UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA
DE
MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

Proiect de diplomă

Random Forest Regression – Pentru evaluarea prețurilor în imobiliare

Absolvent,
Pohrib Eduard-Emanuel

Coordonator științific,
Conf. Dr. Cristian Kevorchian

București, iulie 2021

Rezumatul proiectului de diplomă

Proiectul de diplomă are ca scop antrenarea unui model de învățare automată bazat pe algorimul de pădure aleatoare (Random Forest Regression) ce poate estima cât mai aproape de realitate, un interval de preț în care ar trebui să se încadreze chiria lunară a unui apartament din București. Modelul va fi pus la dispoziția publicului larg prin intermediul unei aplicații web găzduite în cloud care va oferi utilizatorilor un interval de preț indicat pentru chiria lunară. Utilizatorul poate introduce manual detaliile apartamentului, sau în mod automat prin intermediul unui URL de pe o pagină web de imobiliare, apoi aplicația estimează intervalul de preț și se precizează, după caz, dacă apartamentul este evaluat corect, supraapreciat sau subapreciat.

În momentul de față, pentru o estimare corectă a prețului chiriei este necesară intervenția unui evaluator uman. Aplicația dezvoltată în acest proiect de diplomă își propune să înlocuiască acest evaluator și să facă mai accesibilă, atât pentru chiriași cât și pentru proprietarii de apartamente, evaluarea corectă și gratuită a prețului chiriei lunare în piața imobiliară curentă.

Modelul antrenat, nu poate prezice cu o acuratețe foarte mare prețul apartamentelor de pe piață deoarece unul dintre cei mai importanți factori de decizie în stabilirea prețurilor este factorul uman emoțional. Având în vedere că acest factor nu poate fi cuantificat și utilizat în antrenarea modelului, rezultatele prezise sunt corecte pentru doar aproximativ o treime dintre apartamentele folosite pentru test, făcând o medie a pieței.

Bachelor Project summary

The Batchelor Project aims to train a machine learning model that uses the Random Forest Regression algorithm, that can estimate as close as possible to reality, a price range that should include the monthly rent of an apartament in Bucharest. The model will be made available to the public through a web application hosted in the cloud that will offer users a price range indicated for the monthly rent. The user can enter the details of the apartment manually, or automatically via a URL on a real estate web page, then the application estimates the price range and specifies whether the apartment is valued correctly, overvalued or undervalued.

Right now, the intervention of a human appraiser is necessary for a correct estimate of the rental price. The application developed in this project aims to replace this appraiser and make it more accessible for both tenants and apartment owners, the correct and free evaluation of the monthly rental price in the current real estate market.

The trained model cannot estimate with a very high accuracy the price of the apartments on the market because one of the most important decision factors in setting the prices is the human emotional factor. Given that this factor cannot be quantified and used in training the model, the results presented are correct for only about a third of the apartments used for the test, making a market average.

Cuprins

1. Introducere

1.1 Motivație

Termenul de imobiliar este folosit pentru a descrie orice bucată de teren împreună cu orice este atașat permanent de acel teren, cum ar fi clădiri, locuințe, resurse naturale și alte structuri permanente. Accesibilitatea prețurilor imobiliare și a prețului de închiriere, precum și modificările acestor prețuri au un impact direct asupra bogăției proprietarilor și chiriașilor. Aproape toate instituțiile majore, precum Banca Centrală Europeană, OCDE, Fondul Monetar Internațional și Comisia Europeană, se concentrează asupra dinamicii prețurilor imobiliare și asupra factorilor influenți ai acestora. Cercetătorii și politicienii consideră din ce în ce mai mult că piețele imobiliare joacă un rol decisiv în transmiterea impulsurilor de politică monetară. Există o strânsă legătură între piața de imobiliare și stabilitatea economică a unei țări, deoarece bunurile imobiliare reprezintă o mare parte din bogăția individuală și a afacerilor din sectoarele economice. Atunci când prețurile imobilelor cresc, bogăția crește, astfel încât persoanele fizice și întreprinderile sunt mai susceptibile să împrumute și să cheltuiască.

Chiar dacă este un fel de „lege nescrisă”, prețul unei chirii se aliniază unui preț general, valabil într-o anumită perioadă, într-o anumită țară, oraș, regiune. Studiile de specialitate compară prețul locuințelor și al chiriilor din 2007 cu cel din 2019 și demonstrează faptul că, în medie, prețul acestora la nivel european a crescut cu aproape 20%. Conform statisticii, în România, creșterea este chiar mai mare - aproximativ 55%, explicația fiind dată, în principal, de creșterea valorii euro. Apecierea prețului unui imobil reprezintă o etapă importantă în procesul de vânzare, cumpărare sau închiriere. Un specialist, numit și evaluator poate evalua prețul corect justificat de un raport detaliat și poate oferi părților implicate siguranța unei investiții corecte.. Acest specialist, numit și evaluator primește informații despre imobil(data de construcție a clădirii, dimensiunile, echipamentele clădirii, împrejurimi etc.) și folosind diferite abordări bazate pe date de piață stabilește un preț corect pentru vânzarea sau închirierea apartamentului.

Motivația din spatele acestei lucrări a fost dorința de a elimina evaluatorul din procesul de estimare a chiriei pentru apartamentele din București și înlocuirea acestuia cu o alternativă simplă, modernă și eficientă prin care prețul este estimat într-o manieră obiectivă, utilizând metode științifice și matematice complexe. Din cauza creșteii accentuate a numărului de

descoperiri tehnologice este de dorit o schimbare a abordării actuale. În ultimii ani, se pune mai mult accent pe realizarea de sisteme autonome bazate pe învățarea automată, pentru a răspunde nevoilor utilizatorilor.

1.2 Obiectivul lucrării

Proiectul de diplomă a avut ca obiectiv antrenarea și utilizarea unui model de învățare automată bazat pe algoritmul de pădure aleatoare (Random Forest Regression) ce poate estima cât mai aproape de realitate, un interval de preț ce încadrează chiria lunară pentru apartamentele din București. În momentul actual, nu există nicio astfel de resursă gratuită care să fie disponibilă pentru publicul larg. Cea mai similară soluție existentă pe piața este aplicația web „prețExpert" implementată de echipa de la www.imobiliare.ro, dezavantajul principal al acesteia fiind prețul ridicat de 5 euro pentru fiecare proprietate analizată.

Am realizat o astfel de aplicație web, accesibilă de oriunde prin intermediul internetului, ușor de utilizat, gratuită, ce oferă un avantaj utilizatorilor asupra înțelegerii pieței imobiliare. Aplicația vine în ajutorul persoanelor ce își doresc sa ofere spre închiriere un apartament în București, dar și în ajutorul chiriașilor care își caută o nouă locuință și doresc să fie informați corect cu privire la valoarea reală a unui imobil.

1.3 Strucrura lucrării

Lucrarea a fost împărțită în trei capitole ce reflectă pașii prin care s-a realizat aplicația de estimare a prețului chiriei pentru apartamentele din București folosind algoritmul de pădure aleatoare.

În Capitolul 2 al lucrării sunt prezentate conceptele teoretice ce stau la baza învățării automate și a algoritmului de pădure aleatoare. Subcapitolele detaliază principalele aspecte teoretice avute în vedere pe parcursul dezvoltării, dar și în evaluarea modelului antrenat.

În Capitolul 3 au fost introduse tehnologiile principale utilizate în dezvoltarea și rularea aplicației. Acest capitol a avut 3 părți importante: partea de programare propriu-zisă care a introdus limbajul și principalele librării folosite, partea de baze de date în care a fost prezentat limbajul MySQL și o parte de cloud computing cu o introducere în serviciile furnizate de Amazon Web Services.

Ultimul capitol prezintă metoda de implementare a aplicației cu fiecare modul dezvoltat pentru îndeplinirea cu succes a obiectivelor propuse. Pe parcursul acestuia vor fi prezentate atât modul de utilizare a aplicației, cât și componentele acesteia cu scopul evidențierii funcțiilor individuale. Proiectul a avut patru părți majore: antrenarea modelului de învățare automată folosind algoritmul de pădure aleatoare, implementarea părții de backend și de frontend a aplicației web, precum și găzduirea acesteia în cloud.

În final au fost prezentate concluziile din urma dezvoltării proiectului de diplomă, fiind evidențiate contribuțiile personale și aspectele din cadrul proiectului care pot avea parte de îmbunătățiri.

2. Noțiuni teoretice

2.1 Învățarea automată

Învățarea automată (în engleză „Machine Learning”) este o aplicație a inteligenței artificiale care oferă computerelor capacitatea de a învăța singure și de a-și îmbunătăți performanțele din experiență, acestea nefiind programate special pentru realizarea unei sarcini. Algoritmii de învățare automată au o aplicabilitate vastă în domenii precum ar fi în medicină, economie și în computer vision, unde este foarte greu sau imposibil să se dezvolte algoritmi convenționali sau resurse umane pentru a rezolva problemele necesare. În practică, pentru sarcini avansate, este mai eficient ca oamenii să dezvolte aplicații de învățare automată decât să creeze manual algoritmii necesari.

Câteva realizări specifice oferă o privire a stadiului actual al învăţării automate: au fost dezvoltate programe pentru recunoaşterea automată a vorbirii, pentru prezicerea ratei de recuperare a pacienților cu pneumonie, detectarea utilizării frauduloase a cardurilor de credit, conducerea autonomă a vehiculelor, sau să joace jocuri, cum ar fi tablele, la nivelul care se apropie de performanța campionilor mondiali umani. [1]

Procesul de învățare este etapa în care un model primește date pentru a-și adapta parametrii astfel încât să generalizeze cât mai bine pe datele de antrenare cât și pe altele noi, necunoscute.

În loc de scrierea unui algoritm ce rezolvă fiecare sarcină, sunt colectate o selecție de exemple care definesc o ieșire dorită pentru o anumită dată de intrare. Un algoritm de machine learning va folosi aceste exemple și va produce rezultat ce rezolvă problema. Dacă programul rezolvă corect, acesta va oferi rezultatul așteptat și pentru intrări de date noi, dar și pe cele pe care a fost antrenat. Dacă datele își schimbă formatul sau rezultatul așteptat în urma anumitor intrări, algorimul se va modifica şi el prin antrenarea pe noi date.

Aplicațiile de învățare automată se pot clasifica în funcție de scopul urmărit, astfel:

- Sisteme inteligente pentru predicţii - acestea au scopul de a folosi un model antrenat anterior pentru a oferi o predicție pe noile date de intrare
- Sisteme inteligente pentru regresii – acestea au scopul de a folosi un model antrenat anterior pentru a defini forma unei funcții univariantă sau multivariantă.
- Sisteme inteligente pentru clasificare – acestea au scopul de a folosi un model antrenat anterior pentru a încadra o intrare de date în una sau mai multe clase(numite și categorii) folosindu-se de caracteristicile acestora
- Sisteme inteligente pentru planificare – acestea au scopul de a genera pașii optimi ce trebuie urmați pentru a se realiza cu succes o sarcină [2]

În particular, în domeniul imobiliarelor se dezvolta din ce în ce mai multe aplicații, cu diferite scopuri, ce folosesc învățarea automată. Cateva exemple ar fi: aplicații pentru consumatori, Chatbots și interfețe bazate pe învățarea automată, automatizarea în administrarea proprietăților și aplicații de prognoza pe piața imobiliara.

Există trei paradigme în procesul de învățare bazate pe: învățarea supervizată, învățarea nesupervizată și învățarea întărită. În cadrul învățării supervizate fiecare informație transmisă modelului în timpul antrenării este o pereche formată din obiectul de intrare denumit și eșantion, împreună cu eticheta sau valoarea de ieșire corespunzătoare. Modelul realizează o mapare între o intrare dată și ieșirea corespunzătoare, pe baza a ceea ce a învățat din datele de antrenare etichetate.

În cazul învățării nesupervizate, datele de antrenare nu vin însoțite de etichete, scopul acestor algoritmilor este acela de a împarți datele în “clustere” sau grupuri, în funcție de atributele acestora printr-un mecanism de competiție internă. Membri unui astfel de grup trebuie să fie cât mai similari unul față de celălalt, iar distanța dintre clustere să fie cât mai mare.

