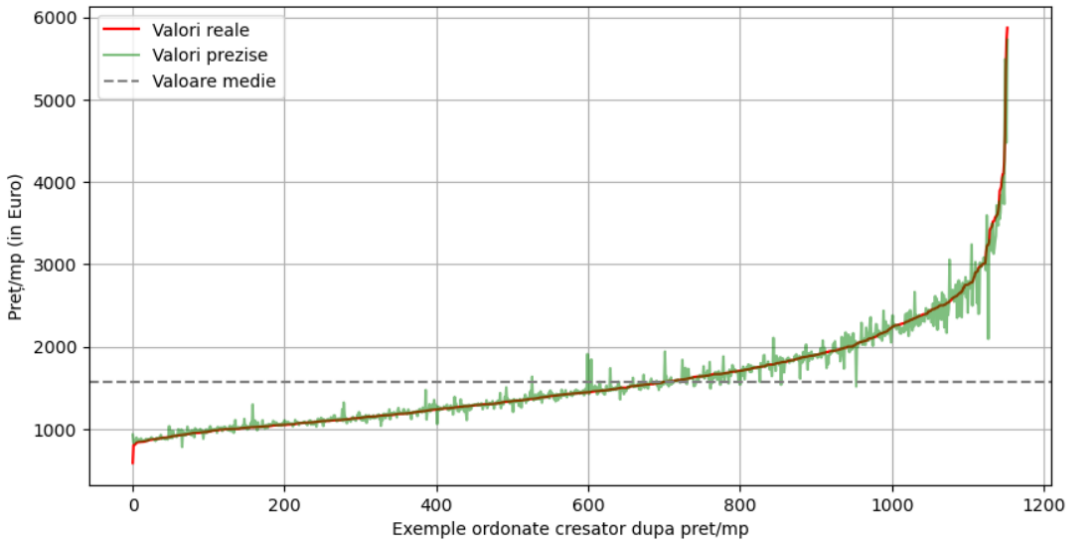


Figura 6-6: Reprezentarea grafică a valorilor prezise ordonate crescător dupa valorile reale obținute în urma predicției prețului toatal pentru modelul cu Random Forest pe un set de date fară anomalii

În Figura 6-7 se poate observa reprezentarea grafică a prețului pe unitatea de suprafață prezis în raport cu cel real pentru modelul dezvoltat folosind regresia cu vectori de suport. De aceasta dată predicția pare sa fie mai mai puțin precisă față de celelalte modele prezentate anterior.

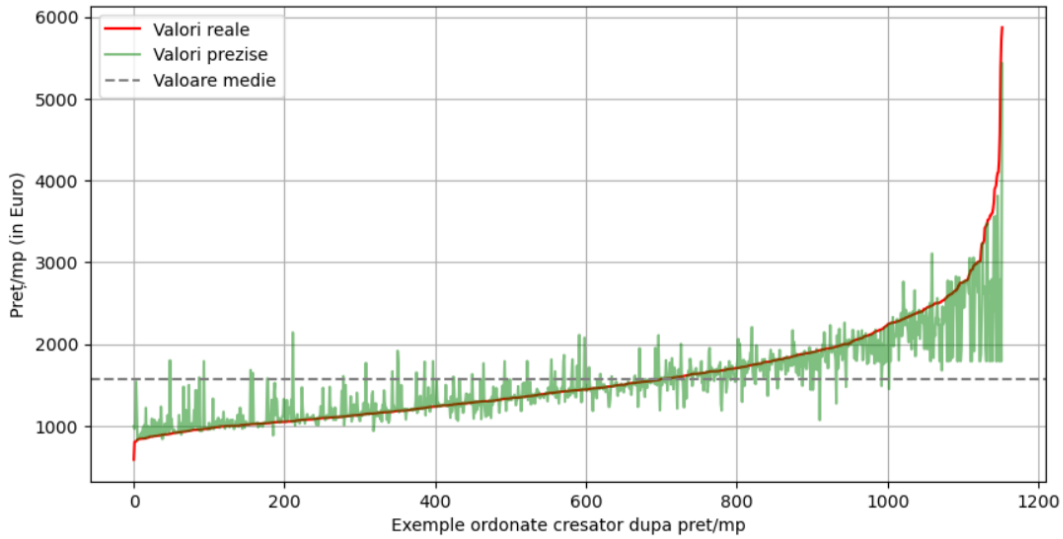


Figura 6-7: Reprezentarea grafică a valorilor prezise ordonate crescător dupa valorile reale obținute în urma predicției prețului toatal pentru modelul cu Support Vector Regression pe un set de date fară anomalii

Dupa compararea grafică a celor trei metode de efectuare a predicției, am calculat acuratețea fiecărui modelu cu metoda amintită mai sus.

În procesul nostru de dezvoltare și validare a modelului de predicție, am testat mai mulți algoritmi și tehnici pentru a asigura obținerea celor mai precise rezultate posibile. Am observat că, în urma testelor noastre, algoritmul Random Forest a obținut o acuratețe de aproximativ 98%, în timp ce regresia liniară a înregistrat o acuratețe de 89%. De asemenea, am testat și un model bazat pe Support Vector Regression (SVR), care a avut o acuratețe de aproximativ 83%. Diferențele între aceste rezultate au devenit vizibile încă din stadiul de analiză grafică a datelor. Cu toate acestea, prin experimentare și îmbunătățirea continuă a modelelor noastre, am reușit să ajungem la un rezultat final mai bun, cu o acuratețe de 98%, ceea ce sugerează că Random Forest a fost cea mai potrivită metodă pentru problema noastră specifică.

```
Scorul pe setul de date de testare curățat: 0.8941890719861232
Scorul pe setul de date de testare curățat rf 0.9800520381613183
Scorul pe setul de date de testare curățat svr 0.8317432784041631
```

Figura 6-8: Scorurile obținute de cele trei modele de predicție

Pentru testarea și validarea modului de predicție pentru setul de date cu anunțuri din Cluj-Napoca colectate am aplicat aceeași pași ca și pentru setul de date din București și am observat o diferență destul de semnificativă a performanței modelului, obținând rezultate mai slabe cel mai probabil datorită dimensiunii mult mai mici a setului de date și diverselor valori aberante.

