Adaptivni sustav za generiranje preporuka pri kupovini i prodaji automobila temeljen na analizi javno dostupnih podataka

Krkobabić, Saša

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 346

ADAPTIVNI SUSTAV ZA GENERIRANJE PREPORUKA PRI KUPOVINI I PRODAJI AUTOMOBILA TEMELJEN NA ANALIZI JAVNO DOSTUPNIH PODATAKA

Saša Krkobabić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 346

ADAPTIVNI SUSTAV ZA GENERIRANJE PREPORUKA PRI KUPOVINI I PRODAJI AUTOMOBILA TEMELJEN NA ANALIZI JAVNO DOSTUPNIH PODATAKA

Saša Krkobabić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 346

Pristupnik:	**Saša Krkobabić (0036526744)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentorica:	prof. dr. sc. Ljiljana Brkić

Zadatak: **Adaptivni sustav za generiranje preporuka pri kupovini i prodaji automobila temeljen na analizi javno dostupnih podataka**

Opis zadatka:

Preporučiteljski sustavi (engl. Recommender Systems) su programski sustavi koji korištenjem prikladnih algoritama predlažu proizvode ili usluge na temelju individualnih preferencija i ponašanja korisnika. Preporuke se mogu odnositi na tematski vrlo raznolika područja. Preporučiteljski sustav izgrađen i prilagođen generiranju preporuka za određenu domenu, u pravilu je neprimjenjiv za generiranje preporuka iz drugih domena. U okviru diplomskog rada potrebno je proučiti metode i principe rada preporučiteljskih sustava. Posebnu pozornost obratiti na iskoristivost nestrukturiranih tekstualnih podataka pri geneririranju preporuka. Proučiti načine na koje se u preporučiteljskim sustavima prevladava problem nepoznavanja korisnikovih interesa (engl. cold start problem) koji predstavlja poteškoću pri generiranju preporuka. Osmisliti i implementirati način prevladavanja navedenog problema i izgradnje korisničkog profila. Korištenjem metoda prikladnih za domenu kupnje i prodaje automobila izgraditi sustav koji će, temeljem korisničkog profila i prikupljenih podataka o kupnji/prodaji automobila, generirati relevantne preporuke za kupnju automobila i određivanje cijene automobila kojeg korisnik želi ponuditi na prodaju. Podatke o automobilima potrebno je prikupljati i pohranjivati periodički pozadinskim procesima iz javno dostupnih izvora s unaprijed definiranih sjedišta weba. Odaberite prikladan sustav za pohranu podataka te implementirajte prateću web-aplikaciju.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

U današnjem digitalnom dobu, internet je postao izuzetno bogato skladište informacija i resursa koje neprestano raste i širi svoje granice. Usporedba s tradicionalnim fizičkim trgovinama jasno ilustrira ovu transformaciju. Na jednoj strani postoje fizičke trgovine koje su ograničene prostorom za skladištenje, a na drugoj strani su online platforme koje ne poznaju takva ograničenja te nude nevjerojatnu raznolikost proizvoda, filmova, glazbe, knjiga i drugih sadržaja. Na primjer, dok su klasične video-trgovine mogle ponuditi ograničen broj DVD-ova ili VHS kaseta, online trgovine poput Amazona ili streaming servisi poput Netflixa pružaju pristup tisućama filmova i serija koje su dostupne na zahtjev korisnika.

Ova beskonačna ponuda predstavlja i izazov i priliku za korisnike. S jedne strane, obilje opcija omogućuje korisnicima da pronađu specifične proizvode ili sadržaje koji ih zanimaju, bez obzira na geografsku lokaciju ili raspoloživost na lokalnom tržištu. S druge strane, pretraga i navigacija kroz toliko raznolikih opcija mogu biti naporne pa zahtijevaju efikasne alate za filtriranje i preporuke.

U takvom kontekstu preporučiteljski sustavi i modeli strojnog učenja postaju ključni. Preporučiteljski sustavi koriste složene algoritme za analizu podataka o ponašanju korisnika, preferencijama, povijesti interakcija i drugim kontekstualnim informacijama kako bi predvidjeli što će korisnicima biti zanimljivo ili korisno. Na primjer, algoritmi mogu analizirati prethodne kupovine korisnika i preporučiti slične proizvode ili usluge, temeljene na njihovim preferencijama. Također, mogu prepoznati sličnosti između različitih proizvoda, filmova ili glazbenih uradaka te preporučiti korisnicima one koji bi im mogli biti interesantni na temelju njihovih dosadašnjih interakcija.

Modeli strojnog učenja su ključni alati za analizu velikih skupova podataka jer omogućuju izvlačenje korisnih informacija i obrazaca iz podataka koji su često složeni i veliki. U kontekstu online trgovina i streaming servisa, ovi modeli mogu primijeniti različite tehnike kako bi poboljšali korisničko iskustvo i ponudili personalizirane preporuke.

Ovaj diplomski rad temelji se na istraživanju preporučiteljskih sustava, analizi različitih metoda njihove implementacije, principa na kojima se temelje kao i primjenu različitih algoritama. Dodatno, istražuje se i opisuje različite vrste strojnog učenja te njihove

specifičnosti. Fokus je na razumijevanju izazova s kojima se susreću preporučiteljski sustavi te načinima kako ih prevladati.