Învățarea întărită nu beneficiază de datele de antrenare însoțite de etichete ca în cazul învățării supervizate, datele vin însoțite de un răspuns binar care oferă o informație calitativă (răspuns bun sau greșit) despre cât de bine și-a atins rețeaua scopul urmărit. Astfel, rețeaua va ști că a greșit dar nu și cu cât a greșit. Rețeaua va fi pedepsită atunci când va produce același rezultat nedorit, iar tendința de a repeta greșelile va scădea. [3]

Învățarea automată poate fi împărțită în numeroase grupuri de algoritmi, fiecare dintre ele fiind utilizate pentru a rezolva un anumit tip de problemă. Printre aceste grupuri se numără:

- Algoritmi de regresie care se referă la modelarea relației dintre variabile, sunt folosite pentru calculele statistice și învățarea automată a statisticilor. Dintre cei mai populari algoritmi enumerăm: regresia liniară, regresia logistică, regresia în trepte, regresia ordinară a celor mai mici pătrate.
- Algoritmi de regularizare care penalizează modelele pe baza complexității lor, favorizând modelele mai simple, cu aptitudini mai bune de generalizare. Câțiva algoritmi cunoscuți sunt: regresia Ridge, regresia cu unghi minim.
- Metoda arborelui decizional care construiește un model de decizii luate pe baza caracteristicilor reale ale atributelor din date. Arborii de decizie sunt instruiți pentru probleme de clasificare și regresie, fiind a alegere populară în învățarea automată. Printre aceștia se numără: arborele de clasificare și regresie, arbori de decizie condiționată.
- Metodele bayesiene sunt folosite pentru probleme de clasificare și regresie și au la bază teorema lui Bayes. Câțiva algoritmi cunoscuți: Naive Bayes, Multinomial Naive Bayes și Gaussian Naive Bayes.
- Clusterizarea este un algoritm de învățare nesupervizat care organizează datele în cel mai bun mod, în grupuri, neavând informații privind etichetele datelor. Printre algoritmi se numără: clusterizarea ierarhică și maximizarea așteptărilor.
- Rețelele neuronale artificiale care sunt inspirate din structura și funcționalitatea rețelei neuronale umane. Sunt una din cele mai populare metode cu un domeniu vast de aplicabilitate, fiind utilizate foarte uzual în probleme de clasificare și regresie. Algoritmi: perceptron, perceptronul multistrat, propagarea înapoi și gradientul stochastic descendent.
- Algoritmi de învățare adâncă care se ocupă cu construirea unor rețele neuronale mult mai mari și mai complexe, folosind seturi mari de date. Acest grup este specializat în prelucrarea de date analogice precum imagini, text video și audio. Algoritmii de învățare adâncă se bucură de o popularitate în continuă creștere, fiind folosiți în foarte multe domenii. Dintre aceștia amintim: rețele neuronale convoluționale, rețele neuronale recurente, rețele de memorie pe termen scurt. [4]

2.1.1. Rețele neuronale artificiale

Rețelele neuronale artificiale sunt construite asemănător cu creierul uman. Unitatea elementară este reprezentată de un neuron artificial numit și nod, care încearcă să imite structura şi funcţionarea unui neuron biologic. Rețelele neuronale sunt alcătuite din mii de altfel de neuroni artificiali care primesc la intrare, diferite forme și structuri de informații bazate pe un sistem intern de ponderi. [5]

Neuronii sunt așezați în straturi, în care ieșirea fiecărui strat formează intrarea stratului următor. La cel mai înalt nivel, există 3 tipuri de straturi, conectate secvenţial între ele:

- Stratul de intrare
- Straturile ascunse
- Stratul de ieșire

Aceste rețele au ca trăsătură principală capacitatea de a învăța folosind exemplele și experiența dobândită anterior pentru a-și îmbunătăți performanțele și de a le face disponibile pentru următoarele utilizări. La intrarea rețelei sunt aduse exemple de antrenare care circulă prin rețea începând de la stratul de intrare și deplasându-se prin straturile ascunse până la stratul de ieșire. Rețelele efectuează prelucrări ale datelor secvențial, pe fiecare strat, unul după altul, utilizând funcții de activare, derivabile, care să permită antrenarea.

Parcurgerea unui exemplu de antrenare, a unui proces complet față - spate poartă denumirea de iterație, iar trecerea prin toate exemplele de antrenare se numește epocă. În scopul minimizării erorii, antrenarea se face în mai multe epoci.

2.1.2. Arbori de decizie

Arborii de decizie sunt o metodă de învățare supervizată non-parametrică utilizată pentru clasificare și regresie ce poate gestiona în mod eficient seturi de date mari și complicate fără a impune o structură parametrică complicată. Arborii de decizie învață din date pentru a aproxima o curbă sinusoidală cu un set de reguli de decizie de tipul dacă - atunci - altfel. Cu cât arborele este mai adânc, cu atât regulile de decizie sunt mai complexe. Într-un arbore de decizie, pentru a prezice o etichetă de clasă, se pornește din nodul “rădăcină” și se vor compara valorile din

rădăcina cu cele ale înregistrării curente. Pe baza comparației, se urmărește ramura corespunzătoare acelei valori și se trece la următorul nod. Algoritmul se oprește atunci când se ajunge într-un nod numit “frunza” sau nod terminal. Arborii de decizie sunt o familie de tehnici analitice ce include CHAID (Chi-square Automatic Interaction Detector) și CART (arbori de clasificare și regresie, în engleză: „ Classification and Regression Trees”).

Arborii de clasificare și regresie sunt ideali pentru analiza datelor complexe. Pentru astfel de date, avem nevoie de metode analitice flexibile și robuste, care pot face față relațiilor neliniare, interacțiunilor de ordin înalt și valorilor lipsă. În ciuda acestor dificultăți, metodele trebuie să fie ușor de înțeles și să dea rezultate ușor de interpretat. Acești arbori pot prezice variația unei singure variabile, numită și etichetă, prin împărțirea repetată a datelor în grupuri cât mai omogene, folosind diferite combinații de variabile explicative. Fiecare grup este caracterizat de o valoare a etichetei, de numărul de observații din grup și de valorile variabilelor explicative pe care o definesc. Arborele este reprezentat grafic, iar acest lucru ajută explorarea și înțelegerea. [6]

Arborii de decizie utilizează mai mulți algoritmi pentru a decide împărțirea unui nod în două sau mai multe subnoduri. Crearea de subnoduri crește omogenitatea subnodurilor rezultate. Cu alte cuvinte, putem spune că puritatea nodului crește în raport cu variabila țintă. Arborele decizional împarte nodurile pe toate variabilele disponibile și apoi selectează împărțirea care are ca rezultat cele mai multe subnoduri omogene.

Algoritmul ID3 construiește arbori de decizie folosind o abordare de căutare lacomă de sus în jos prin spațiul unor posibile ramuri, fără backtracking. Un algoritm lacom, așa cum sugerează și numele, face întotdeauna alegerea care pare a fi cea mai bună în acel moment.

Pași pentru algoritmul ID3:

1. Începe cu setul original S ca nod rădăcină
2. La fiecare iterație a algoritmului, iterează prin atributul cel mai neutilizat al setului S și calculează “Entropia” și “Câștigul de Informații” al acestui atribut.
3. Apoi selectează atributul care are cea mai mica entropie sau cel mai mare câștig de informație.

4. Setul S este apoi împărțit de atributul selectat pentru a produce un subset de date.
5. Algoritmul continuă să se repete pe fiecare subset, luând în considerare numai atributele selectate până acum. [7]

Arborii de decizie sunt un instrument statistic puternic pentru clasificare, predicție și manipularea datelor. Avantajele utilizării acestei metode de învățare automată sunt:

- Este ușor de înțeles și de interpretat, chiar și pentru utilizatorii ce nu au foarte multă experiență în domeniu
- Este un metodă de învățare robustă
- Simplifică relațiile complexe dintre variabilele de intrare și variabilele țintă prin împărțirea variabilelor de intrare originale în subgrupuri semnificative
- Valorile lipsă sunt ușor de înlocuit fără a afecta performanța modelului

O problemă apare atunci cand nu este stabilită nicio limită pentru un arbore de decizie, acesta va oferi o acuratețe de 100% pe setul de antrenare deoarece, în cel mai rău caz, va ajunge să facă câte o frunză pentru fiecare înregistrare. Astfel, aceasta afectează acuratețea atunci când se prezic date folosind eșantioane care nu fac parte din setul de antrenare. O metodă de a scapă de "overfitting" (în traducere "supraantrenare") este algoritmul de pădure aleatorie. Un model supraantrenat este un model ce este util doar pe baza de date folosită pentru antrenare și nu pentru alte date din exterior.

2.2 Algoritmul de pădure aleatorie (Random Forest Regression)

S-a demonstrat că pentru clasificare cât și pentru regresie, utilizând ansambluri de arbori acuratețea crește semnificativ. Predicțiile finale sunt obținute prin agregarea(votarea) de către ansamblul de arbori, de obicei, aceștia având greutate decizională egală. „Bagging" reprezintă un exemplu timpuriu de implementare în care arborii sunt construiți folosind eșantioane din caracteristicile datelor de intrare. Reducerea variației crescute a deciziei finale este rezultată din votul cu greutate egală ar arborilor. Pentru a se ajunge la o variantă și mai scăzută se poate reduce corelația dintre cantitățile de date mediate. Acesta este principiul ce stă la baza algoritmului de pădure aleatoare.

Pădurile aleatoare încearcă sa efectueze o reducere a corelației dintre caracacteristici printr-o injecție suplimentară de randomitate. În loc de a determina împărțirea optimă a unui nod dat din arbore prin evaluarea tuturor împărțirilor admise pe toate variabilele, se folosește un subset al acestora alese la întâmplare. Cunoscutul statistician Leo Breiman de la Universitatea din California susține că pădurile aleatorii se bucură de o precizie de predicție excepțională și că aceasta este atinsă pentru o gamă largă de setări ale parametrilor utilizați. [8]

Algoritmul pădure aleatoare este un exemplu de învățare, ce are la bază arborii decizionali, dar în care se combină mai mulți algoritmi de învățare automată pentru a obține o performanță predictivă mai bună. Arborii decizionali sunt creați prin extragerea unui subset de date de antrenare prin înlocuire. Aceasta înseamnă că poate fi selectat de mai multe ori același eșantion, în timp ce altele pot sa nu fie selectate niciodată. Aproximativ două treimi din datele disponibile sunt folosite pentru antrenarea arborilor, în timp ce restul de o treime sunt utilizate într-o tehnica de validare încrucișată pentru a estima cât mai bine performanța modelului.

Fiecare arbore de decizie este produs independent fără nicio tăiere și fiecare nod este împărțit utilizând un număr de caracteristici definite de utilizator. Prin creșterea pădurii până la un număr de copaci definit de utilizator, algoritmul creează copaci care au varianță mare și părtinire scăzută. Decizia finală de clasificare se ia prin calcularea (utilizând media aritmetică) a probabilităților de atribuire a clasei calculate de toți arborii produși și apoi se calculează eroarea medie folosindu-se fie media pătratică, fie media absolută a erorii. O nouă intrare de date, fără etichetă, este astfel evaluată în raport cu toți arborii de decizie creați în ansamblu și fiecare arbore votează pentru o etichetă. [9] Clasa cu voturile maxime va fi cea care este selectată în cele din urmă după cum poate fi observat și în figura 2.1.

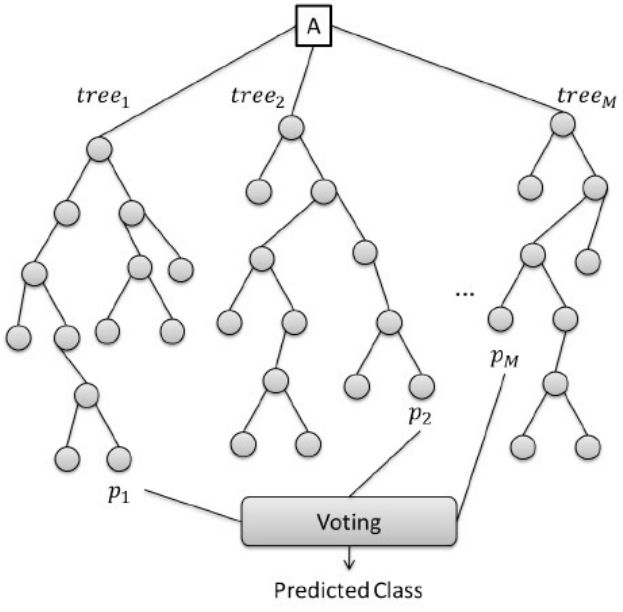


Fig. 2.1

Pădurile aleatoare pot fi de două tipuri: clasificatoare sau regresori.