Scorurile dupa eliminarea anomaliilor sunt urmatoarele: 64% acuratețe pentru modelul dezvoltat folosind regresia liniară, 69% acuratețe pentru modelul dezvoltat folosind Random Forest si 36% acuratețe pentru modelul construit folosind Support Vector Regression. Putem observa ca cel mai bun scor îl are și în acest caz modelul dezvoltat cu Random Forest, iar datorită volumului mic de date de antrenare Support Vector Regression nu se dovedește a fi un algoritm eficient.

Pentru a analiza faptul că volumul de date a fost un factor negativ in cazul acestui model am analizat rezultatele pe fiecare cartier în parte și am ajung la concluzia că cele mai bune rezultate de predicție s-au înregistrat pentru cartierul Florești, cel cu cele mai multe anunțuri in setul de date , acest lucru se poate observa și în graficul de mai jos.

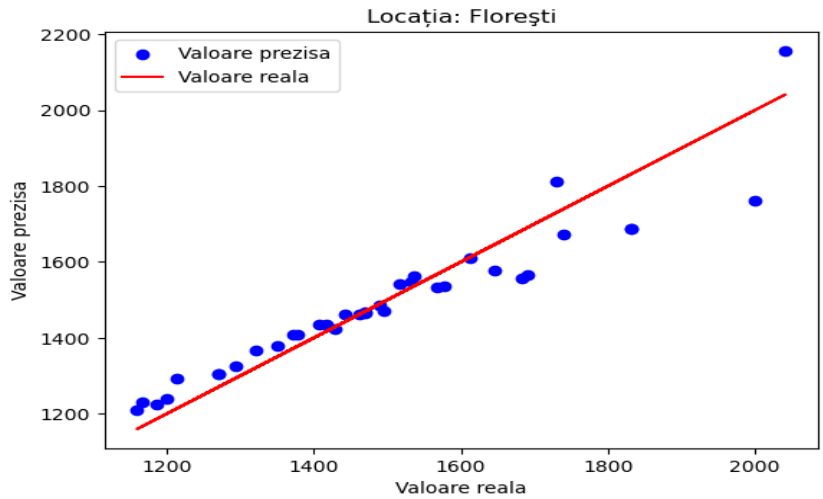


Figura 6-9: Reprezentarea sub formă de perechi de valori a valorilor reale și a celor prezise de către modelul antrenat pe setul de date din Cluj-Napoca

6.2. Testarea și validarea modulului care colectează anunțuri

Pentru testarea functionalitatii de colectare de anunturi de pe Internet vom merge pe pagina administratorului si vom apasa butonul ‘actualizeaza’ pentru a incepe colectarea. Apoi, daca procesul s-a terminat cu succes, vom vizualiza inregistrarile din fisierul output.csv. Pentru validarea functionarii corecte a acestui modul analizam valorile stocate in fiecare coloana, ne asiguram daca o anumita coloana are un tip de valoare valid , de exemplu daca pe coloana ‘Nr. Camere’ este scris un numar si nu valoarile altei caracteristici cum ar fi cartierul din care este anuntul. De asemenea, chiar daca valorile sunt de tipul corect, pentru validare am luat in considerare si sa nu fie prezente valori aberante, deoarece dupa prima testare am observat ca pentru coloana pretului aveam o valoare foarte mica, ceea ce ridica anumite semne de intrebare, in final realizand faptul ca un punct pus pe website-ul respectiv pentru o vizualizare mai buna a pretului era transpus in virgula la salvarea datelor. Ultimul pas de validare consta in navigarea catre link-ul salvat pe coloana ‘Link anunt’ pentru a verifica daca valorile caracteristicilor sunt exact cele salvate in excel.

În Figura 6-10 se pot observa în partea de sus datele colectate și salvate în excel, iar în partea de jos datele corespunzătoare anunțului colectat de pe site-ul web.

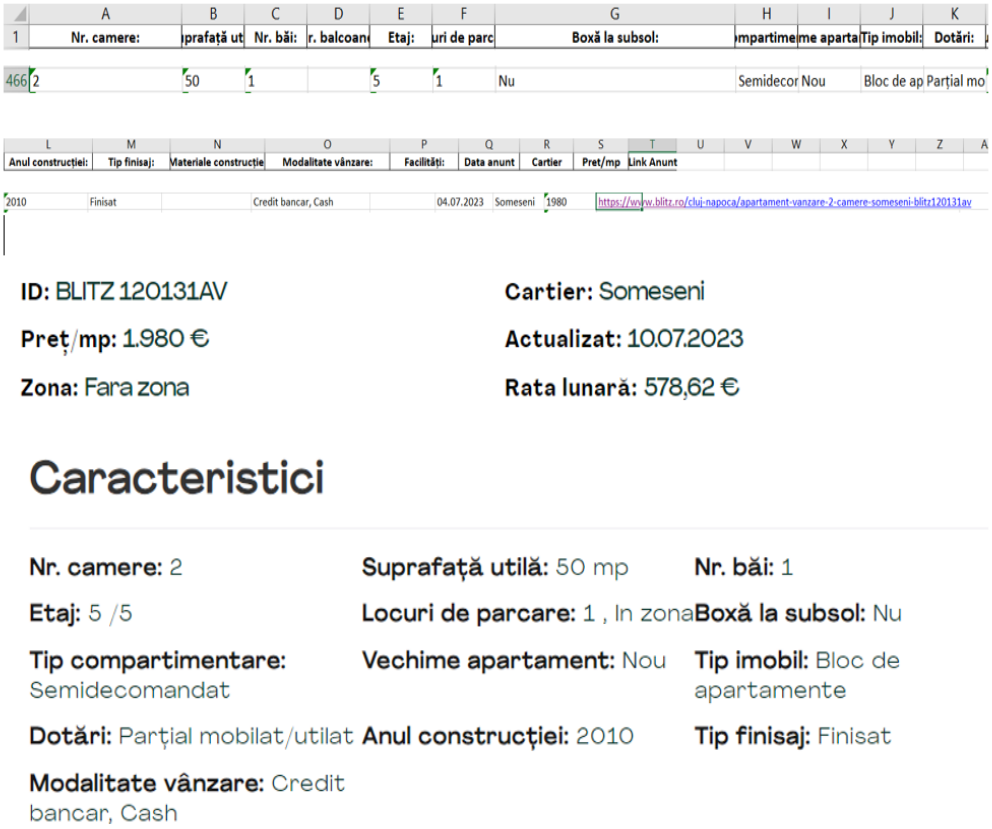


Figura 6-10: Validarea modulului de colectare a anunțurilor

6.3. Testarea și validarea aplicației de gestiune a utilizatorilor

Pentru a testa functionalitățile acestei aplicații vom utiliza atât Postman cât si aplicația Web pentru a ne asigura și ca aceste endpoint-uri sunt integrate corespunzător în partea de frontend.