U sklopu ovog rada razvijena je i implementirana aplikacija koja demonstrira primijenjene modele i tehnike. Cilj je prikazati kako teorijska saznanja o preporučiteljskim sustavima i strojnom učenju mogu biti praktično primijenjena u realnom svijetu, kroz konkretnu aplikaciju koja korisnicima pruža personalizirane preporuke na temelju njihovih preferencija i ostale korisne informacije.

1. Preporučiteljski sustavi

Preporučiteljski sustavi su sofisticirani softverski alati koji imaju sve veću ulogu u suvremenom digitalnom svijetu. Njihova osnovna svrha je pružiti personalizirane preporuke korisnicima o različitim proizvodima, uslugama, sadržaju ili drugim elementima na temelju njihovih prethodnih interakcija, ponašanja ili izraženih preferencija. Ovi sustavi koriste širok spektar podataka kako bi predvidjeli što će se korisniku svidjeti i ponudili mu relevantne preporuke.

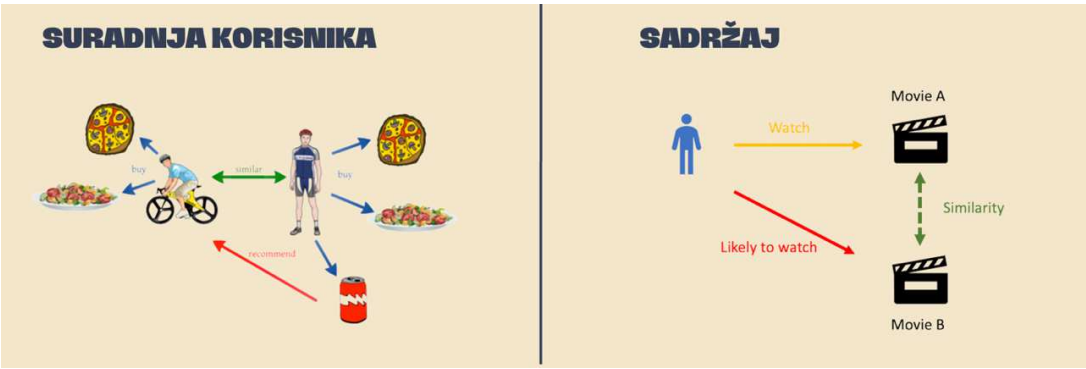
Povijest kupovina, pregledani proizvodi, ocjene, komentari, ponašanje na društvenim mrežama te brojne druge informacije služe kao ulazni podaci u preporučiteljske sustave. Na temelju tih podataka, sustavi koriste različite tehnike i algoritme kako bi stvorili profil korisnika i identificirali obrasce ponašanja. Primjena umjetne inteligencije, strojnog učenja i analize podataka omogućuje ovim sustavima da dublje razumiju individualne interese i preferencije korisnika te im nude personalizirane preporuke koje su prilagođene njihovim specifičnim potrebama.

Primjena preporučiteljskih sustava je izuzetno raznovrsna i obuhvaća različite industrije kao što su e-trgovina, streaming servisi, društvene mreže, mediji, turizam i još mnoge druge. Na primjer, u e-trgovini, ovi sustavi pomažu korisnicima da lakše pronađu proizvode koji odgovaraju njihovim ukusima i potrebama, čime se povećava korisničko iskustvo i vjerojatnost kupovine. U streaming servisima, preporučiteljski sustavi predlažu filmove, serije ili glazbu temeljenu na dosadašnjim gledanjima ili slušanjima korisnika, čime se personalizira sadržaj i povećava angažman.

Uz svoju ključnu ulogu u poboljšanju korisničkog iskustva i povećanju poslovne učinkovitosti, preporučiteljski sustavi ostaju centralni element digitalne strategije mnogih organizacija, omogućujući im da bolje razumiju svoje korisnike, povećaju zadovoljstvo korisnika i ostvare veći uspjeh na tržištu.

Od modela preporučiteljskih sustava postoje sustavi bazirani na sadržaju, suradničko filtriranje, preporuke temeljene na znanju i hibridni modeli.

Na slici (Slika 1.1) prikazana je razlika između dva modela preporučiteljskih sustava: modeli bazirani na sadržaju (*eng. content-based*) i suradničko filtriranje (*eng. collaborative filtering*).