O pădure aleatoare clasificatoare lucrează cu date ce au etichete discrete, cunoscute și sub numele de clasă. Acestea nu pot prezice o etichetă în afara intervalului de etichete cunoscut. Un exemplu de astfel de clasificator: software ce prezice dacă o recenzie este pozitivă sau negativă analizând textul acesteia.

O pădurea aleatoare bazată pe regresie (numit în engleză „Random Forest Regresor”) prezice date ce au o ieșire numerică continuă și nu pot fi definite în clase. Un exemplu de aplicație: estimarea prețului chiriei lunare, tema acestui proiect de diplomă.

Principalele avantaje al folosirii algoritmilor bazați pe păduri aleatoare sunt:

- Reducerea supraantrenării și ajută la îmbunătățirea acurateței de prezicere
- Poate fi folosit atât în probleme de regresie, cât și în probleme de clasificare
- Nu este necesară normalizarea datelor
- Funcționează bine atât cu valori continue cât și cu valori categorice

Cu toate acestea, există și câteva dezavantaje în folosirea algoritmilor bazați pe păduri aleatoare, și anume: aceștia necesită multă putere de calcul, deoarece se construiesc numeroși arbori și necesită mult timp pentru antrenare, mai ales în cazul seturilor de date foarte mari.

3. Tehnologii utilizate

3.1 Limbajul Python

Limbajul Python este limbajul de programare principal folosit în elaborarea lucrării. Acesta se bucură de o popularitate în creștere din cauza simplității codului, sintaxa le ofera dezvoltatorilor posibilitatea să dezvolte algoritmi într-o modalitate mai clară și mai concisă, în timp ce alte limbaje de programare ce au o sintaxă dificilă. Python este un limbaj de programare dinamic, de nivel înalt, nu este compilat, ci interpretat. Avantajul cel mai mare al acestui limbaj îl constituie existența de librării sau biblioteci predefinite gratuite care simplifică implementarea algoritmilor și metodelor necesare aplicației. Unele din cele mai folosite librării ale limbajului Python sunt: Pandas, Numpy, Scikit-learn, TensorFlow, Scrapy, Flask, SciPy.

Privitor la paradigma de programare, Python poate fi folosit ca limbaj pentru aplicații de tipul obiect-orientat, dar permite și o abordare funcțională, imperativă sau procedurală. Limbajul este ușor de utilizat atât de dezvoltatori cu experiența în alte limbaje de programare dar și de utilizatori începători.

Multitudinea de librării predefinite, face ca limbajul Python să fie foarte folosit in aplicațiile de inteligență artificială și învățare automată. Bibliotecile Python oferă elemente de bază care permit anumite funcționalități sau efectuarea de diverse acțiuni, astfel încât dezvoltatorii nu trebuie să le codifice de la bun început de fiecare dată. Programele de învățare automată necesită o procesare continuă a datelor iar bibliotecile Python ofera uneltele potrivite pentru accesarea, gestionarea și transformarea seturilor de date. Dezvoltatorii pot să-și indrepte atenția către rezolvarea problemelor de învățare automată în loc să se concentreze pe nuanțele tehnice ale limbajului. [10]

Pentru gestionarea acestor librării s-au folosit aplicații precum Pip și Conda, care se ocupă cu descărcarea și instalarea librăriilor dar și cu asigurarea existenței dependințelor necesare. Pip este managerul de pachete standard pentru limbajul Python, permite instalarea și gestionarea de pachete suplimentare care nu fac parte din biblioteca standard Python. Utilizarea managerului Pip simplifică procesul de accesare a librăriilor, fiind foarte prietenoasă cu utilizatorii. [11]

3.2 MySQL

MySQL este un sistem folosit pentru gestionara bazelor de date relaționale open-source, dezvoltata în limbajele C++ și C, fiind accesibil pe peste 20 de platforme, precum Windows, Mac, Linux și Unix. Bazele de date relaționale organizează datele în diferite tabele, fiecare având specificații proprii, aceste tabele fiind legate între ele prin chei, structurarea datelor făcându-se folosind aceste relații, Limbajul folosit pentru extragerea, modificarea și crearea datelor este SQL, acesta fiind folosit si pentru controlul permisiunilor utilizatorilor la baza de date.. Implementarea bazelor de date relaționale se face cu ajutorul MySQL, acesta gestionează conturile utilizatorlor, testează integritatea bazei de date și crează copii de rezerveră, numite si backup-uri.

MySQL folosește un model client - server. Nucleul sistemului este reprezentat de un server MySQL, ce are rolul de a administra toate comenzile aplicate pe baza de date. Acest server este disponibil ca program separat pentru utilizare într-o instanță ce utilizează rețeaua client-server dar și ca bibliotecă externă ce poate fi legată de diferite aplicații. Instrucțiunile sunt trimise la server folosindu-se un client MySQL, instalat pe computer-ul de pe care se lucrează. MySQL este de obicei instalat pe o singură mașină, dar prezintă și facilitatea de a trimite o bază de date în diferite locații, utilizatorii fiind capabili să o acceseze prin diferite interfețe de client MySQL. Interfețele trimit instrucțiunile SQL către server și apoi rezultatele sunt afișate utilizatorului. Pentru securitate, MySQL folosește privilegiile de acces și un sistem de parole criptate ce permit verificarea identității de către gazdă. [12]

MySQL a fost conceput pentru a putea fi utilizat împreună cu alte sisteme, acesta suportă implementarea în medii virtualizate, cum ar fi Amazon Aurora pentru MySQL, Amazon RDS pentru MySQL și Amazon RDS pentru MariaDB. Astfel, utilizatorii au posibilitatea de a-și migra datele aflate într-o bază de date locală folosind diferite instrumente precum AWS Database Migration Service.

3.3 Amazon Web Services

“Cloud Computing” reprezintă o paradigma tehnologică modernă prin care dezvoltatorii au acces la cerere, prin internet, la diferite resurse de calcul: aplicații, servere (fizice sau virtuale),

stocare de date, instrumente de dezvoltare și multe altele. Acestea sunt gazduite de un centru de date la distanță administrat de un furnizor de servicii cloud. Furnizorii pun aceste resurse la dispoziție utilizatorilor independenți, dar și companiilor pentru un abonament lunar sau le facturează serviciile în funcție utilizare.

Dintre cele mai populare tipuri de cloud computing enumerăm:

- Cloud public - este cel mai răspândit tip și resursele cloud sunt deținute și operate de un furnizor terț de servicii cloud și livrate prin internet. În cloud-ul public, furnizorul de cloud deține toate componentele hardware și software. Câteva exemple: Google Cloud Platform, Amazon Web Services, Microsoft Azure etc.
- Cloud privat - reprezintă o suită de resurse de cloud computing utilizate exclusiv de o companie. Pe lângă avantajele unui cloud public (autoservire, scalabilitate și elasticitate), cloud-ul privat oferă personalizarea resurselor dedicate și o securitate ridicată atât prin firewall-urile companiei, cât și prin găzduirea internă pentru a se asigura că operațiunile și datele sensibile nu sunt accesibile furnizorilor terți.
- Cloud hibrid - reprezintă o soluție care combină cloud-ul privat cu unul sau mai multe servicii de cloud public. Acesta oferă companiilor o flexibilitatea de a muta sarcinile de lucru între cele soluțiile de cloud în funcție de nevoi și costuri.

Amazon Web Services (AWS) este cel mai mare furnizor de servicii cloud de pe piață ce oferă o varietate de instrumente de infrastructură tehnică. Unul dintre cele mai cunoscute servicii oferite de aceștia este Amazon Elastic Compute Cloud (sau EC2), care permite utilizatorilor să aibă la dispoziție un cluster virtual de computere, personalizate după cerințele și nevoile fiecăruia, disponibile tot timpul, prin intermediul internetului. Aceste computere virtuale simulează majoritatea atributelor unui computer real. Unul dintre avantajele principale în folosirea cloud computingului este posibilitatea de a înlocui cheltuielile necesare procurării de servere și de alte dispozitive necesare unei infrastructuri IT, cu costuri mult mai reduse. Serviciul de cloud poate înlocui ușor cateva sute sau mii de servere, având rezultate rapide la prețuri reduse. Costurile se adaptează în funcție de volumul de muncă, numărul de servere folosite precum și timpul de utilizare. Pe lângă costurile reduce, AWS oferă agilitate și elasticitate instantanee, având o infrastructură care permite implementarea și experimentarea de noi aplicații în orice moment, fără a avea probleme pe partea de hardware. [13]

Există trei modele principale pentru cloud computing, fiecare model reprezintă o parte diferită a stivei de cloud computing: infrastructura ca serviciu (acronimul specific: IaaS), platforma ca serviciu (acronimul specific: PaaS), software ca serviciu (acronimul specific: SaaS).

Infrastructura ca serviciu conține elementele de bază pentru a lucra cu cloud-ul și oferă în general acces la funcții de rețea, computere și spațiu de stocare a datelor. Aceasta oferă utilizatorilor un nivel de flexibilitate înalt și controlul gestionării resurselor IT și este cel mai asemănător cu resursele IT deja existente. Câteva din utilizările acestui tip de serviciu sunt: testarea și dezvoltarea de noi aplicații, stocarea și recuperarea datelor, infrastructură pentru aplicații web, migrarea și gestionarea aplicațiilor.

Platformele ca serviciu elimină necesitatea gestionării infrastructurii subiacente (de obicei hardware și sisteme de operare) și permit ca dezvoltatorii să își concentreze atenția asupra dezvoltării aplicațiilor. Serviciul duce la scăderea timpului de dezvoltare pentru noi aplicații și implicit crește eficiența dezvoltatorilor deoarece elimină grijile cu privire la achiziția de resurse, planificarea capacității și întreținerea software-ului.

Software-ul ca serviciu oferă un produs complet rulat și administrat de furnizorul de servicii cloud. Un exemplu cunoscut de aplicație de tip SaaS este poșta electronică, unde se pot trimite și primi e-mailuri fără a fi nevoie să se gestioneze sau să se modifice caracteristici la produsul finit de e-mail nici să se mențină serverele și sistemele de operare pe care rulează acesta. Serviciul oferă utilizatorilor oportunitatea de a accesa aplicații sofisticate deja implementate, contactarea imediată și facilă a echipei de lucru și utilizarea aplicațiilor de oriunde prin intermediul internetului.

Fig 3.1

În figura 3.1 pot fi observate într-o reprezentare grafică diferenențele dintre cele trei tipuri de servicii oferite de furnizoorii de cloud computing. Software-ul ca serviciu este cel mai cuprinzător dintre ele deoarece oferă dezvoltatorilor instrumente software predefininte, fără ca aceștia să se îngrijoreze de platformele și ifrastructurile pe care acestea sunt găzduite.

3.4 Web Scrapers

Un crawler web este un bot de internet ce navighează în mod sistematic pe internet în scopul indexării web. Majoritatea motoarele de căutare web și diferite site-uri utilizează software-uri de accesare cu crawlere pentru a-și reactualiza frecvent conținutul lor sau pentru a actualiza indicii de conținut web al altor site-uri. Există metode prin care un site web ce nu dorește sa fie accesat de crawlere poate face cunoscut acest lucru agentului de crawling. De exemplu, includerea unui fișier numit ”robots.txt” poate solicita boților să indexeze doar părți ale unui site web sau nimic.

Crawler-ul începe cu o listă de adrese URL, face cereri către server și extrage informațiile relevante folosind diferite limbaje pentru a analiza răspunsul primit. Limbajele utilizate pentru analiza răspunsului sunt: XPath și CSS. Pe măsură ce crawlerul vizitează aceste adrese URL, identifică hyperlinkurile din pagini și le adaugă la lista URL-urilor de vizitat. Procesul de analiză al răspunsului se repetă pentru fiecare dintre acestea.

Limbajul utilizat în acest proiect pentru analiza răspunsului a fost XPath. Acesta oferă posibilitatea de a accesa într-o manieră flexibilă diferite părți ale unui document HTML. XPath folosește notații de cale pentru a naviga prin structura ierarhică a unui document.