6.3.1. Testarea functionalității de înregistrare a unui cont

Vom face o cerere HTTP de tip POST pentru testarea funcționalității de inregistrare în cazul in care contul este înregistrat cu succes. În Figura 6-11 se poate observa că in urma efectuării cererii am primit ca raspuns mesajul care indica efectuarea cu succes a cererii.

Figura 6-11: Validarea metodei de înregistrare a unui cont

Pentru a valida înregistrarea contului, vom analiza datele din baza de date și după cum se poate observa în Figura 6-12 contul utilizatorului a fost adăugat cu succes.

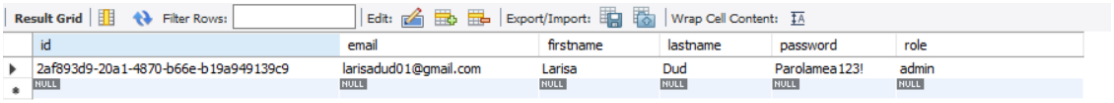


Figura 6-12: Baza de date cu contul înregistrat anterior

De asemenea, vom testa și cazul în care un utilizator încearca să se înregistreze cu un cont deja existent, pentru aceasta cerere vom introduce datele utilizate anterior.

În Figura 6-13 de mai jos se poate observa că am primit mesajul la care ne așteptam.

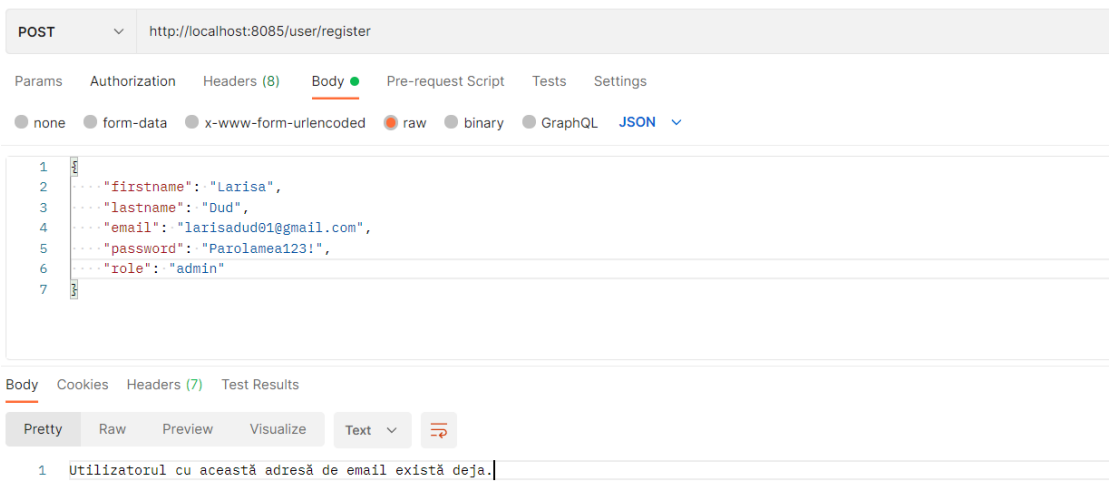


Figura 6-13: Validarea metodei de înregistrare cont în cazul în care acesta există deja

Pentru siguranță validăm si datele din baza de date și după cum se poate observa în figura de mai jos, nu a fost adaugat un nou cont cu aceeasi adresă. Deci, putem spune ca metoda de inregistrare a unui cont este o metodă validă.

id	email	firstname	lastname	password	role
2af893d9-20a1-4870-b66e-b19a949139c9	larisadud01@gmail.com	Larisa	Dud	Parolamea123!	admin
e9578c60-f6e4-427a-a4d2-c56dc8f26a29	mariapop@yahoo.com	Maria	Pop	Parolamea123!	client
NULL	NULL	NULL	NULL	NULL	NULL

Figura 6-14: Validarea datelor din baza de date

6.3.2. Testarea funcționalității de autentificare

Am făcut o cerere HTTP de tip POST pentru testarea funcționalității de autentificare în cazul in care contul este înregistrat, credențialele introduse sunt corecte și contul este de admin. În Figura 6-15 se poate observa că in urma efectuării cererii autentificarea s-a efectuat cu succes și am primit rolul de admin ca răspuns.

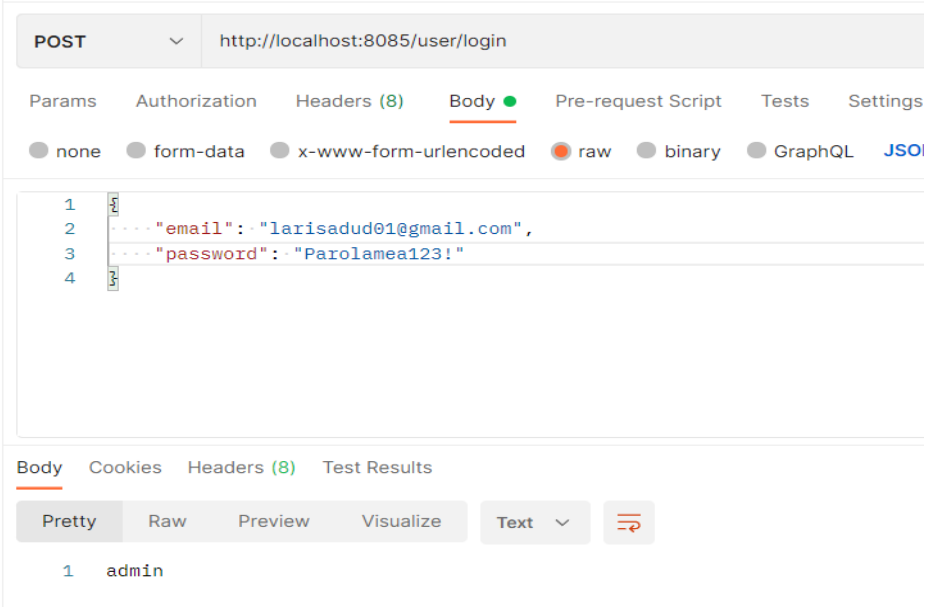


Figura 6-15: Validarea metodei de login în cazul autentificării administratorului

Apoi, am făcut o cerere ca cea de mai sus dar pentru un utilizator de tip client. In Figura 6-16 se observa ca și aceasta cerere s-a efectuat cu succes.