Slika 1.1 Razlika između sustava temeljenom na suradnji korisnika i temeljeno na sadržaju

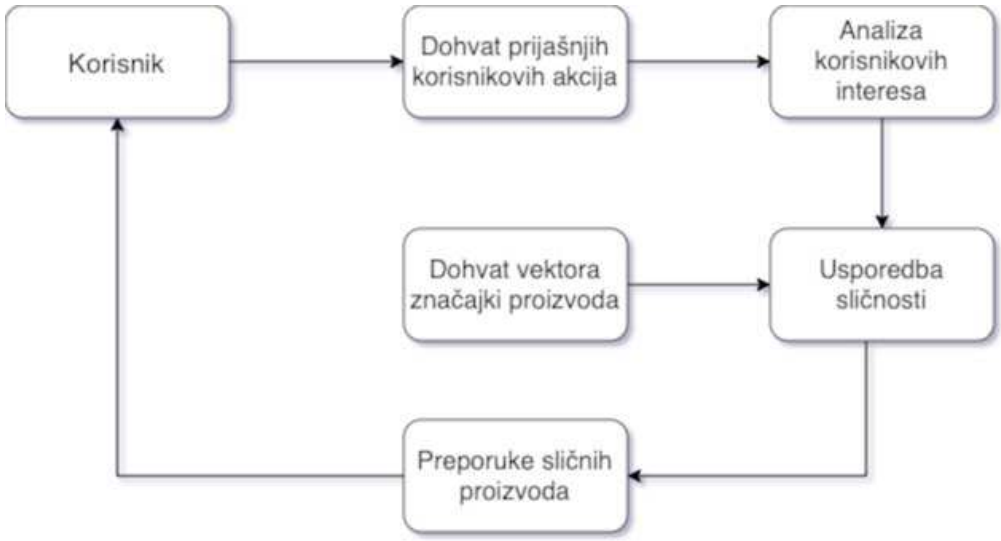
1.1. Sustavi bazirani na sadržaju

Sustavi bazirani na sadržaju koriste karakteristike ili značajke proizvoda ili sadržaja kako bi izveli preporuke. Ovi sustavi analiziraju atribute specifične za svaki predmet koji se preporučuje, u kontekstu filmova to bi bile značajke tipa žanr, godina izlaska, glavni glumci, ocjene, ključne riječi iz opisa i mnogi drugi faktori. Na primjer, prilikom preporučivanja filmova, sustav može identificirati slične filmove koje korisnik može uživati analizirajući značajke filmova koje je korisnik već gledao. Ako je korisnik gledao film znanstvene fantastike s visokim ocjenama i određenim glumcima, sustav će pretražiti bazu podataka za druge filmove sličnog žanra, ocjena i s istim ili sličnim glumcima te ih onda preporučiti korisniku.

Konkretno, kako bi preporučio film sličan filmu koji je korisnik već gledao, sustav bi mogao koristit tf-idf algoritam da izvadi ključne riječi iz opisa filma i ocjene korisnika za usporedbu s drugim filmovima koji dijele slične ključne riječi. Na taj način, sustav bi mogao preporučiti filmove koji su slični po sadržaju, žanru i stilu, ali koji nisu nužno popularni ili dobro ocijenjeni, ali bi se mogli svidjeti korisniku. Uz to, sustavi bazirani na sadržaju često koriste i druge vrste značajki, poput atributa proizvoda, cijene ili kategorije kako bi napravili konačnu preporuku.

Kako ovi sustavi koriste samo značajke proizvoda, a ne i korisnikove interakcije onda su osobito korisni u situacijama gdje nema puno informacija o korisnicima poput novih korisnika. Također su korisni za proizvode koji nemaju puno interakcije kao na primjer proizvodi koji su rijetki ili nedavno pušteni na tržište. Integracija različitih značajki omogućuje sustavu da uzme u obzir više aspekata proizvoda ili sadržaja prilikom izvođenja preporuka. Pojednostavljena ilustracija rada sustava baziranog na sadržaju prikazan je

slikom (Slika 1.2). Sustav dohvaća prijašnje korisnikove akcije što može uključivati ocjene predmeta, preglede, kupnje i slično. Iz tih podataka predviđa korisnikove interese koji se mogu prikazati u obliku vektora gdje svaka vrijednost određuje koliko neka značajka interesira tog korisnika. Dohvaćaju se vektori značajki proizvoda i uspoređuju sa vektorom korisnika te se slični proizvodi onda preporučuju korisniku.



Slika 1.2 Prikaz rada sustava baziranog na sadržaju

TF-IDF (term frequency–inverse document frequency) je matematički model za određivanje sličnosti tekstualnih sadržaja koji uzima u obzir dvije komponente: frekvenciju pojave riječi u dokumentu (term frequency - TF) i inverznu frekvenciju pojave riječi u drugim dokumentima (inverse document frequency - IDF).

TF ocjenjuje važnost riječi unutar pojedinog dokumenta. Riječi koje se često pojavljuju u dokumentu dobivaju veću ocjenu, sugerirajući da su važne za sadržaj tog dokumenta. TF se računa prema izrazu (1)

$$TF(t) = \frac{N(t)}{N} \tag{1}$$

gdje je $N(t)$ broj pojavljivanja riječi t u dokumentu, a N ukupni broj riječi u cijelom dokumentu.

IDF procjenjuje važnost riječi u cijeloj kolekciji dokumenata. Riječi koje se rijetko pojavljuju u drugim dokumentima, a česte su u trenutnom dokumentu, dobivaju veću težinu. To ukazuje na to da takve rijetke riječi mogu biti ključne za razumijevanje sadržaja tog dokumenta. IDF se računa prema izrazu (2)

$$IDF(t) = \log\frac{N}{N_d(t)} \tag{2}$$

gdje je N ukupan broj dokumenata, a $N_d(t)$ broj dokumenata koji sadrže riječ t.