Scraperele web folosesc aceeași tehnologie ca și crawlerele dar au scopul de a descărca informații specifice de pe site-urile pe care le vizitează. Folosind limbaje de analiză a răspunsului acestea identifică informații specifice care vor fi stocate în baze de date sau în fișiere de tip CSV pentru a fi folosite ulterior. Scraperele web pot fi fie implementate de către utilizatori sau software-uri predefinite (extensii pentru browsere). Pentru a creea unul, sunt necesare cunoștințe de programare, dar avantajul pe care îl oferă această abordare este dat de nivelul mare de personalizare al funcțiilor scraperului și al datelor extrase, spre deosebire de extensiile pentru

browsere ce sunt ușor de folosit, dar care au funcționalități limitate și pot descărca doar anumite informații.

Scraperele web pot rula pe computerul personal, local, sau în cloud. Atunci când sunt rulate local ele folosesc conexiunea la internet și resursele locale, ceea ce înseamnă o utilizare ridicată a procesorului și a memoriei RAM. De asemenea, pentru site-uri web mari, scraping-ul poate dura ore, sau chiar zile. Dar scraperele ce rulează în cloud, oferă posibilitatea da a folosi resursele unui server extern pentru a colecta datele, resursele computerului personal fiind eliberate. Astfel utilizatorul poate lucra și la alte sarcini, fiind notificat ulterior odată ce scraperul poate exporta datele. [14]

Câteva dintre aplicațiile în care sunt folosite tehnici de web scraping sunt:

- Analiza sentimentelor cu privire la diferite subiecte în postările de pe rețelele sociale
- Monitorizarea prețurilor de pe diferite magazine online
- În învățarea automată, unde este necesară aducerea de date noi într-un mod constant pentru îmbunătățirea modelului

Majoritatea instrumentelor de web scraping oferă o multitudine de formate în care datele descărcate pot fi stocate, cele mai populare fiind: fișiere CSV, Excel sau baze de date SQL.

4. Implementare

Implementarea aplicației a avut două etape importante: implementarea modelului de învățare automată și implementarea aplicației web.

4.1 Implementarea modelului

4.1.1 Baza de date

Baza de date folosită pentru a antrena modelul de învățare automată este o bază de date MySQL și a fost obținută prin colectarea de informații folosind scrapere web de pe diferite site-uri cunoscute cu anunțuri de imobiliare din București. Câteva dintre site-urile web folosite pentru extragerea datelor sunt: anuntul.ro, imobiliare.ro, homezz.ro și olx.ro. În baza de date sunt peste 200.000 de intrări, cu anunțuri pentru închiriere începând din Septembrie 2019 până în Ianuarie 2021. Informațiile disponibile în baza de date, pentru fiecare apartament sunt: prețul chiriei, locația, numărul de camere, numărul de băi, dacă este mobilat, suprafața construită, suprafața utilă, etajul, numărul total de etaje al clădirii, anul de construcție al clădirii, dacă este mobilat, numărul de balcoane, numărul de balcoane închise, confort și tipul apartamentului. Pentru fiecare locație/adresă a unui apartament a fost indexată, într-un tabel separat, zona din București de care aparține.

Structura bazei de date:

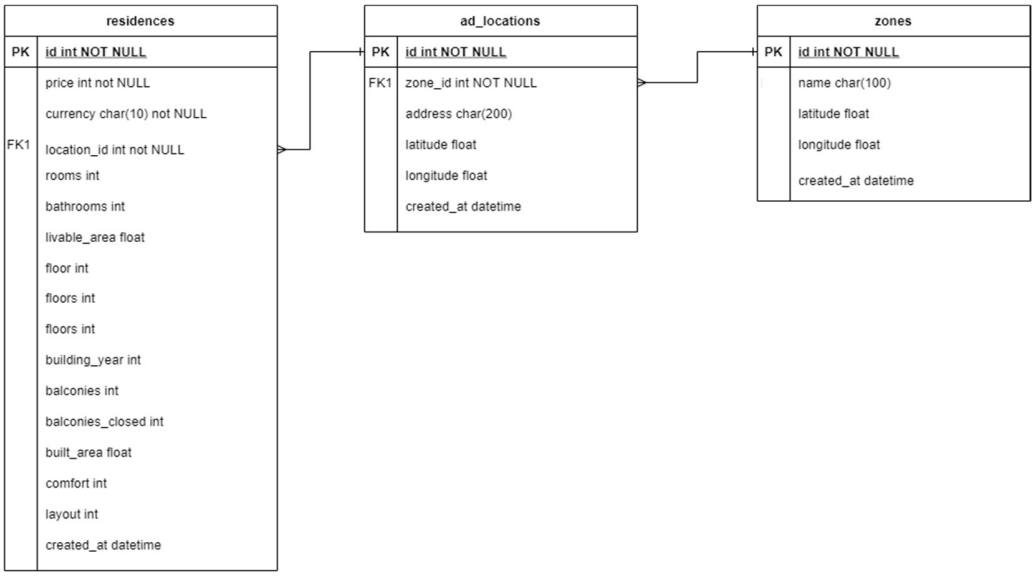


Fig 4.1

În figura 4.1 este prezentată structura bazei de date. Există trei tabele: „residences”, „ad_locations” și „zones”. În tabelul „residences” se regăsesc detaliile despre apartament precizate anterior și tot aici se găsește si o cheie externa numită „location_id” ce face legătura între tabelul „residences” și „ad_locations”. În „ad_locations” au fost inserate detaliile despre locația apartamentului din anunț, precum si un câmp „zone_id” ce reprezintă o cheie externă către tabelul „zones” în care se află o listă a tuturor zonelor disponibile pentru locuit în București.

Legătura între baza de date și aplicația propriu-zisă a fost implementată prin intermediul librăriei SQLAlchemy din Python. SQLAlchemy oferă dezvoltatorilor flexibilitate și putere completă asupra bazelor de date SQL prin modelele concepute pentru a fi adaptate în stilul de programare specific pentru Python. Astfel, a fost folosită o clasă de bază în care se creează conexiunea cu baza de date, aceasta are rolul de a fi extinsă în cele trei tabele (residences, ad_locations și zones), fiecare tabel având propriile detalii specifice și metode de extragere a datelor.

Pentru a fi accesibilă de oriunde, baza de date a fost mutată de pe serverul local de MySQL în Amazon Relational Database Service, numit și Amazon RDS. Acest serviciu oferă o implementare ușoară numită „Easy Create” în care utilizatorul trebuie doar să specifice tipul de bază de date pe care dorește să îl folosească (exemplu: SQL sau MySQL). Baza de date a fost migrată cu ușurință în Amazon RDS cu ajutorul unui fișier .sql în care s-au regăsit instrucțiunile SQL pentru crearea de tabele și inserarea datelor.

4.1.2 Data cleaning

Data cleaning (în traducere „curățarea datelor”) este procesul de pregătire a datelor pentru antrenare prin eliminarea sau modificarea datelor incorecte, incomplete, inexacte, irelevante, duplicate sau formatate necorespunzător. Acesta este un proces critical în procesarea datelor deoarece sporește consistența datelor, încredera în date și valoarea acestora. Inexactitățile obișnuite în date include valori lipsă sau intrări greșite și erori de tipografie. În unele cazuri pentru curățarea datelor este necesarp completarea sau corectarea datelor, în timp ce în alte cazuri, valorile trebuie eliminate.

În învățarea automată datele eronate pot avea un efect negativ semnifcativ asupra deciziilor luate de modelul antrenat. Cu cât cantitățile de date cresc, cu atât este mai necesară curățarea datelor.

Principalele avantaje al procesului de data cleaning sunt:

- Îmbunătățește modul în care sunt luate deciziile
- Crește eficiența, fiind eliminate datele eronate, dezvoltatorii au de a face doar cu datele corecte
- Crește competitivitatea, o companie ce se respectă nu va livra niciodată date eronate

Prin procesarea setului de date s-au remarcat valori ce ieșeau din domeniul de valori dorit și existența unor caracteristici imposibile a apartamentelor (exemplu: suprafața locuibila a unui apartament fiind trecută pe site ca fiind 9,999 mp2 sau prețul chiriei lunare de 2.100.000 euro pe lună). Astfel s-au stabilit câteva limite arbitrare pentru a elimina aceste intrări:

Câmp	Tip limită	Valoare admisă
Preț	Minim	75 euro
	Maxim	1500 euro
Suprafața utilă	Minim	9 mp2
	Maxim	260 mp2
Număr de camere	Minim	1 cameră
	Maxim	8 camere
Prețul pe metrul pătrat (calculat ca preț/suprafața locuibilă)	Minim	2 euro/mp2
	Maxim	19 euro/mp2

Numărul de intrări ce s-au încadrat în aceste limite a fost de 152.768, fiind eliminate aproximativ 50.000 din cele 200.000 de intrări inițiale.

Moneda folosită pentru evaluarea prețului chiriei a fost Euro, astfel pentru prețurile listate în Ron, s-a realizat o conversie în Euro, folosind rata de conversie de 1 Euro la 4.8 Ron, aceasta fiind media ratei din perioada Septembrie 2019 - Ianuarie 2021.

Pentru înțelegerea setului de date au fost create următoarele grafice ce reprezintă caracteristicile importante ale datelor:

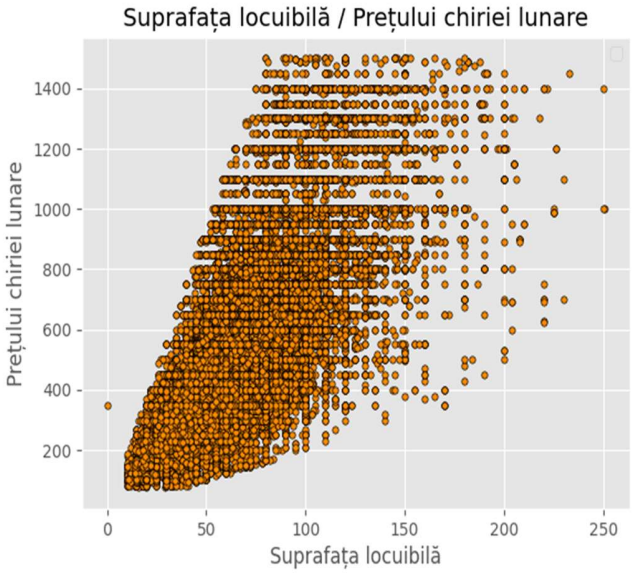


Fig. 4.2

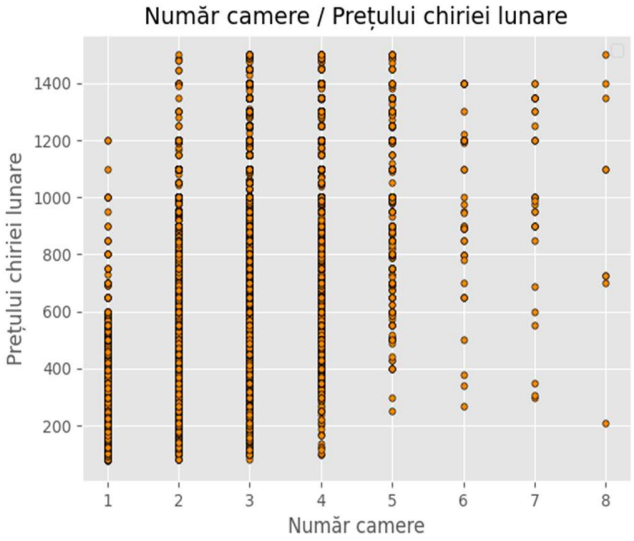


Fig 4.3

Din cele două grafice(figura 4.2 și figura 4.3) se poate observa că prețul chiriei crește odată cu creșterea suprafeței locuibile în schimb ce numărul de camere al apartamentului nu este atât de relevant în evaluarea acestuia. Suprafața locuibilă este unul dintre cei mai importanți parametrii de care se ține cont în stabilirea prețului unui apartament, de multe ori aceasta face ca alți parametrii ca etajul, numărul de camere, numărul de băi să devină neglijabili. Un alt lucru ce poate fi observat pe baza graficelor din figurile de mai sus este că odată cu creșterea suprafeței locuibile sau a numărului de camere din apartament, scade numărul de apartamente disponibile pentru închiriere, lucru ce ne sugerează că prețul va fi mai ridicat și din cauza ofertei reduse.