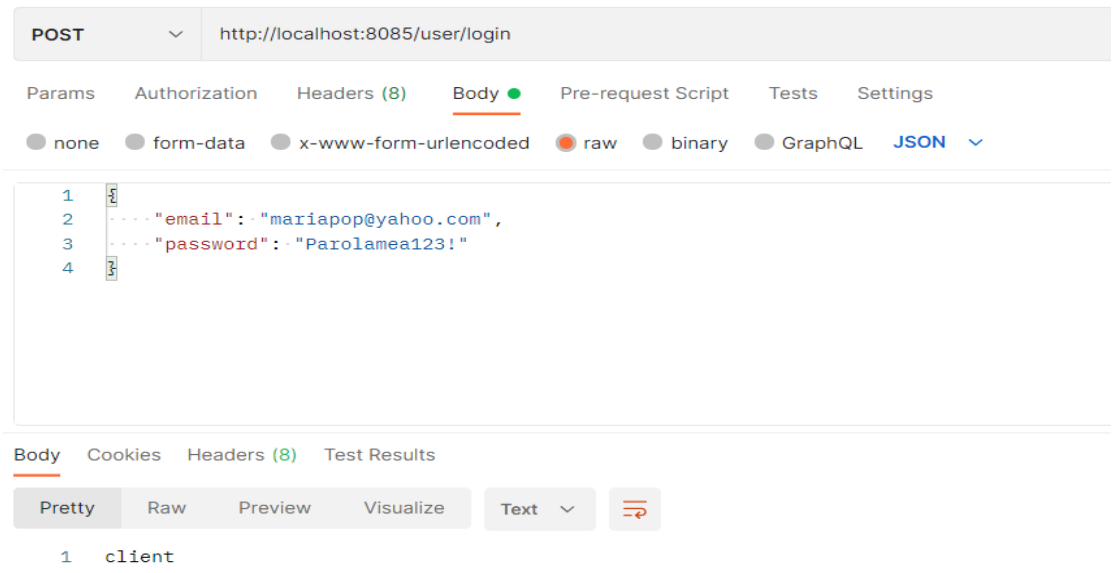


Figura 6-16: Validarea metodei de login în cazul autentificării unui client

În cele din urmă, vom testa și cazul negativ, în care utilizatorul își greșeste credențialele. În Figura 6-17 se poate observa ca am introdus o parola greșită și am primit mesajul de Credențiale gresite, deci putem spune ca și metoda de autentificare este o metodă validă.

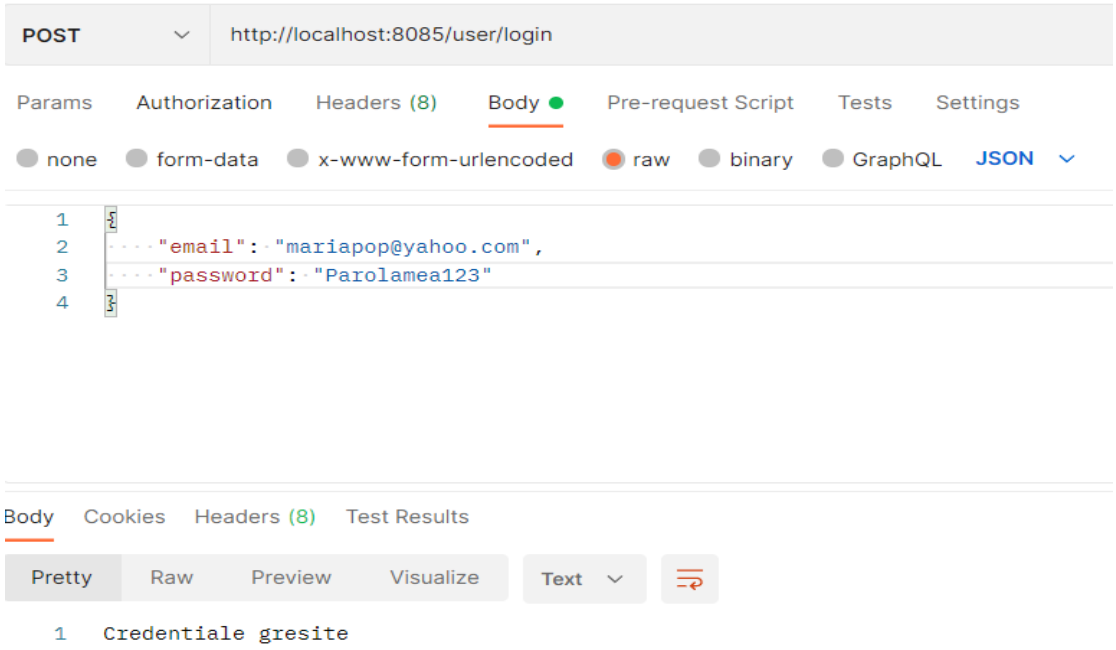


Figura 6-17: Validarea metodei de login în cazul introducerii unor credențiale greșite

Capitolul 7. Manual de instalare si utilizare

În cadrul capitolului curent se vor detalia resursele necesare (atât software, cât și hardware) pentru instalarea și rularea sistemului, dar se va include și o descriere pas cu pas a procesului de instalare.

Totodată, în cadrul acestui capitol, se va descrie modul de utilizare al aplicației din punct de vedere al utilizatorului, fără a se menționa aspecte tehnice interne, ci doar cunoștințe de bază deținute de orice utilizator neexperimentat.