Za kraj kombiniranjem TF i IDF, tf-idf algoritam izračunava težinu svake riječi u dokumentu koja se računa prema izrazu (3).

$$TF - IDF(t) = TF(t) \times IDF(t) \tag{3}$$

Riječi s većom težinom smatraju se značajnijima za sadržaj tog dokumenta.

Svaki dokument se pretvori u vektor tako da svaka dimenzija odgovara jednoj riječi iz rječnika svih dokumenata. Vrijednost svake dimenzije u vektoru je TF-IDF vrijednost odgovarajuće riječi u tom dokumentu. Ako neka riječ nije prisutna u dokumentu, njezina TF-IDF vrijednost je nula.

Dakle, određivanje sličnosti teksta se svelo na određivanje sličnosti njihovih vektora. Za taj izračun najčešće se koristi **kosinus sličnosti** koja se izračunava pomoću formule (3)

$$kosinus\ sličnost = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\|\|\vec{B}\|} \tag{3}$$

odnosno podijeli se skalarni produkt ta dva vektora s umnoškom njihovih Euklidskih normi.

Kosinus sličnost vraća vrijednost između -1 i 1 gdje 1 označava da su vektori potpuno slični, a -1 potpuno različiti.

1.1.1. Primjer

Jedan od najpoznatijih primjera sustava baziranog na sadržaju Amazon dolazi iz svijeta online trgovine. Kada korisnik kupi određeni proizvod, Amazonov preporučiteljski sustav analizira značajke tog proizvoda, uključujući kategoriju, marku, cijenu, tehničke specifikacije i recenzije. Na temelju tih značajki, sustav preporučuje druge proizvode koji dijele slične karakteristike. Na primjer, ako korisnik kupi laptop određene marke, sustav će preporučiti druge laptope iste ili slične marke, sličnih tehničkih specifikacija ili iz iste cjenovne kategorije.

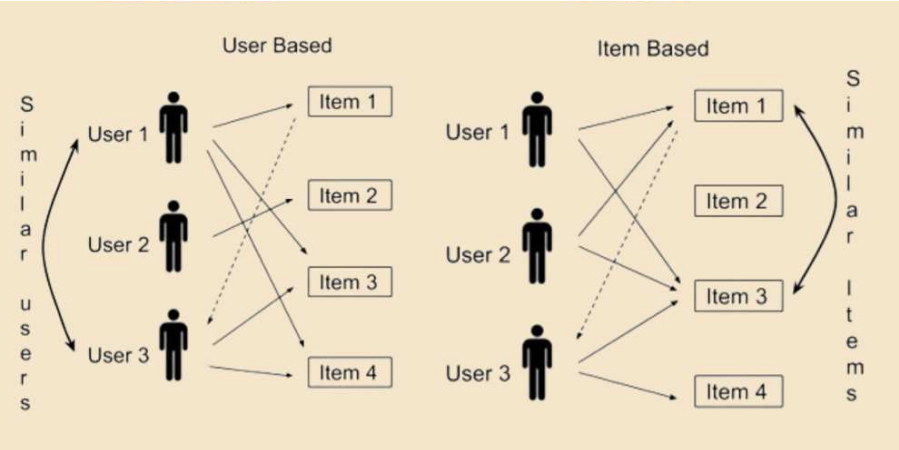
Također, ako korisnik kupi kuhinjski aparat, sustav može preporučiti slične kuhinjske aparate ili dodatke koji idu uz taj aparat. Ovakav sustav omogućuje korisnicima da otkriju proizvode koji su relevantni za njihove potrebe na temelju sličnosti s već kupljenim proizvodima, čime se povećava zadovoljstvo korisnika i vjerojatnost ponovne kupovine. Iako ovaj sustav može pružiti visoko relevantne preporuke, može također dovesti do ograničene raznolikosti preporučenih proizvoda.

Na slici (Slika 1.3) se može primijetiti da se korisniku koji je kupio Kindle preporučuju slični proizvodi kao na primjer navlake za Kindle ili noviji model.

Slika 1.3 Primjer sustava baziranog na sadržaju [1]

1.2. Suradničko filtriranje

Suradničko filtriranje (*collaborative filtering*) je popularna tehnika u području preporuka koja se temelji na suradnji među korisnicima ili predmetima kako bi se pružile personalizirane preporuke. Ova metoda pretpostavlja da ako dva korisnika dijele slične ukuse ili preferencije u prošlosti, vjerojatno će im se svidjeti isti ili slični predmeti u budućnosti. Postoje dvije osnovne vrste suradničkog filtriranja: filtriranje temeljeno na korisnicima (*user-based*) i filtriranje temeljeno na predmetima (*item-based*). Ilustraciju ta dva pristupa može se vidjeti na slici (Slika 1.4) gdje je na lijevoj strani prikazano filtriranje temeljeno na korisnicima, a na desnoj filtriranje temeljeno na predmetima.