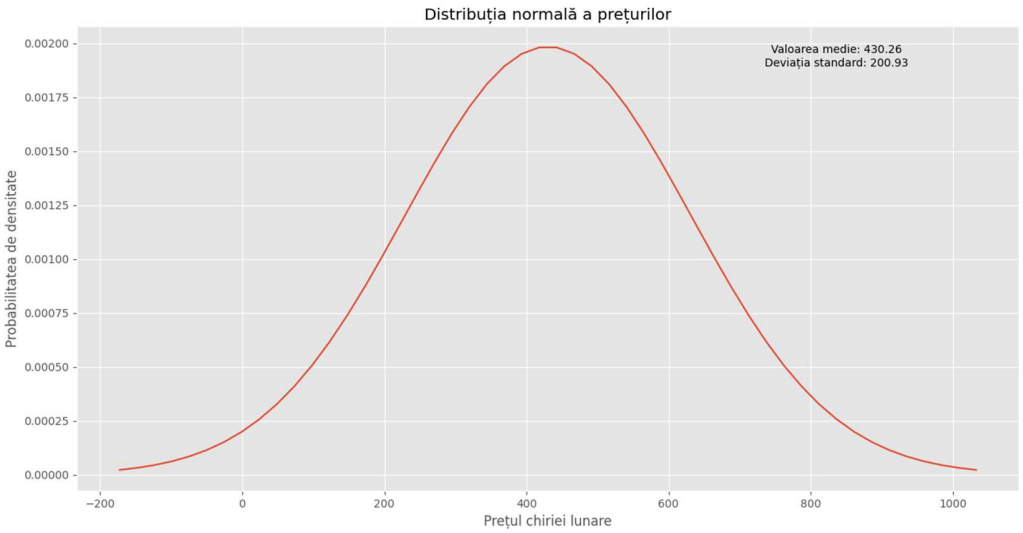


Fig 4.4

În figura 4.4 a fost realizată distribuția prețurilor în baza de date. Se observă că prețurile chiriei lunare au o distribuție normală cu valoarea medie de 430 euro și deviația standard de aproximativ 200. Aproximativ 68% din numărul total de intrări sunt cuprinse între 230 și 630 de euro și aproximativ 95% din intrări sunt cuprinse între 30 și 830 de euro. Pentru ca informațiile prezise de modelul de învățare automată să fie relevante, a fost ales arbitrar un interval de 50 de euro în care să fie încadrate apartamentele în funcție de caracteristicile acestora, fiind una din cele mai caracteristice sume atunci când există fluctuații în piața chiriilor imobiliarelor.

4.1.3 Modelul

Algoritmul de pădure aleatoare a fost implementat cu ajutorul librăriei ”sklearn” din Python. „Sklearn” sau ”Scikit-learn” este o librărie gratuită de machine learning ce conține numeroși algoritmi de clasificare și regresie. Modelul primește pentru antrenare următorii hiper parametrii: numărul de arbori (numiți și estimatori), numărul minim de eșantioane, numărul minim de noduri frunză, adâncimea maximă a arborilor, modul de împărțire al caracteristicilor etc. Acești parametri au fost aleși inițial într-o manieră aleatoare, optimizarea acestora se va face ulterior. Pentru a scurta durata de rulare pentru antrenarea modelului se setează ca hiper parametru n_jobs = -1, acesta este utilizat pentru a specifica câte procese sau fire simultane ar trebui utilizate pentru rutinele care sunt paralelizate. N_jobs este un număr întreg, care specifică

numărul maxim de procesoare care rulează simultan. Dacă valoarea sa este egală cu 1, nu se utilizează deloc paralelismul, ceea ce este util pentru depanare. Dacă valoarea este setată la -1, sunt utilizate toate procesoarele. Pentru n_jobs sub -1, sunt utilizate un număr de procesoare egal cu suma dintre numărul de procesoare disponibile, parametrul n_jobs și o unitate (n_cpus + 1 + n_jobs). De exemplu, cu n_jobs = -2, sunt utilizate toate procesoarele, cu excepția unuia.

Alți hiperparametri ce pot fi folosiți sunt:

- **n_estimators** - acesta reprezintă numărul de arbori din pădure
- **min_samples_split** – un număr minim de caracteristici necesare pentru împărțirea unui nod intern. Acesta poate fi: fie un număr întreg, atunci fiind considerat ca numărul minim; fie un număr real cu virgulă mobilă, și este considerat o ca o fracțiune din numărul total de caracteristici
- **min_samples_leaf** - un număr minim de caracteristici ce pentru a fi un nod frunză. Un nod divizat la orice adâncime din arbore va fi luat în considerare numai dacă lasă cel puțin min_samples_leaf probe de antrenament în fiecare dintre ramurile stânga și dreapta. Acest lucru poate avea ca efect netezirea modelului, în special în regresie. La fel ca min_samples_split, acesta poate fi un număr întreg, fiind numărul minim de caracteristici, sau un număr real cu virgulă mobilă, fiind o fracțiune din numărul total de caracteristici
- **max_features** - reprezintă numărul de atribute ce vor fi luate în considerare atunci când este căutată cea mai bună divizare. Acesta poate fi: un întreg ce reprezintă numărul maxim de caracteristici, un număr real în virgulă mobilă reprezentând o fracțiune din numărul total. Dacă max_features ia valoarea „auto”, atunci numărul maxim este numărul total de caracteristici, pentru valoarea „sqrt” max_features este egal cu radical din numărul total de caracteristici, pentru valoarea „log2” numărul maxim este egal cu logaritm în baza 2 din numărul total de caracteristici
- **criterion** - reprezintă funcția evaluare a calității unei împărțiri. Acesta poate lua valoarea „mse” și reprezintă eroarea pătratică medie sau „mae” ce reprezintă eroarea absolută medie
- **bootstrap** - poate avea valoarea „True” sau „False”. Dacă este setat ca „True” atunci sunt folosite metode de împărțire a datelor pentru construirea arborilor. Dacă este setat ca „False”, atunci întregul set de date este utilizat pentru a construi fiecare copac. [15]

Modul de delcarare al modelului:

```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators = 300,
                                 min_samples_split = 2,
                                 min_samples_leaf = 1,
                                 max_features = 'sqrt',
                                 max_depth = 50,
                                 criterion = 'mse',
                                 bootstrap = False,
                                 n_jobs = -1)
```

Algoritmul de pădure aleatoare este o colecție de arbori, fiecare fiind creeat independent folosind date de formare de intrare etichetate și complete. Prin complet, înțelegem în mod explicit că nu lipsesc valori, adică valori NULL sau NaN. Aceasta problemă poate fi gestionată în două moduri: renunțarea la datele cu valori lipsă sau completarea valorilor lipsă cu o valoare aleasă ca fiind media datelor de intrare. În acest caz, s-a ales cea de-a doua variantă unde datele de intrare lipsă au fost verificate și completate cu valoarea medie a datelor de intrare.

Pentru a evalua modelul, din numărul total de intrări din baza de date, vor fi folosite 90% pentru antrenarea modelului, și 10% pentru testarea ulterioară a acestuia. Astfel, după antrenare, media erorii prețului pe setul de antrenare a fost de 77 euro și dintr-un total de 15277 de intrări de test, doar pentru 3776 a fost estimat intervalul corect de preț, rezultând într-o acuratețe de 24.71%.

Pentru antrenarea modelului se folosește funcția „`rf.fit(X, y)`” unde „rf” reprezintă modelul definit mai sus, „X” reprezintă totalitatea datelor folosite pentru antrenare și „y” reprezintă etichetele datelor de antrenare. Folosind metoda „`rf.predict(X)`” se vor face predicții folosind modelul antrenat pe un eșantion de date ce nu a fost folosit în antrenare.

Folosind modelul antrenat mai sus poate fi reprezentată grafic importanța fiecărei caracteristici utilizată în luarea deciziilor de către arborii din pădure. Aceste date pot fi găsite în atributul „feature_importances_” din modelul antrenat și vor fi afișate pe un grafic.

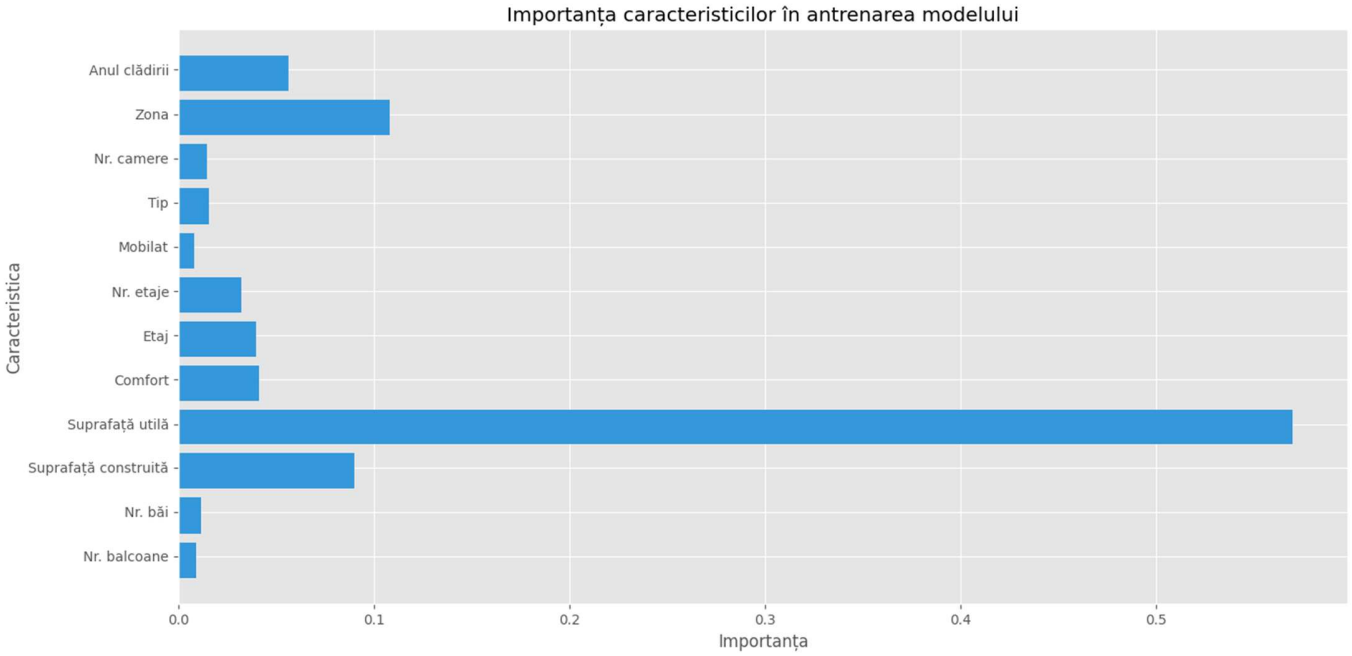


Fig 4.5

În figura 4.5 pot fi observate caracteristicile cele mai importante în evaluarea unui apartament folosind metoda descrisă mai sus. Se poate observa că cea mai importantă caracteristică este suprafața utilă/locuibilă. Acest lucru poate fi înțeles deoarece de acest parametru depind și alți parametri precum: numărul de camere, numărul de băi și suprafața construită. A doua cea mai importantă caracteristică este reprezentată de zona în care apartametnul este localizat. Parametrii cu cea mai mică importanță în luarea de decizie a modelului sunt: dacă apartamentul este mobilat sau nu, numărul de băi și numărul de balcoane.

4.1.4 Optimizarea modelului

Pentru optimizarea hiper parametrilor modelului a fost folosit ”RandomizedSearchCV”, din librăria sklearn. „Căutarea aleatorie este o tehnică în care sunt utilizate combinații aleatorii ale hiper parametrilor pentru a găsi cea mai bună soluție pentru modelul construit. Acesta implementează metode de ”antrenare” și ”scor” pentru a căuta într-o manieră aleatoare hiper parametrii cei mai potriviți pentru un model. Deoarece selectarea parametrilor este complet aleatorie, și întrucât nu se folosește nicio inteligență pentru a testa aceste combinații, implică un timp ridicat de calcul pentru a acoperi fiecare combinație din spațiul de lucru. Acest algoritm de

optimizare funcționează cel mai bine în ipoteza că nu toți hiper parametrii sunt la fel de importanți. În acest model de căutare, combinații aleatorii de parametri sunt luate în considerare în fiecare iterație.” [16]

RandomizedSearchCV a fost rulat de mai multe ori, făcând câte 100 de iterații la fiecare rulare și astfel au fost obținute mai multe combinații de parametri:

Configurație	Parametri	Media absolută a erorii	Numărul de intervale estimate corect
1	n estimators = 1400	62	6155/18284
	min samples leaf = 1		
	min samples split = 2		
	max depth = 100		
	max features = 'auto'		
	bootstrap = True		
2	n estimators = 600	63	5983/18284
	min samples leaf = 1		
	min samples split = 5		
	max depth = 80		
	max features = 'auto'		
	bootstrap = True		
3	n estimators = 1400	62.5	6001/18284
	min samples leaf = 2		
	min samples split = 2		

	max depth = 70		
	max features = 'auto'		
	bootstrap = True		

Astfel configurația aleasă pentru modelul final a fost Configurația 1, aceasta având o acuratețe de 33.66%. Pentru salvarea și folosirea ulterioară a modelul a fost folosită librăria ”joblib” din Python ce furnizează instrumente pentru serializarea modelelor de învățare automată antrenate.