7.1. Resurse necesare

Pentru a putea utiliza aceasta aplicatie vom avea nevoie de urmatoarele resurse software:

Resursa software de bază este ca pe calculator să fie prezent un sistem de operare Windows 7 sau 10 m pentru a putea instala restul resurselor software necesare

In plus, vom avea nevoie să instalăm următoarele aplicații: IntelliJ, Visual studio code, Python, Node.js, MySQL Workbench. Instalarea acestora se va face dupa manualul de instalare dedicat de pe site-ul fiecarei aplicatii in parte.

Pentru instalarea dependențelor interne putem utiliza diverse comenzi rulate în terminal cum ar fi npm install pentru aplicația web, pip install pentru aplicația python iar cele din aplicația inteligenței se vor instala automat.

Această aplicație va putea fi transferată de pe un calculator pe altul prin intermediul unui stick sau prin intermediul diferitelor platforme precum gitHub de unde se va putea descărca codul.

După dezarhivarea codului vom regăsi 3 foldere, cel cu numele userManagement va conține soluția scrisă în IntelliJ, folderul cu numele prediction a conține aplicația web și predicție va conține aplicația parton care va fi deschisă folosind deja studio cod.

Pentru a rula întreaga soluție vom rula fiecare aplicație în parte.

Primul pas este acela de a deschide aplicația MySQL Workbranch si de a crea o noua baza de date cu numele users-management

Pentru a rula aplicația de gestiune a utilizatorilor vom deschide mediul de programare IntelliJ, vom apăsa pe opțiunea File apoi pe Open iar în caseta deschisă vom selecta ruta către folderul PricePrediction și vom apăsa Open. După ce proiectul este deschis și s-au instalat toate dependințele vom apăsa pe butonul Run.

Pentru a rula aplicația web, deschidem folderul price-prediction în Visual Studio Code, apoi se va deschide un terminalul și se va rula comanda npm start pentru a porni aplicația web.

Pentru a crea modelele de predicție, se va deschide Jupyther Notebook din directorul prediction_models și rulați toate celulele pentru a crea modelele.

Pentru a rula aplicația de predicție și colectarea a datlor, se va deschide fisierul app.py în Visual Studio Code și se va utiliza comanda flask run pentru a porni aplicația Flask.

Cu acești pași, veți putea utiliza întreaga aplicație, inclusiv gestionarea utilizatorilor, aplicația web de predicție și modelul de predicție dezvoltat.

7.2. Manual de utilizare

Pentru a ne înregistra un cont în aplicație trebuie să apăsăm pe opțiunea înregistrare din meniu.

Figura 7-1: Pagina principală cu opțiunea înregistrare evidențiată

Apoi vom fii redirectati către pagina cu formularul de înregistrare unde trebuie să completăm toate câmpurile cu date valide și să apăsăm butonul register.

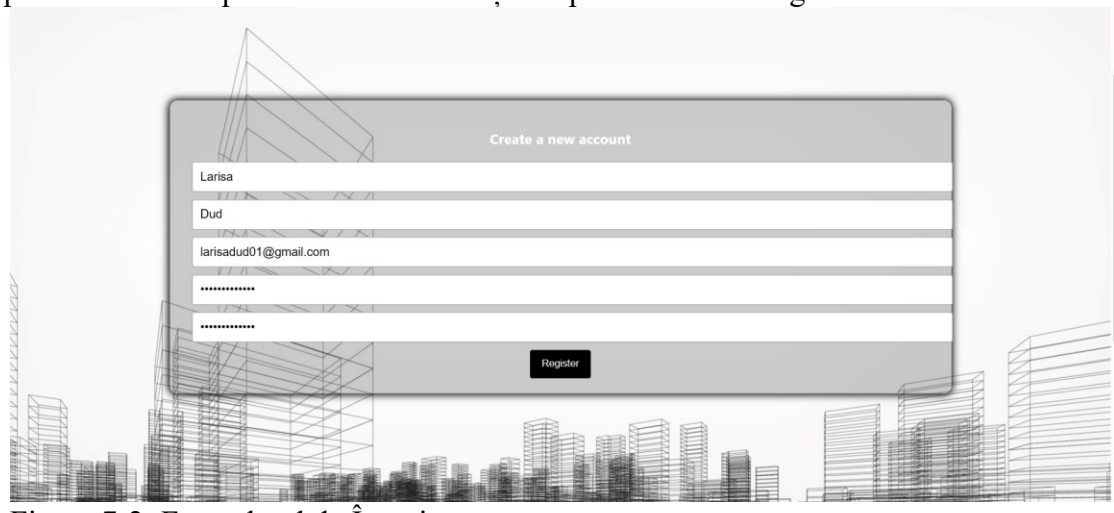


Figura 7-2: Formularul de Înregistrare cont

Pentru a ne autentifica în aplicație trebuie să apăsăm pe opțiunea login din meniu care ne va redirecta către pagina de login unde trebuie să introducem credențialele noastre și să apăsăm butonul de login.