Slika 1.4 Razlike između korisnik-korisnik i predmet-predmet pristupa

Filtriranje temeljeno na korisnicima analizira sličnost između korisnika na temelju njihovih ocjena, povratnih informacija ili ponašanja. Na taj način se pronalaze slični korisnici i preporučuju predmeti koje su slični korisnici ocijenili visoko, a koji su korisniku još nepoznati. Ova metoda koristi matricu korisnika i predmeta, gdje su redci korisnici, a stupci predmeti koje su ocijenili. Algoritmi poput k najbližih susjeda (k-NN) koriste se za pronalaženje korisnika s najsličnijim ocjenama.

Filtriranje temeljeno na predmetima, s druge strane, usredotočuje se na sličnost među predmetima na temelju korisničkih ocjena ili drugih interakcija s tim predmetom. Ova metoda pronalazi slične predmete na temelju sličnih uzoraka ocjena od strane korisnika i preporučuje korisniku predmete slične onima koje je već ocijenio. Ovdje se koristi matrica sličnosti predmeta, gdje se za svaki par predmeta izračunava sličnost koristeći metode poput kosinus sličnosti ili Pearsonovog korelacijskog koeficijenta.

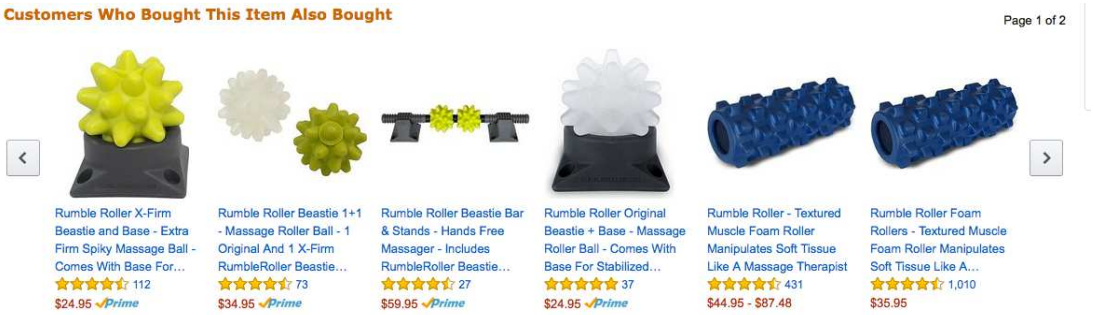
Suradničko filtriranje može biti vrlo učinkovito jer se oslanja na ponašanje stvarnih korisnika i njihove povratne informacije. To omogućuje generiranje personaliziranih preporuka koje uzimaju u obzir individualne preference i interese korisnika. Ova tehnika je široko primijenjena u različitim područjima, uključujući preporuke filmova, glazbe, knjiga, proizvoda i još mnogo toga.

1.2.1. Primjer

Jedan od najpoznatijih primjera suradničkog filtriranja u stvarnom svijetu je Netflix. Netflix koristi suradničko filtriranje kako bi preporučio filmove i serije korisnicima na temelju

njihovih prethodnih ocjena i ponašanja. Ako dva korisnika imaju slične ocjene za niz filmova, sustav će preporučiti filmove koje je jedan korisnik ocijenio visoko drugom korisniku. Slično, ako je korisnik ocijenio određeni film visoko, Netflix će preporučiti slične filmove na temelju ocjena drugih korisnika koji su također ocijenili taj film visoko.

Amazon također koristi suradničko filtriranje kako bi preporučio proizvode. Kada korisnik kupi ili ocijeni proizvod, Amazon analizira te podatke u kontekstu svih ostalih korisnika koji su kupili ili ocijenili slične proizvode. Na temelju toga, Amazon može preporučiti druge proizvode koje su ti korisnici također kupili ili ocijenili visoko. Na primjer, ako korisnik kupi digitalni fotoaparat, Amazon može preporučiti dodatke poput objektiva, torbi za kamere ili memorijskih kartica koje su kupili drugi korisnici koji su kupili isti fotoaparat. Jedan takav primjer prikazan je na slici (Slika 1.5) gdje se može vidjeti da Amazon ima sekciju gdje prikazuje koje stavke su korisnici kupili uz određeni proizvod.



Slika 1.5 Primjer sustava temeljeno na suradnji korisnika [1]

1.3. Preporuke temeljene na znanju

Preporučivanje temeljeno na znanju (*eng. Knowledge-Based Recommendation*) je pristup koji se koristi u situacijama gdje se stavke ne kupuju često ili gdje nema dovoljno podataka o povijesnim interakcijama korisnika s određenim proizvodima ili uslugama. Ovaj pristup je posebno koristan u domenama kao što su kupovina nekretnina, turističke ponude ili općenito kupovina predmeta koji se kupuju relativno rijetko, poput pametnih telefona. U takvim situacijama, tradicionalne metode preporučivanja, koje se oslanjaju na povijesne ocjene ili interakcije korisnika, nisu učinkovite jer nedostaje dovoljno podataka za analizu.