4.2 Implementarea aplicației web

4.2.1 Flask

Flask este un framework web popular dezvoltat în cadrul limbajului Python utilizat pentru dezvoltarea cu ușurință a aplicațiilor web. Un framework web reprezintă o colecție de biblioteci și module ce permit dezvoltatorilor de aplicații web să scrie aplicații fără să se îngrijoreze de detalii de nivel scăzut, cum ar fi protocoale, gestionarea firelor și așa mai departe. Flask este numit de multe ori microframework deoarece este conceput pentru a menține nucleul aplicației simplu și scalabil. Câteva dintre avantajele utilizării framework-ului Flask sunt:

- Compatibilitate mare cu cele mai noi tehnologii
- Ușor de utilizat pentru cazuri uzuale și experimente tehnice
- Scalabilitate ridicată pentru aplicații simple
- Rutarea URL se realizează cu ușurință
- Aplicații ușor de dezvoltat și întreținut
- Baza de date se integrează ușor
- Este o platformă minimalistă, dar puternică

Prin convenție, șabloanele și fișierele statice sunt stocate în subdirectoare în arborele sursă Python al aplicației, acestea fiind numite „templates” și „static”. Deși acest lucru poate fi schimbat, de obicei nu este necesar, mai ales atunci când începeți.

Flask introduce un template engine numit Jinja2 folosit în aplicație pe partea de frontend. Template engine sunt similare cu limbajele de programare și fiecare dintre acestea, are o anumită înțelegere despre modul în care funcționează lucrurile. Jinja2, de exemplu, are un sistem extins de filtrare, un anumit mod de a face moștenirea șablonului, suport pentru blocuri reutilizabile (macro-uri) care pot fi utilizate din interiorul șabloanelor și, de asemenea, din codul Python. Jinja2 acceptă redarea șablonului iterativă, are sintaxa configurabilă și multe altele.

Jinja folosește un obiect central numit șablon environment. Instanțele din această clasă sunt utilizate pentru a stoca configurația și obiectele globale și sunt utilizate pentru a încărca șabloane din sistemul de fișiere sau din alte locații. Flask creează automat un obiect de tip environment la inițializarea aplicației care va fi folosit pentru încărcarea șabloanelor.

Un șablon Jinja este pur și simplu un fișier text. Jinja poate genera orice format bazat pe text (HTML, XML, CSV, LaTeX etc.), acesta nu trebuie sa aibă o extensie specifică cum ar fi .html sau .xml. Un șablon conține variabile și expresii, care sunt înlocuite cu valori atunci când un șablon este redat;precum și etichete, care controlează logica șablonului. Sintaxa șablonului este puternic inspirată de Django și Python.

Delimitatorii Jinja impliciți sunt configurați după cum urmează:

- {% ...%} pentru declarații
- {{...}} pentru expresii ce urmează să fie tipărite în rezultatul șablonului
- {# ... #} pentru comentarii [17]

Flask suportă toate metodele de cereri HTTP. HTTP funcționează ca un protocol de cerere-răspuns între client și server și definește un set de metode de solicitare de la server utiliazte pentru indicarea acțiunii dorite ce trebuie efectuată pentru o resursă dată. Metodele cele mai cunoscute pentru a realiza cererile HTTP sunt:

- Metoda GET – utilizată pentru a solicita o simplă reprezentare a unei resurse specificate. Cererile care utilizează metoda GET trebuie doar să recupereze date.
- Metoda POST - folosită pentru a transmite o entitate la o resursă specificată, cauzând în general o schimbare a stării sau diferite efecte pe server.

- Metoda PUT – utilizată pentru a înlocui sau actualiza reprezentările actuale ale unei resursă țintă
- Metoda DELETE – utilizată pentru a șterge diferite resurse. [18]

Prin intermediul framework-ului Flask s-a implementat o nouă aplicație web ce rulează pe portul 8080, ce are rolul de a prezice intervalul de preț al chiriei lunare pentru apartamente din București folosind detalii despre imobilul respectiv, introduse manual de utilizator sau prin introducerea unui URL de pe un site de imobiliare. De asemenea, a fost implementat și un API ”/prediction-api” ce poate fi apelat din exterior cu metoda POST cu o listă ce conține detaliile unui apartament și care întoarce intervalul de preț în care acesta ar trebui să se încadreze. Detaliile trebuie ce vor fi folosite pentru prezicere trebuie să fie introduse într-o structură de date de tip JSON. Prezicerile de preț sunt realizate cu modelul antrenat anterior. Aplicația web a fost implementată pe ruta ”/predict” și dacă se încearcă accesarea oricărei alte rute neimplementate, aplicația redirecționează automat utilizatorul către ”/predict”. Prin metoda de cerere GET, aplicația întoarce o pagină statică redată prin intermediul Jinja. Detaliile introduse de utilizator pe pagina afișată, sunt trimise prin metoda POST către server, acesta urmând să descarce caracteristicile imobilului folosind un web scraper pentru un URL dat, dacă acest lucru este necesar, urmând să ofere o predicție asupra intervalului de preț printr-o notificare de tip alert.

4.2.2 Scrapy

Pentru dezvoltarea scraper-ului web ce descarcă detaliile despre apartamente, a fost folosit framework-ul Scrapy. Scrapy este un framework pentru dezvoltarea de crawlere web și de web scraping, utilizat pentru extragerea de date structurate din pagini web. Poate fi folosit într-o varietate mare de scopuri, de la extragerea datelor la monitorizare și testare automată.

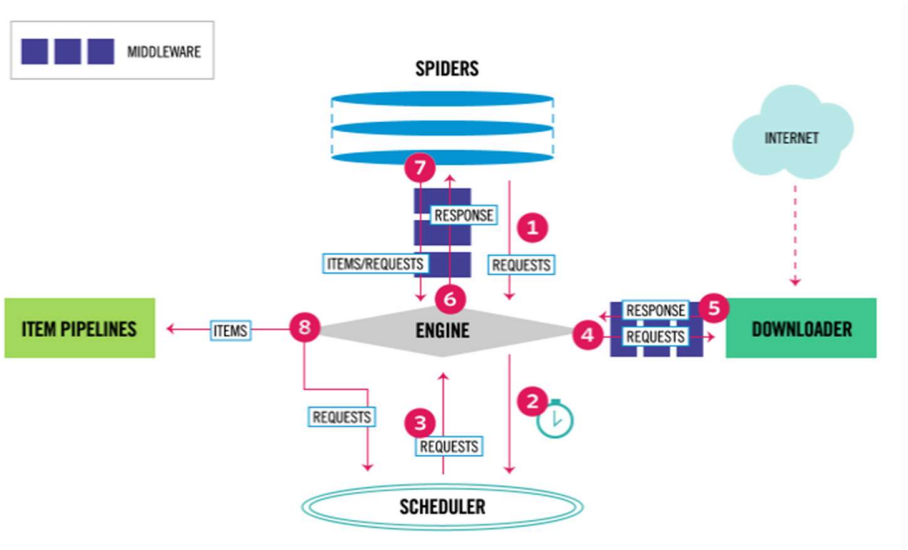


Fig 4.6

Diagrama din figura 4.6 prezintă o prezentare generală a arhitecturii Scrapy cu componentele sale și o schemă internă a fluxului de date. Acest flux de date din Scrapy este controlat de componenta motor de execuție astfel: motorul primește cererile inițiale pe care trebuie să le execute, după aceasta motorul planifică cererile primite prin intermediul unui componente numite Scheduler. Scheduler-ul returnează următoarele cereri către motor, acestea sunt trimise către o instanță numită Downloader. Odată ce pagina a fost descărcată, se generează un răspuns care este trimis la motor, acesta îl trimite mai departe către Spider, unde este procesat prin intermediul componentei Item Pipeline și sunt returnate articolele de interes. Procesul se repetă până când nu mai există cereri din partea Scheduler-ului. [19]

Spiderii sunt clase personalizate scrise de utilizatorii Scrapy pentru a analiza răspunsurile folosind XPath/CSS și de a prelua elemente din ele sau cereri suplimentare de urmat. S-au implementat două Spidere ce au rolul de a extrage informațiile relevante de pe paginile web de imobiliare din România: www.homezz.ro și www.imobiliare.ro. Aceste date extrase sunt preluate de la spidere și trimise către aplicația web cu ajutorul unei instanțe numită CrawlerRunner. CrawlerRunner este o clasă de ajutor convenabilă care ține evidența, gestionează și rulează crawlerele într-un reactor deja configurat de către Scrapy.

Informațiile extrase despre apartamente cu ajutorul celor doi spideri sunt:

- „balconies” – ce reprezintă numărul de balcoane
- „balconies_closed” – este numărul de balcoane închise
- „bathrooms” – numărul de băi
- „built_area” – suprafața construită
- „livable_area” – suprafața utilă/construită
- „comfort” – nivelul de comfort
- „floor” – etajul la care este localizat apartamentul
- „floors” – numărul total de etaje al clădirii
- „layout” – tipul apartamentului (exemplu: decomandat, semidecomandat, nedecomandat)
- „rooms” – numărul de camere
- „zone” – zona în care este localizat apartamentul
- „zone_id” – va fi extras din baza de date pe baza zonei descrise anterior
- „building_year” – anul în care a fost construită clădirea
- „price” – prețul apartamentului
- „currency” – moneda folosită pentru preț (există suport doar pentru EURO sau RON)

După descărcarea acestor informații, ele sunt puse la dispoziția instanței de CrawlerRunner prin intermediul unui pipeline numit „ItemCollectorPipeline” ce are rolul de a salva în memorie articolul creeat de către spdier. Aceste date sunt apoi curățate folosindu-se aceleași reguli descrise în secțiunea 4.1.2 Data cleaning ca mai apoi să fie folosite în obținerea unei predicții de preț.

Instanțele de CrawlerRunner ce rulează în reactorul configurat de Scrapy pentru descărcarea acestor informații sunt închise imediat după ce articolul ajunge la serverer-ul de Flask. În cazul în care descărcarea nu este reușită sau se încearcă descărcarea de detalii de pe un site pentru care nu a fost implementat un spider, CrawlerRunner întoarce un mesaj de eroare.

4.2.3 Interfața grafică

Pentru interfața grafică a aplicației web au fost folosite tehnologiile HTML, CSS și Bootstrap, precum și librăria „sweetalert2” din JavaScript.

„HTML, acronimul în limba engleză pentru „Hyper Text Markup Language”, este un limbaj folosit în descrierea structurii unei pagini web folosind etichete, elemente și atribute. Etichetele sunt reprezentate de cuvinte cheie aflate între caracterele „<” și „>”, marea majoritate a etichetelor sunt prezente în perechi, cu o etichetă de început și una de încheiere. Ambele conțin același cuvânt cheie plasat între caracterele „<” și „>”, diferența dintre acestea fiind dată de fapul că în cazul etichetei de încheiere, cuvântul este precedat de caracterul „ / ”. Un exemplu de pereche este „`<span>`” și „`</span>`” folosită pentru definirea unei secțiuni într-un document. Există însă și etichete ce nu prezintă o pereche. De exemplu, pentru a defini o imagine este suficientă folosirea etichetei „`<img>`”.” [20] Atributele sunt specificate în eticheta de început a unei perechi și acestea au rolul da oferi informații suplimentare despre elementul HTML. Un exemplu de atribut este calea către o imagine folosit în elementul „`<img>`”, sub forma de cheie valoare „src = cale”, elementul final având următoarea formă „`<img src = cale>`”.