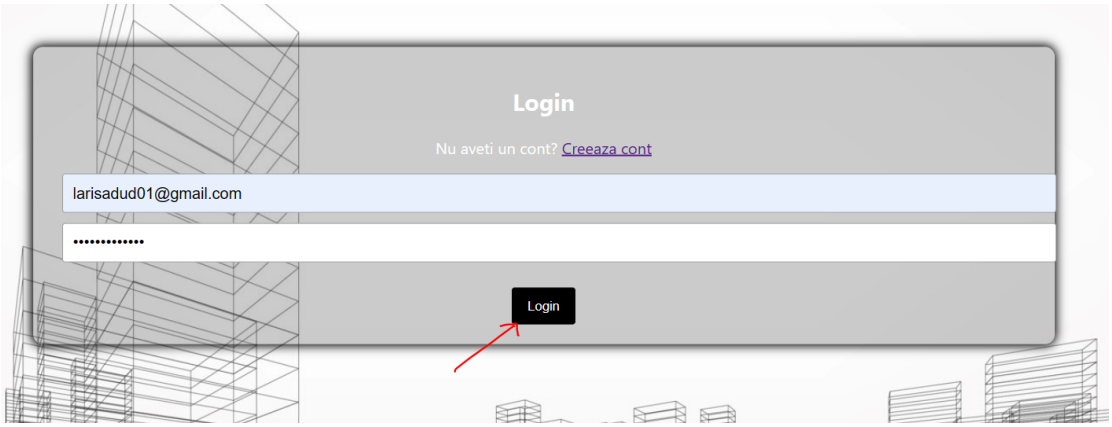
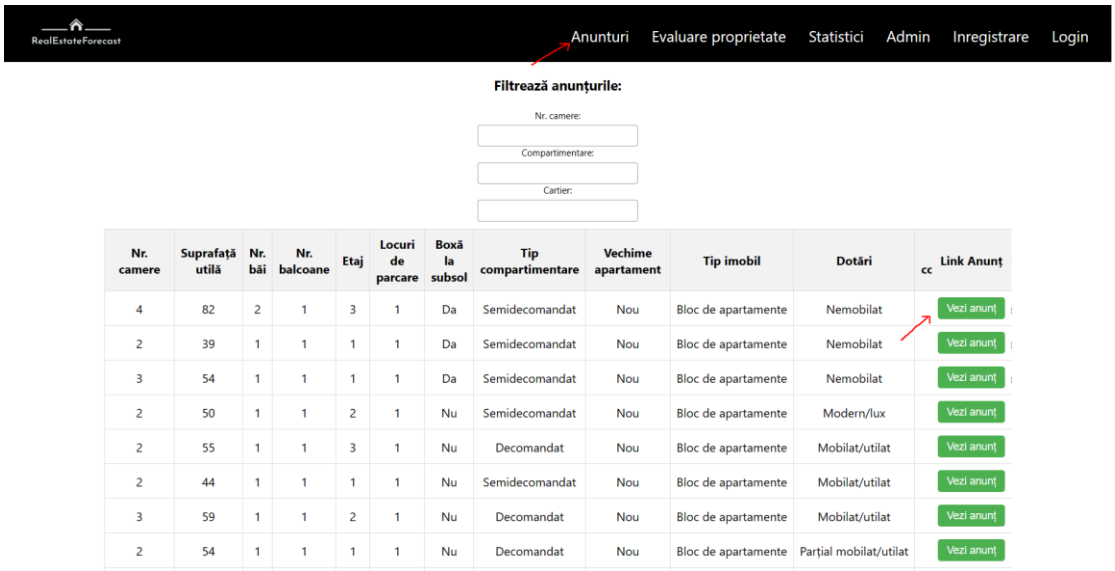


Figura 7-3: Formularul de login

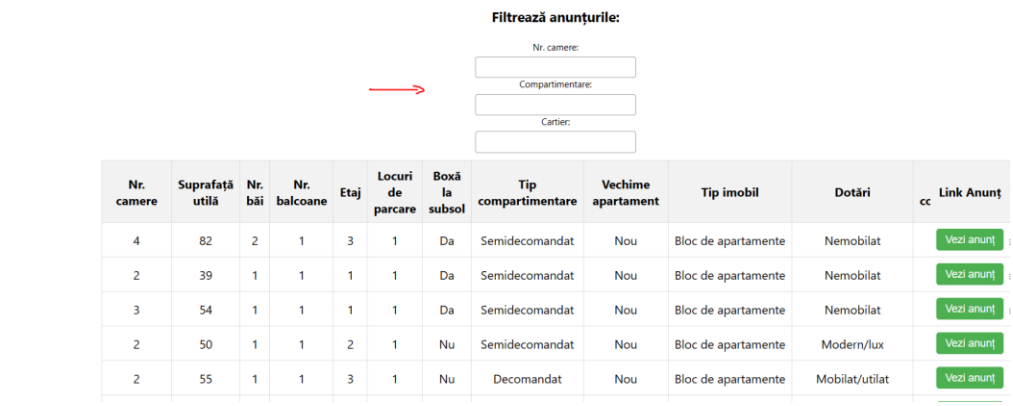
Pentru a vizualiza anunțurile disponibile în setul de date vom selecta opțiunea anunțuri din meniu fiind redirectați către pagina cu anunțuri iar pentru a vedea mai multe detalii despre un anumit anunț din tabel vom da click pe butonul vezi anunț din dreptul fiecărei înregistrări și se va deschide o nouă pagină cu anunțul original.



Nr. camere	Suprafață utilă	Nr. băi	Nr. balcoane	Etaj	Locuri de parcare	Boxă la subsol	Tip compartimentare	Vechime apartament	Tip imobil	Dotări	cc	Link Anunț
4	82	2	1	3	1	Da	Semidecomandat	Nou	Bloc de apartamente	Nemobilat		Vezi anunț
2	39	1	1	1	1	Da	Semidecomandat	Nou	Bloc de apartamente	Nemobilat		Vezi anunț
3	54	1	1	1	1	Da	Semidecomandat	Nou	Bloc de apartamente	Nemobilat		Vezi anunț
2	50	1	1	2	1	Nu	Semidecomandat	Nou	Bloc de apartamente	Modern/lux		Vezi anunț
2	55	1	1	3	1	Nu	Decomandat	Nou	Bloc de apartamente	Mobilat/utilat		Vezi anunț
2	44	1	1	1	1	Nu	Semidecomandat	Nou	Bloc de apartamente	Mobilat/utilat		Vezi anunț
3	59	1	1	2	1	Nu	Decomandat	Nou	Bloc de apartamente	Mobilat/utilat		Vezi anunț
2	54	1	1	1	1	Nu	Decomandat	Nou	Bloc de apartamente	Parțial mobilat/utilat		Vezi anunț

Figura 7-4: Pagina destinată vizualizării anunțurilor colectate

Tot pe pagina de anunțuri pentru a filtra anunțurile afișate trebuie să introducem caracteristicile dorite în câmpurile din secțiunea filtrează anunțurile iar acestea se vor filtra automat fără să fie nevoie de alte acțiuni.