Umjesto toga, preporučivanje temeljeno na znanju koristi eksplicitno izražene interese korisnika kako bi se generirale relevantne preporuke. Korisnicima se prezentiraju različiti atributi proizvoda ili usluga, poput cijene, veličine, boje, značajki i drugih specifičnih karakteristika, ovisno o domeni. Korisnik tada može prilagoditi te atribute prema svojim

preferencijama. Na primjer, prilikom pretraživanja nekretnina, korisnik može definirati željeni raspon cijene, broj soba, starost nekretnine, lokaciju i druge relevantne parametre. Sustav zatim filtrira dostupne opcije prema tim preferencijama i prikazuje samo one koje odgovaraju zadanim kriterijima.

Osim eksplicitnog definiranja atributa, korisnicima se mogu ponuditi primjeri stavki koje oni onda ocjenjuju te sustav iz tih podatka predviđa interese.. Na primjer, sustav može prikazati nekoliko primjeraka nekretnina, a korisnik može odabrati one koje mu se sviđaju. Na temelju tih odabira, sustav može bolje razumjeti korisnikove preferencije i ponuditi slične stavke. Ovaj pristup je sličan preporučivanju na temelju sličnosti sadržaja, ali ključna razlika je u tome što korisnik izravno definira svoje interese, umjesto da sustav zaključuje preferencije iz povijesnih interakcija.

Prednosti preporučivanja temeljenog na znanju su brojne. Prvo, ovaj sustav ne ovisi o velikoj količini podataka o povijesnim interakcijama korisnika, što ga čini idealnim za domene gdje se stavke rijetko kupuju ili koriste. Drugo, korisnici imaju potpunu kontrolu nad definiranim atributima, što može povećati njihovo zadovoljstvo preporukama jer su one prilagođene njihovim specifičnim zahtjevima. Međutim, ovaj sustav također ima svoje mane. Bez jasnih i eksplicitno definiranih korisnikovih interesa, sustav ne može funkcionirati učinkovito. Također, ako korisnici nisu sigurni koje atribute preferiraju ili ako nisu voljni investirati vrijeme u prilagođavanje tih atributa, sustav može generirati manje relevantne preporuke.

1.3.1. Primjer

Primjer preporuka temeljenih na znanju može se vidjeti u sustavu za putovanja. U ovom slučaju, od korisnika se traži da unese svoje preferencije vezane uz putovanje, kao što su budžet, trajanje putovanja, vrsta odmora, preferirane destinacije i aktivnosti. Na primjer, korisnik bi mogao unijeti da ima budžet između 1000 i 2000 eura, da planira putovanje od 7 dana, da preferira odmor na plaži, planinarenje i kulturne ture, te da ga zanimaju destinacije poput Kariba ili Mediterana, uz aktivnosti poput posjeta muzeju i planinarenja. Na temelju ovih podataka sustav onda predlaže putovanja koja bi mogla odgovarati korisniku. Ovaj pristup omogućava personalizirane preporuke koje su prilagođene specifičnim zahtjevima i interesima korisnika, osiguravajući da preporučene destinacije i aktivnosti najbolje odgovaraju njihovim preferencijama.

1.4. Hibridni pristup

Hibridni preporučiteljski sustavi kombiniraju tri glavna modela preporučivanja: kolaborativno filtriranje, filtriranje temeljeno na sadržaju i preporučivanje temeljeno na znanju. Kolaborativno filtriranje koristi podatke o sličnim korisnicima za generiranje preporuka, filtriranje temeljeno na sadržaju analizira atribute stavki koje korisnik preferira, dok preporučivanje temeljeno na znanju koristi eksplicitno izražene interese korisnika i specifične atribute stavki. Kombiniranjem ovih pristupa, hibridni sustavi mogu iskoristiti prednosti svakog modela i nadoknaditi njihove slabosti, pružajući točnije i personaliziranije preporuke. Na primjer, kada nedostaje dovoljno podataka za kolaborativno filtriranje, sustav može koristiti sadržaj ili eksplicitne preferencije korisnika za generiranje relevantnih preporuka, osiguravajući time dosljednu kvalitetu preporuka u različitim kontekstima.

1.5. Cold-start

Cold-start problem je jedan od najznačajnijih izazova s kojima se susreću preporučiteljski sustavi. Ovaj problem nastaje kada sustav nema dovoljno informacija o novim korisnicima ili novim stavkama, što otežava generiranje relevantnih preporuka. U osnovi, cold-start problem se može podijeliti na tri glavna scenarija: problem novog korisnika, problem nove stavke i problem početnog sustava.

Problem novog korisnika javlja se kada novi korisnik pristupi sustavu, a sustav nema podataka o njegovim preferencijama ili povijesti interakcija. Bez ovih podataka, sustav ne može primijeniti kolaborativno filtriranje, koje se oslanja na podatke o sličnim korisnicima, niti filtriranje temeljeno na sadržaju, koje zahtijeva informacije o preferencijama korisnika prema specifičnim atributima stavki. Jedan od pristupa rješavanju ovog problema je postavljanje početnog upitnika korisnicima kako bi se prikupile osnovne informacije o njihovim interesima i preferencijama.