CSS, acronimul în limba engleză pentru „Cascading Style Sheet”, este un limbaj folosit pentru stilizarea paginilor web, prin modificarea: fontului, culorilor, aspectului așezării în pagină. Altfel spus, CSS descrie cum vor fi redate elementele HTML ale unei pagini web. Adaptarea la diferite tipuri de dispozitive, precum: ecrane mari, ecrane mici sau dispozitive mobile; se realizează cu ușurință folosind CSS. Pentru ca stilizarea precizată într-un fișier .css să fie prezentă într-o pagina web este necesară includerea acestuia într-un element „`<link>`” din fișierul .html în care se va preciza în atributul „href” calea către fișierul .css dorit.

Bootstrap este un framework de CSS gratuit folosit în dezvoltarea interfețelor grafice a aplicațiilor web. Acesta oferă multe facilități, ce ușurează munca dezvoltatorilor, precum:
șabloane stilizate pentru butoane, formulare și tabele. Pentru a beneficia de aceste facilități oferite de Bootstrap este necesară includerea fișierelor specifice Bootstrap în fișierul .html. Există și varianta includerii directe într-un element „`<link>`”, deoarece fișierele se află pe server-ele din rețeaua Bootstrap.

Librăria „sweetalert2” oferă o metodă de a înlocui alertele clasice din JavaScript cu notificări personalizate, accesibile și stilizate într-un stil modern. Aceasta poate fi inclusă în fișierul .html prin intermediul unui element „`<script>`” ce primește ca atribut „src” calea către

fișierele din rețeaua „sweetalert2”. Pentru a crea aceste alerte personalizate, dezvoltatorul le introduce în interiorul unui element „`<script>`” și declanșează notificarea folosind sintaxa „`Swal.fire()`”. Pot fi specificați multipli parametri pentru această funcție, câteva exemple fiind: „title” ce reprezintă titlul din interiorul alertei, „text” ce reprezintă textul din interiorul acesteia și „confirmButtonText” pentru mesajul de pe butonul ce închide notificarea.

Folosind aceste tehnologii a fost dezvoltată o interfață grafică ce prezintă inițial o pagină web în care utilizatorul are opțiunea de a insera un URL sau se poate face o introducere manuală a detaliilor apartamentului într-un formular stilizat. După ce server-ul întoarce o predicție, aceasta este afișată într-o notificare de tip „sweetalert2” sub forma unui mesaj. Dacă utilizatorul dorește, poate accesa secțiunea „Vezi mai multe detalii”, unde poate vedea: prețul estimat pentru chirie, prețul actual de pe site-ul de imobiliare, detaliile apartamentului și două grafice reprezentative pentru modelul de învățare automată.

În dezvoltarea interfeței grafice s-a urmărit realizarea unei interfețe ușor de înțeles pentru utilizatorii obișnuiți, prezentată într-o maniera interactivă și atractivă. Astfel au fost folosite imagini și culori aprinse ce atrag ochii vizitatorilor. Imagini reprezentative pentru interfața grafică pot fi vizualizate în Anexa 1.

4.2.4 Găzduirea aplicației într-o instanță de Amazon EC2

Ultimul pas în dezvoltarea aplicației a fost găzduirea și rularea acesteia într-o instanță de Amazon EC2. Principalele avantaje pentru a găzdui aplicația în cloud sunt: aplicația devine disponibilă de oriunde, nu mai este necesară rularea constantă pe computerul personal, acesta având resurse alocate și ușor de scalat după nevoi.

Amazon EC2 reprezintă un serviciu oferit de Amazon Web Services ce oferă capacitate de calcul în cloud ce poate fi redimensionată cu ușurință. Oferă infrastructură ca serviciu (IaaS). Acesta oferă utilizatorilor controlul complet asupra resurselor de calcul, ce pot fi scalate conform cerințelor. Un server virtual utilizat pentru rularea aplicațiilor în Amazon EC2 se numește instanță. Instanța este o parte mică a unui computer mare, ce are propriul hard disk, conexiune la rețea și sistem de operare. EC2 folosește AWS Management Console, AWS CLI sau AWS SDKs pentru gestionarea scalării în funcție de nevoile în schimbare și simplifică implementarea serverelor virtuale și întreținerea stocării.

Există o gamă largă de tipuri de instanțe de EC2 disponibile in Amazon Web Services. Scopul acestei diverisificări de tipuri este de oferi fiecărui utilizator exact ce are nevoie, instanțele având caracteristici specifice (exemple: tip de procesor, număr CPU-uri, memorie RAM). Pentru a găsi tipul de instanță cu specificațiile de care un utilizator are nevoie, acesta trebuie sa consulte documentația oficiala oferită de AWS unde poate găsi toate caracteristicile fiecărui tip de instanță. Scalarea aplicațiilor de pe aceste instanțe se face cu ușurință, din interfața grafica a serviciului Amazon EC2, și oferă posibilitatea utilizatorilor de a trece la o configurație mai puternică sau mai slabă, după nevoie, păstrand fișierele de pe hard disk.

În cazul de față a fost folosită o instanță din EC2 de tipul „t2.xlarge" ce are 4 CPU-uri și 16 GB memorie RAM, această instanța a fost necesară pentru încărcarea modelului de învățare automată în memorie, modelul având o dimensiune de aproximativ 4 GB. Sistemul de operare folosit pentru această instanță este „Ubuntu Server 20.04" pe care s-au implementat setările de siguranță standard, cu excepția a 4 porturi (80, 22, 443 și 8080) ce au fost selectate ca fiind disponibile de oriunde, ip-ul sursă acceptat fiind 0.0.0.0/0.

Conectarea la această instanță s-a facut utilizând protocolul SSH (acronimul în engleză de la „Secure Socket Shell") pe portul 22 al instanței. SSH este un protocol de rețea ce oferă utilizatorilor o metodă sigură de a accesa un computer, respectiv o instanță găzduită în cloud, peste o rețea nesigură. Toate datele trimise de un computer către o rețea este criptată automat de către acest protocol, și decriptată pe mașina gazdă. Rezultatul se numește „encriptare transparentă", adică utilizatorii pot lucra normal, fără a fi conștienți că datele au fost criptate si decriptate. Protocolul are la bază un sistem de autentificare bazat pe o cheie privată și o cheie pubilcă. [21]

Pe această instanță de Amazon EC2 a fost instalat Python, pip și tmux (folosind comanda „`sudo apt install python3 python3-pip tmux`") și toate librăriile necesare pentru a rula aplicația (librării ce au fost descrise anterior in capitolul 4, Implementare) folosind pip. Aplicația a fost mai apoi instalată și rulată pe această instanță într-un serviciul „tmux", folosindu-se ca ip gazdă 0.0.0.0 si portul 8080, aplicația fiind acum disponibilă pe ip-ul public al instanței de EC2. „Tmux" oferă utilizatorilor posibilitatea de a crea și accesa mai multe terminale dintr-un singur ecran. Astfel aplicația poate fi lăsată să ruleze pe fundal în timp ce utilizatorul poate

accesa și alte funcționalități oferite de instanța de EC2. Folosind tmux pentru a rula aplicația web de Flask, se poate închide terminalul SSH cu ajutorul căruia se comunică cu instanța, iar aplicația continuă să ruleze. Pentru a închide aplicația trebuie ca utilizatorul să acceseze terminalul deschis cu tmux și sa apese simultan pe tastatură pe „CTRL + C”.

Costurile estimate pentru găzduirea și rularea aplicației de către Amazon Web Services este de aproximativ 50$ pe lună, însă în realitate acesta poate varia în funcție de mai mulți factori, precum: numărul de accesări simultane al aplicației și rata de utilizare. Amazon Web Services oferă un serviciu numit „Billing and Cost Management” în care utilizatorii își pot urmări în timp real costurile de rulare a aplicațiilor.

Concluzii

Concluzii Generale

Principalul obiectiv al lucrării de diplomă a fost de a implementa o aplicație web ce folosește tehnici de machine learning pentru estimarea corectă a prețului chiriei lunare pentru apartamentele din București. Aplicația a fost realizată pentru a oferi publicului larg o metodă de a evalua prețul chiriei fără nevoia unui specialist în domeniul imobiliar, fiind de ajutor atât pentru chiriași, cât și pentru proprietarii de apartamente.

Pentru a atinge acest scop a fost antrenat un model de învățare automată folosind algoritmul de pădure aleatoare (Random Forest Regression) ce este accesibil publicului larg prin intermediul unei aplicații web găzduită în cloud. Pentru antrenarea modelului de evaluare a prețului chiriei au fost luați în calcul mai mulți parametri specifici pentru apartamente precum: numărul de camere, suprafața construită și suprafața utilă, numărul de băi, etajul și zona.

În această lucrare au fost prezentate atât noțiunile teoretice, cât și metoda de implementare a aplicației. Prima etapă importantă în dezvoltarea acesteia a fost curățarea datelor și antrenarea modelului de învățare automată, urmată de dezvoltarea aplicației web și găzduirea acesteia în cloud.

Unul dintre aspectele observate în urma antrenării modelului de învățare automată este că acuratețea prezicerilor nu se poate apropia întotdeauna de realitate, deoarece în evaluarea prețului chiriei, un factor important de decizie este reprezentat de factorul uman emoțional. Persoanele ce dețin apartamente și doresc să le dispună chiriașilor tind să evalueze subiectiv și să supraestimeze sau subestimeze valoarea reală a imobilului deținut, în timp ce modelul realizează o medie a pieței.

Contribuții personale

Contribuțiile autorului lucrării de diplomă au fost evidențiate în Capitolul 4, Implementare. În dezvoltarea lucrării, din cauza datelor eronate, s-a început prin curățarea și prelucrarea datelor utilizate în antrenarea modelului de învățare automată. S-a realizat optimizarea modelului folosind RandomizedSearchCV pentru a se găsi cei mai buni hiper parametri și pentru a avea o acuratețe de predicție cât mai ridicată, fiind testate multiple

configurații. Următoarea etapă a fost dezvoltarea unei aplicații web interactive și ușor de utilizat de către publicul larg. În cadrul acestei etape a fost folosită librăria Flask pentru backend-ul aplicației, iar pentru redarea stilizată a aplicației au fost folosite limbajele HTML și CSS. Folosind tehnici de web scraping s-a realizat introducerea automată a detaliilor unui imobil postat pe unul dintre site-urile: imobiliare.ro sau homezz.ro. Pe baza acestor detalii, ce pot fi adăugate cu metoda descrisă anterior sau introduse manual, se va estima intervalul de preț recomandat pentru chiria lunară. Pentru ca aplicația să poată fi scalată și accesibilă de oriunde, aceasta a fost găzduită și rulată într-un serviciu cloud oferit de Amazon Web Services.

Îmbunătățiri viitoare

Aspectele prezentate mai jos pot fi îmbunătățite teoretic, atât pe partea de învățare automată, cât și pe baza aplicației web.

Pentru antrenarea modelului au fost folosite date istorice din intervalul Septembrie 2019 - Ianuarie 2021. Având în vedere că piața imobiliară este într-o continuă schimbare este esențială introducerea de date de actualitate și reantrenarea regulată a modelului astfel încât datele prezise să fie de relevante. În cadrul acestei lucrări, pentru optimizarea modelului au fost folosiți doar o parte din hiper parametrii disponibili pentru algoritmul de pădure aleatoare în scopul obținerii unor rezultate ce au o acuratețe mai ridicată este necesar ca toți acești parametri să fie luați în calcul. Acest lucru nu a fost fizic realizabil din cauza resurselor limitate, timpul de optimizare a hiper parametrilor fiind direct proporțional cu numărul de parametri utilizați, pentru un număr mare de parametri durata de optimizare ar fi mult prea ridicată pentru a se realiza pe computerul personal.

Aspectele aplicației web ce ar putea fi îmbunătățite sunt: reducerea timpului de execuție necesar prezicerii intervalului de preț pentru un apartament și introducerea unui număr mai mare de Web Scrapres care să colecteze datele despre apartamente de pe mai multe site-uri de specialitate astfel încât utilizatorii să nu fie constrânși folosirea exclusivă a secțiunii de introducere manuală a detaliilor.

Bibliografie

[1] A. Bîrluțiu, "Introducere în domeniul învățării automate," 2019.

[2] M. Cristei, G. Marin and V. Stelea, "Prezicerea performanțelor studenților folosind învățarea automată," 2017.

[3] D. M. Dobrea, "Tehnici de inteligență computațională. Aplicații în electronică și biomedicină".

[4] J. Brownlee, "Machine Learning Mastery," 2019. Disponibil: https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/. [Accesat 6 Mai 2021].