Nr. camere	Suprafață utilă	Nr. băi	Nr. balcoane	Etaj	Locuri de parcare	Boxă la subsol	Tip compartimentare	Vechime apartament	Tip imobil	Dotări	cc	Link Anunț
4	82	2	1	3	1	Da	Semidecomandat	Nou	Bloc de apartamente	Nemobilat		Vezi anunț
2	39	1	1	1	1	Da	Semidecomandat	Nou	Bloc de apartamente	Nemobilat		Vezi anunț
3	54	1	1	1	1	Da	Semidecomandat	Nou	Bloc de apartamente	Nemobilat		Vezi anunț
2	50	1	1	2	1	Nu	Semidecomandat	Nou	Bloc de apartamente	Modern/lux		Vezi anunț
2	55	1	1	3	1	Nu	Decomandat	Nou	Bloc de apartamente	Mobilat/utilat		Vezi anunț

Figura 7-5: Evidențierea campurilor pentru filtrarea anunțurilor

Dacă utilizatorul dorește să evalueze o proprietate va da click pe opțiunea evaluare proprietate din meniu acesta fiind redirectat către pagina corespunzătoare acestei acțiuni. Pe pagina de evaluare, utilizatorul va introduce valorile caracteristicilor proprietății sale în câmpurile aferente iar apoi va apăsa butonul estimează preț urmând ca prețurile să fie afișate în secțiunea rezultat.

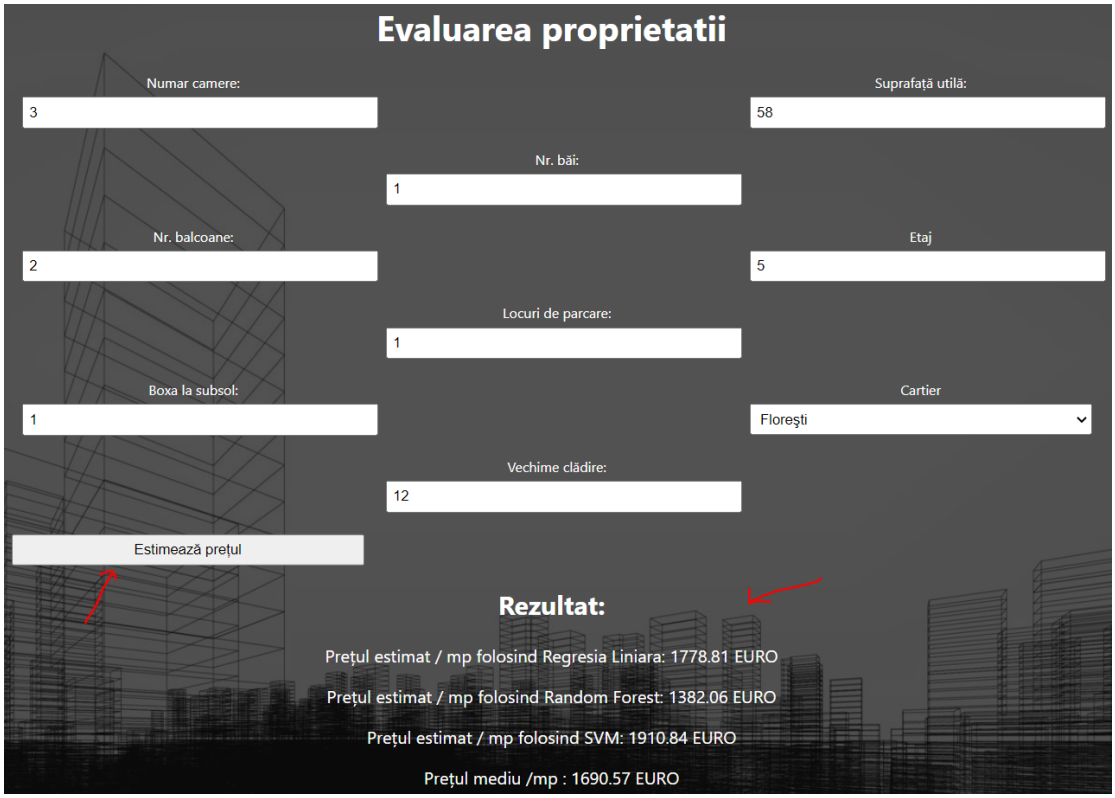


Figura 7-6: Formularul de evaluare a unei proprietăți

Pentru vizualizarea anumitor statistici utilizatorul trebuie să dea click pe opțiunea statistici din meniu acesta fiind redirectat către o pagină care conține un meniu cu toate graficele disponibile iar pentru a le vizualiza trebuie să facă clic pe opțiunea dorita.

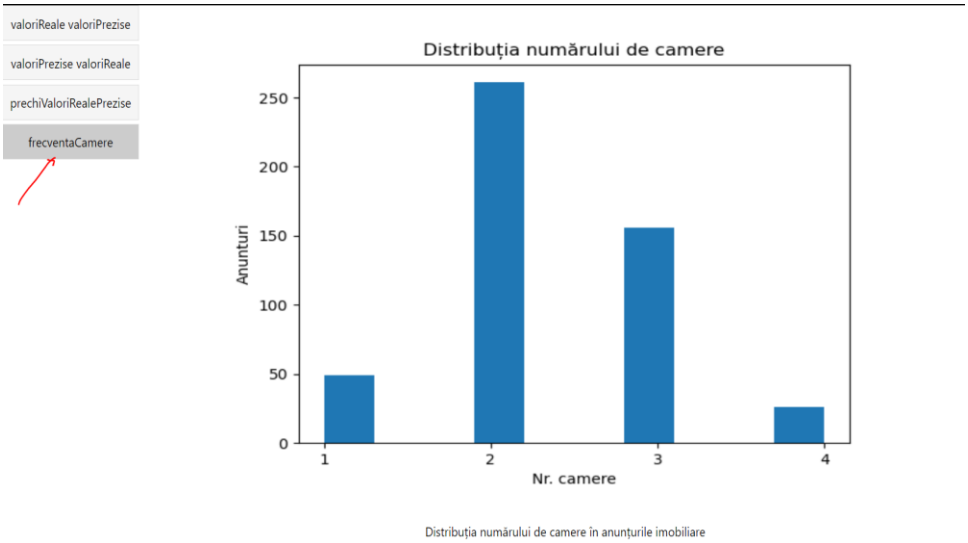


Figura 7-7: Pagina de vizualizare a statisticilor

Pentru a actualiza setul de date folosit pentru modelul de predicție administratorul va accesa pagina destinat strict lui făcând click pe opțiunea admin din meniu după care dintr-un select box selectează dacă colectarea se va face pentru setul de date din Cluj sau din București și apasă butonul actualizare.