Problem nove stavke nastaje kada se u sustav dodaju nove stavke za koje ne postoje ocjene ili povijesne interakcije korisnika. Bez ovih podataka, sustav ne može preporučiti nove stavke korisnicima koristeći tradicionalne metode kolaborativnog filtriranja. Rješenje za ovaj problem može uključivati korištenje filtriranja temeljenog na sadržaju, gdje se analiziraju atributi novih stavki i uspoređuju s atributima stavki koje su korisnici već ocijenili pozitivno.

Problem početnog sustava javlja se kada je preporučiteljski sustav tek pokrenut i nema dovoljno podataka o korisnicima ili stavkama. U ovoj fazi, svi korisnici i stavke su novi, što značajno ograničava sposobnost sustava da generira točne preporuke. Za rješavanje ovog problema, sustavi često koriste metode poput hibridnih preporuka, koje kombiniraju različite tehnike preporučivanja, uključujući demografske podatke, sadržaj i eksplicitno definirane interese korisnika. Također, sustavi mogu koristiti inicijalne marketinške kampanje ili poticaje kako bi potaknuli korisnike da ostavljaju ocjene i recenzije, čime se ubrzava prikupljanje podataka potrebnih za učinkovitije preporuke.

Jedan od inovativnih pristupa rješavanju cold-start problema je korištenje vanjskih izvora podataka, poput društvenih mreža, gdje korisnici već imaju izražene interese i interakcije. Integriranjem ovih podataka, preporučiteljski sustav može bolje razumjeti preferencije novih korisnika i generirati relevantnije preporuke od samog početka.

Dakle cold-start problem predstavlja značajan izazov za preporučiteljske sustave, ali kombinacijom različitih metoda i tehnika, uključujući upitnike za nove korisnike, analiziranje atributa novih stavki, korištenje hibridnih sustava i integraciju vanjskih izvora podataka, moguće je ublažiti njegove negativne učinke i pružiti korisnicima kvalitetne preporuke čak i u početnim fazama korištenja sustava.

2. Strojno učenje

Strojno učenje je grana umjetne inteligencije koja se bavi razvojem algoritama i tehnika kojima računalni sustavi uče iz podataka, bez izričitog programiranja. Osnovna ideja je da računalo može identificirati obrasce i donositi zaključke iz podataka, te poboljšavati performanse s iskustvom. Postoje tri osnovne vrste učenja: nadzirano učenje, nenadzirano učenje i učenje pojačanjem.

Nadzirano učenje se primjenjuje na podatke koji sadrže parove ulaznih i željenih izlaznih podataka.

Nenadzirano učenje se koristi kada su podaci neoznačeni ili nema jasnih izlaznih vrijednosti te model u njima nalazi obrasce.

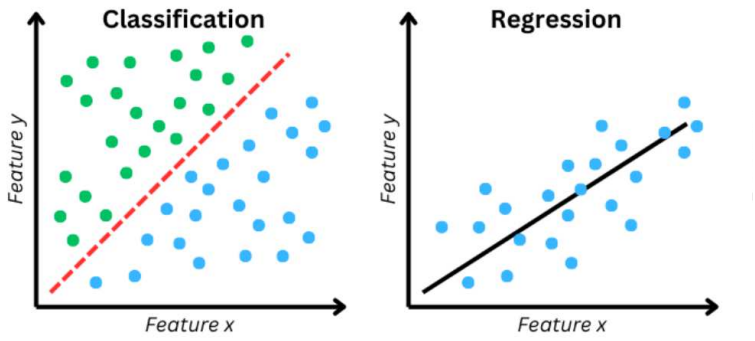
Učenje pojačanjem uključuje učenje kako bi sustav postigao određeni cilj putem iterativnog iskustva i povratnih informacija u obliku nagrada ili kazna.

Strojno učenje se koristi u različitim područjima poput prepoznavanja uzoraka, klasifikacije, predikcije i optimizacije. Izlaz modela može biti regresijski ili klasifikacijski.

Regresija traži korelaciju između zavisnih i nezavisnih varijabli te pokušava predvidjeti kontinuiranu varijablu poput cijene.

Klasifikacija pokušava podijeliti skup podataka u skupine te onda ulazu dodijeliti diskretnu vrijednost jednog od tih skupova. Tako na primjer možemo koristiti klasifikaciju kako bismo odredili je li neki e-mail neželjena pošta, tu bismo imali dva skupa: da i ne.

Na slici (Slika 2.1) prikazana je vizualna razlika između te dvije vrste modela.



Slika 2.1 Razlika između klasifikacije i regresije [2]

2.1. Nadzirano učenje

Nadzirano učenje, temeljna paradigma u strojnom učenju, pruža računalima sposobnost učenja iz podataka koji su označeni ili klasificirani. Ovaj pristup se široko koristi u raznim područjima, od prepoznavanja uzoraka do predviđanja budućih vrijednosti. Bitan aspekt nadziranog učenja je sposobnost razvijanja modela koji može precizno mapirati ulazne podatke na odgovarajuće izlazne vrijednosti. Na primjer, u zadatku prepoznavanja lica, sustav bi mogao koristiti skup podataka koji sadrži fotografije ljudskih lica zajedno s imenima tih pojedinaca kako bi naučio prepoznavati pojedince na novim fotografijama.