[5] J. Frankenfield, "Artificial Neural Network (ANN)," 2020. Disponibil: https://www.investopedia.com/terms/a/artificial-neural-networks-ann.asp. [Accesat 17 Mai 2021].

[6] G. De'ath and K. E. Fabricius, CLASSIFICATION AND REGRESSION TREES: A POWERFUL YET SIMPLE TECHNIQUE FOR ECOLOGICAL DATA ANALYSIS, 2000.

[7] N. S. Chauhan, "Decision Tree Algorithm, Explained," 2020. Disponibil: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html. [Accesat 28 Mai 2021].

[8] M. R. Segal, Machine Learning Benchmarks and Random Forest Regression, 2004.

[9] M. Belgiu and L. Dragut, Random forest in remote sensing: A review of applications and future directions, 2016.

[10] A. Luashchuk, "Why I Think Python is Perfect for Machine Learning and Artificial Intelligence," 2019. Disponibil: https://towardsdatascience.com/8-reasons-why-python-is-good-for-artificial-intelligence-and-machine-learning-4a23f6bed2e6. [Accesat 28 Mai 2021].

[11] I. Rodriguez, "What Is Pip? A Guide for New Pythonistas" 2020. Disponibil: https://realpython.com/what-is-pip/. [Accesat 28 Mai 2021].

[12] S. Suehring, MySQL Bible, 2002.

[13] "About AWS," 2021. Disponibil: https://aws.amazon.com/about-aws/. [Accesat 30 Mai 2021].

[14] M. Perez, "What is Web Scraping and What is it Used For?," 2019. Disponibil: https://www.parsehub.com/blog/what-is-web-scraping/. [Accesat 31 Mai 2021].

[15] "Sklearn RandomForestRegressor Documentation," 2020. Disponibil: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accesat 1 Iunie

2021].

[16] K. Maladkar, "Why Is Random Search Better Than Grid Search For Machine Learning," 2018. Disponibil: https://analyticsindiamag.com/why-is-random-search-better-than-grid-search-for-machine-learning/. [Accesat 31 Mai 2021].

[17] "Jinja Documentation," 2007. Disponibil: https://jinja.palletsprojects.com/en/3.0.x/. [Accesat 2 Iunie 2021].

[18] R. Fielding, J. Gettys and H. Frystyk, Hypertext Transfer Protocol -- HTTP/1.1, 1999.

[19] "Scrapy Documentation," 2021. Disponibil: https://docs.scrapy.org/en/latest/. [Accesat 2 Iunie 2021].

[20] R. Duncan, "A Simple Guide to HTML," 2017. Disponibil: http://www.simplehtmlguide.com/whatishtml.php. [Accesat 4 Iunie 2021].

[21] D. J. Barrett and R. E. Silverman, SSH, The Secure Shell. The Definitive Guide, 2001.

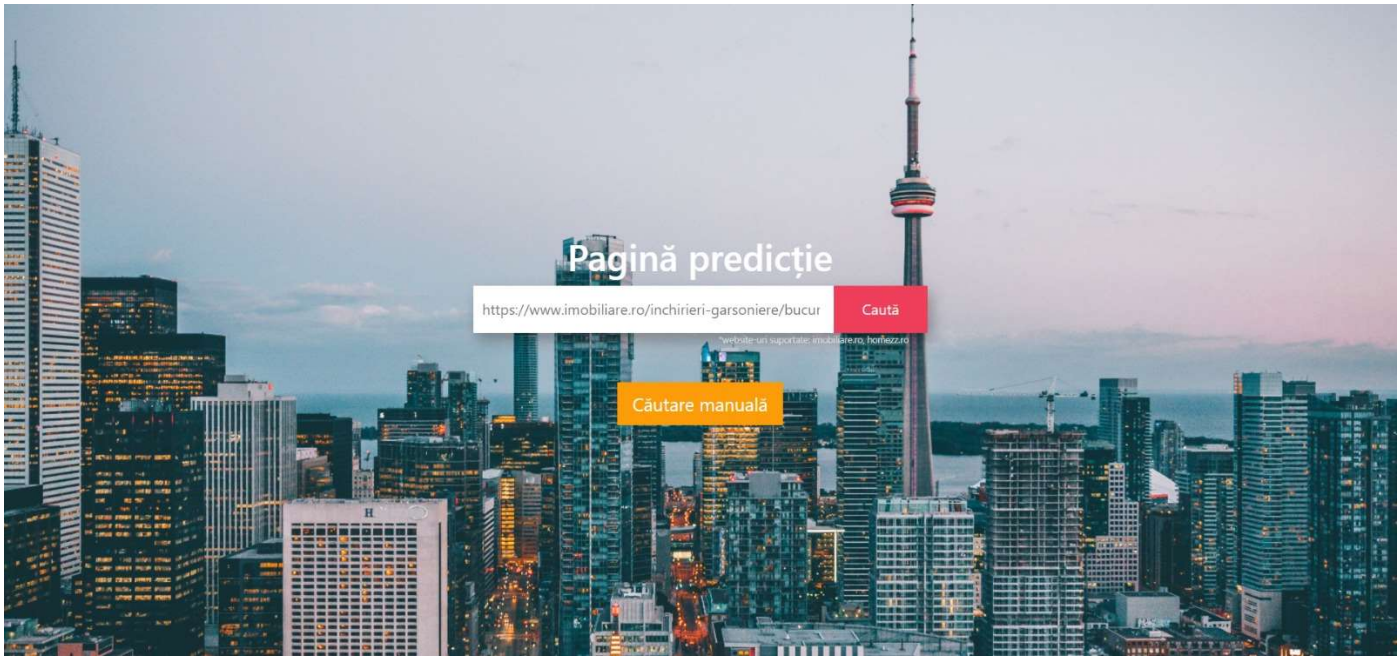
Fig 2.1 Semih Dinc, Ramazan Aygun. "Evaluation of Hyperspectral Image Classification Using Random Forest and Fukunaga-Koontz Transform". 2013

Fig 3.1 Documentație Microsoft Azure – „What is IaaS?” https://azure.microsoft.com/en-us/overview/what-is-iaas/ [Accesat 30 Mai 2021]

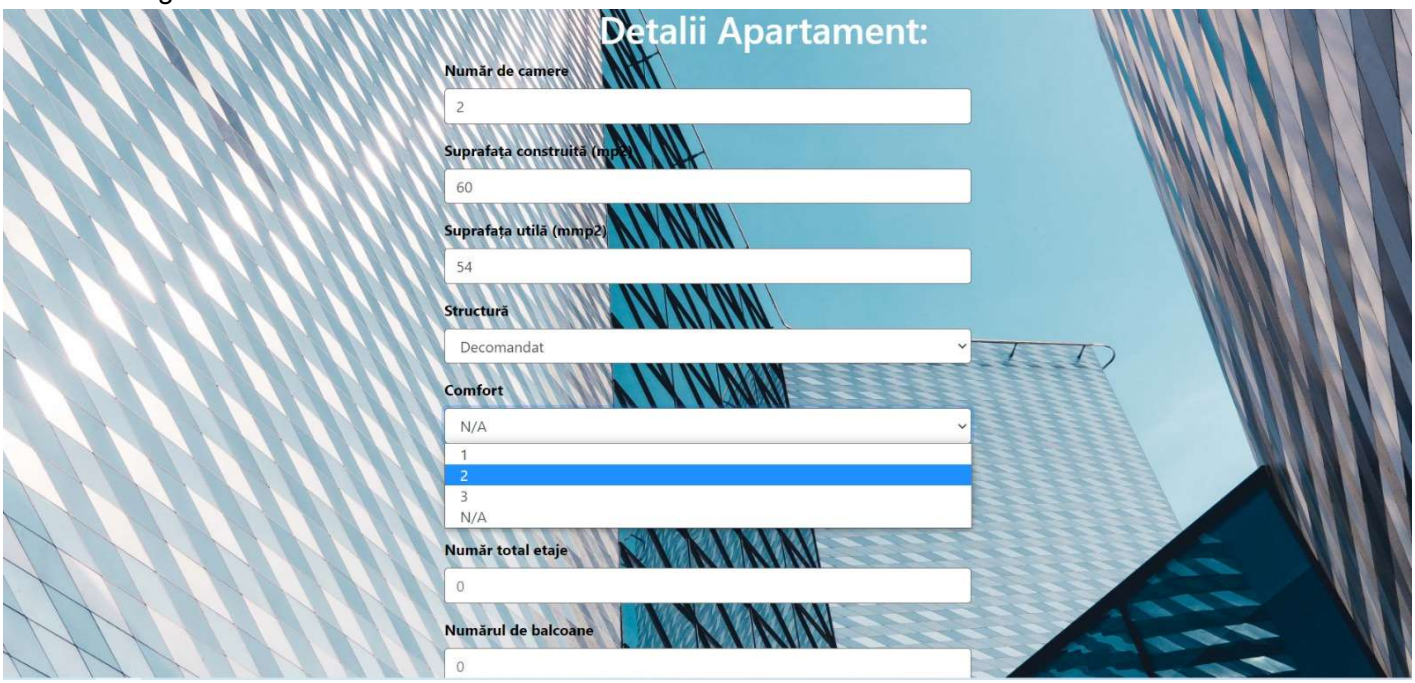
Fig 4.6 Documentație Scrapy „Architecture overview” https://docs.scrapy.org/en/latest/topics/architecture.html [Accesat 2 Iunie 2021]

Anexa 1

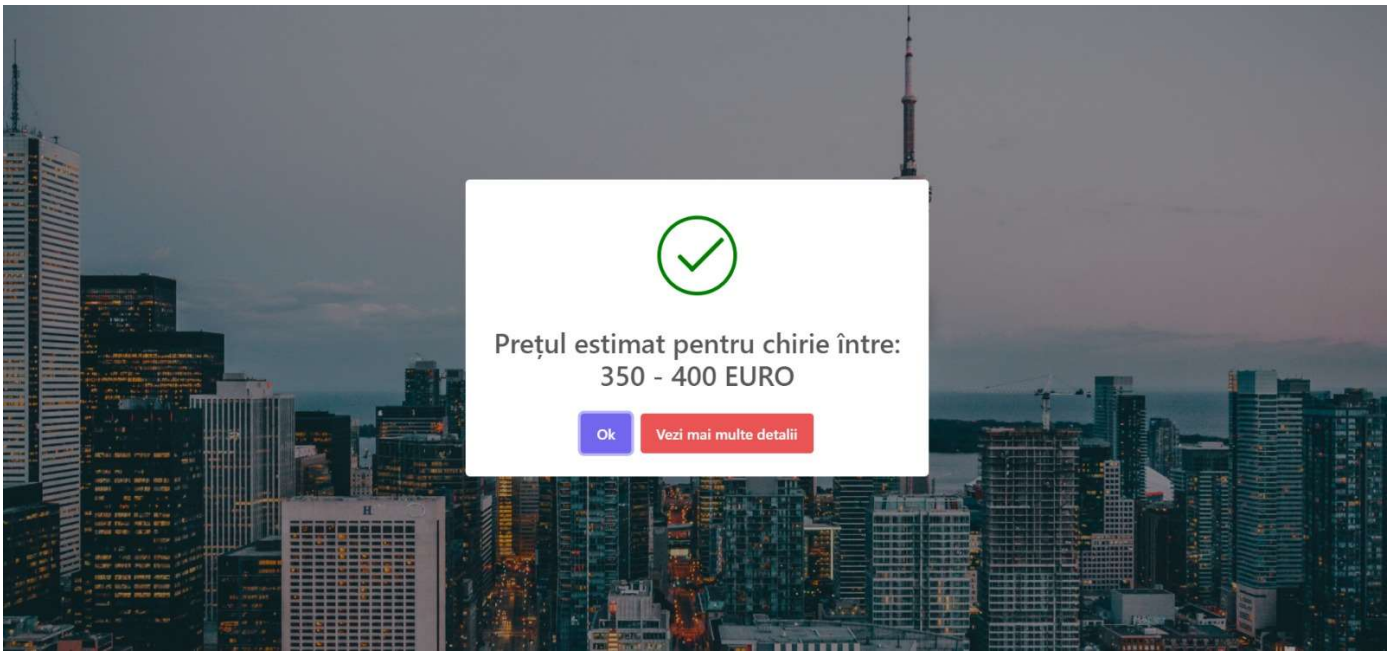
1.1 Pagina introducere URL



1.2 Pagina introducere detalii manual



1.3 Alertă predicție



1.4 Pagina „Mai multe detalii”