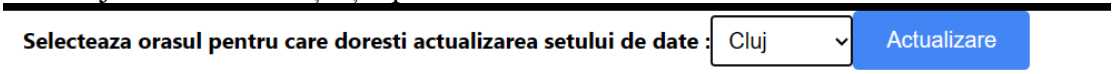


Figura7-8: Pagina administratorului

Pentru a ne deloga trebuie să facem click pe opțiunea Logout din meniu.

Capitolul 8. Concluzii

Acest capitol va include concluziile ce se pot trage in urma realizarii acestei lucrari prezentand o imagine de ansamblu asupra contributiilor noastre, o analiza critica a rezultatelor obtinute si posibile cai de dezvoltare si imbunatatiri ulterioare.

8.1. Contribuții personale

În această secțiune se vor evidenția contribuțiile personale aduse proiectului.

Una dintre contribuțiile mele principale a fost dezvoltarea unui model precis de predicție a prețurilor în domeniul imobiliar. Pentru a realiza acest lucru, am implementat și analizat trei algoritmi diferiți: regresie liniară, Random Forest și regresie vectorială de suport (SVR). În plus, o contribuție semnificativă adusă dezvoltării acestui model a fost preprocesarea datelor, asigurându-mă că datele de intrare au fost rafinate și pregătite pentru modelare precisă. Acest pas a îmbunătățit semnificativ acuratețea predicțiilor și a pus o bază solidă pentru succesul proiectului.

O altă contribuție notabilă pe care am adus-o proiectului a fost crearea unui modul de colectare a datelor. Acest modul a fost esențial în menținerea setului nostru de date actualizat prin colectarea de anunțuri în timp real de pe internet. Această abordare dinamică a îmbogățirii datelor ne-a permis să analizăm comportamentul modelului și pe un set de date mai puțin rafinate și să ne adaptăm în mod eficient la condițiile în schimbare ale pieței.

De asemenea, o altă contribuție e reprezentată de implementarea unui sistem de management al utilizatorilor, permițând procese de înregistrare și autentificare, pe baza de ro, a utilizatorilor în cadrul aplicației noastre.

În plus, pentru a îmbunătăți experiența utilizatorului, m-am concentrat pe crearea unei interfețe grafice intuitive care să încapsuleze toate funcționalitățile pe care le-am dezvoltat. Această interfață ușor de utilizat servește ca o poartă de acces pentru utilizatori pentru a valorifica fără efort puterea modelelor noastre de predicție, a actualizărilor setului de date și a funcțiilor de gestionare a utilizatorilor.

8.2. Analiza critică a rezultatelor obținute

Una dintre realizările-cheie ale acestui proiect este dezvoltarea unui model de predicție precis pentru prețurile din domeniul imobiliar. Acest model a fost construit și optimizat folosind trei algoritmi diferiți: regresie liniară, Random Forest și regresie vectorială de suport (SVR). Rezultatele obținute au demonstrat că modelul Random Forest a obținut cea mai înaltă acuratețe, cu un scor impresionant de 98%.

8.3. Posibile dezvoltări și îmbunătățiri ulterioare

O prima dezvoltare ulterioara a aplicatiei ar fi imbunatatirea afisarii si filtrarii anunturilor, aceasta ar presupune dezvoltarea modului de colectare pentru a prelua si imagini corespunzatoare anunturilor, modificarea modulului de afisare al anunturilor si adaugarea de noi filtre relevante. Acest aspect ar face aplicatia mai accesibilia din punctul de vedere al vizionarii anunturilor.

O alta dezvoltare ar presupune posibilitatea de adaugare de anunturi imobiliare atat de catre admin cat si de catre clienti.

Aplicatia se mai poate dezvolta ulterior si prin imbunatatirea modelului de predictie, implementandu-se si alte metode de predictie a pretului. De asemnea, pentru setul de date cu anunturi colectate, prin colectarea mai multor anunturi in timp, preprocesarea mai amanuntita a acestora si eliminarea anunturilor mai vechi se poate ajunge la o predictie mai buna si relevanta pretului actual al pietei.

Capitolul 9. Bibliografie

[1] "Bucharest House Price Dataset". Available: https://www.kaggle.com/datasets/denisadutca/bucharest-house-price-dataset. [Accesat 15 martie 2023].
[2] "About Linear Regression | IBM". Available: https://www.ibm.com/topics/linear-regression. [Accesat 20 aprilie 2023].
[3] "Linear Regression Analysis - an overview | ScienceDirect Topics". Available: https://www.sciencedirect.com/topics/medicine-and-dentistry/linear-regression-analysis. [Accesat 20 aprilie 2023].
[4] "Linear Regression", *Google Books*. Available: https://books.google.com/books/about/Linear_Regression.html?hl=ro&id=IvAw_1MTASsC. [Accesat 21 aprilie 2023].
[5] Hujia Yu, Jiafu Wu, Real Estate Price Prediction with Regression and Classification, CS 229 Autumn, 2016.
[6] House Price Prediction Using Machine Learning
[7] Abigail Bola Adetunji, Oluwatobi Noah Akande, Funmilola Alaba Ajala, Ololade Oyewo, Yetunde Faith Akande, Gbenle Oluwadara, "House Price Prediction using Random Forest Machine Learning Technique", *Procedia Computer Science*, Volume 199, pp. 806-813, 2022.
[8] M. Khder, "Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application", *International Journal of Advances in Soft Computing and its Applications*, vol. 13, no. 3, pp. 145–168, Noiembrie 2021.
[9] Lin, Hong & Chen, Kuentai, Predicting price of Taiwan real estates by neural networks and support vector regression, pp. 220-225, Iulie 2011.
[10] João Manuel Azevedo Santos, Real Estate Market Data Scraping and Analysis for Financial Investments, 2018.
[11] B. A. Manko, "Teaching user-friendly web design: A case study on Zillow.com in the real estate industry," *Journal of Information Technology Teaching Cases*, vol. 12, no. 1, pp. 35–42, Martie 2021.