Ključni korak u nadziranom učenju je proces treniranja modela. Tijekom ovog procesa, model se izlaže velikom broju primjera podataka zajedno s odgovarajućim oznakama ili izlaznim vrijednostima. Korištenjem različitih algoritama i tehnika, model se prilagođava podacima kako bi naučio odgovarajuće obrasce i zakonitosti. Važno je napomenuti da je kvaliteta i raznolikost skupa podataka ključna za uspješno treniranje modela u nadziranom učenju.

Nadzirano učenje omogućuje rješavanje širokog spektra problema, uključujući klasifikaciju, regresiju, detekciju anomalija i još mnogo toga. Primjene nadziranog učenja mogu se pronaći u medicini, financijama, marketingu, računalnom vidu i mnogim drugim područjima. Ovaj pristup ostaje ključan za razvoj inteligentnih sustava koji mogu donositi informirane odluke na temelju raspoloživih podataka, što ga čini neprocjenjivim alatom u suvremenom svijetu tehnologije i podataka.

2.2. Nenadzirano učenje

Nenadzirano učenje predstavlja ključnu paradigmu u strojnom učenju koja omogućuje računalima da otkriju obrasce, strukture i zakonitosti u neoznačenim podacima. Ova metoda je izuzetno važna u situacijama gdje nema dostupnih oznaka ili klasifikacija za podatke, što je čest slučaj u stvarnim scenarijima. Jedan od glavnih ciljeva nenadziranog učenja je grupiranje (*eng. clustering*) podataka, tj. identifikacija prirodnih grupa ili klastera unutar skupa podataka. Na primjer, u analizi kupovnih navika, nenadzirano učenje može identificirati skupine kupaca sličnih preferencija ili ponašanja, omogućavajući tvrtkama da prilagode marketinške strategije prema različitim segmentima klijenata.

Drugi važan aspekt nenadziranog učenja je smanjenje dimenzionalnosti (*eng. dimensionality reduction*) podataka. Ovo uključuje proces transformacije podataka s više dimenzija u prostor s manje dimenzija zadržavajući bitne informacije. Time se olakšava interpretacija i analiza podataka te se može smanjiti količina resursa potrebnih za pohranu i obradu podataka. Primjerice, u analizi slika visoke rezolucije, nenadzirano učenje može pomoći u identifikaciji ključnih značajki ili uzoraka koji opisuju sadržaj slike, omogućavajući efikasniju obradu i pretraživanje velikih skupova slika.

Nenadzirano učenje također može biti korisno za otkrivanje skrivenih značajki ili struktura u podacima koji nisu lako uočljivi ljudskim promatranjem. Ova sposobnost omogućava otkrivanje novih uvida ili informacija koje bi inače moglo biti teško ili nemoguće identificirati. Sve u svemu, nenadzirano učenje predstavlja snažan alat koji omogućava dublje razumijevanje podataka i njihovih skrivenih uzoraka, omogućavajući različite primjene u istraživanju, analizi podataka i donošenju odluka.

2.3. Učenje pojačanjem

Učenje pojačanjem je važna paradigma u strojnom učenju koja simulira način učenja ljudi i životinja kroz iskustvo i povratne informacije. Ova metoda se često koristi u situacijama gdje je potrebno donositi odluke kako bi se maksimizirala nagrada ili minimizirao gubitak kroz iterativno iskustvo. Ključni elementi učenja pojačanjem uključuju agenta, okoliš i nagrade. Agent je sustav koji donosi odluke, okoliš predstavlja kontekst u kojem agent djeluje, dok su nagrade signali koji ocjenjuju akcije agenta. Cilj učenja pojačanjem je naučiti strategiju ili politiku koja će maksimizirati ukupnu kumulativnu nagradu tijekom vremena.

Primjene učenja pojačanjem su raznolike i mogu se naći u područjima kao što su robotika, upravljanje resursima, financije i računalne igre. Na primjer, u kontekstu autonomne vožnje, sustav učenja pojačanjem može učiti kako donositi odluke u prometnim situacijama kako bi minimizirao rizik od nesreća ili optimizirao vrijeme dolaska na odredište. U financijskim tržištima, algoritmi učenja pojačanjem mogu se koristiti za automatsko trgovanje kako bi se maksimizirali profiti ulagača ili minimizirali gubici.

Jedna od ključnih izazova u učenju pojačanjem je ravnoteža između istraživanja novih akcija i iskorištavanja poznatih akcija. Previše istraživanja može rezultirati gubitkom vrijednih nagrada, dok premalo istraživanja može dovesti do suboptimalnih strategija. Stoga je važno

razviti algoritme učenja pojačanjem koji mogu prilagoditi razinu istraživanja i iskorištavanja kako bi se postigla optimalna ravnoteža.

Unatoč izazovima, učenje pojačanjem ostaje moćan alat za razvoj inteligentnih sustava koji mogu donositi informirane odluke u dinamičkim i nepredvidljivim okruženjima. Kroz iterativni proces učenja i interakcije s okolinom, sustavi učenja pojačanjem mogu postići visoku razinu prilagodljivosti i učinkovitosti u različitim situacijama, čineći ih ključnim dijelom suvremenih tehnoloških rješenja.

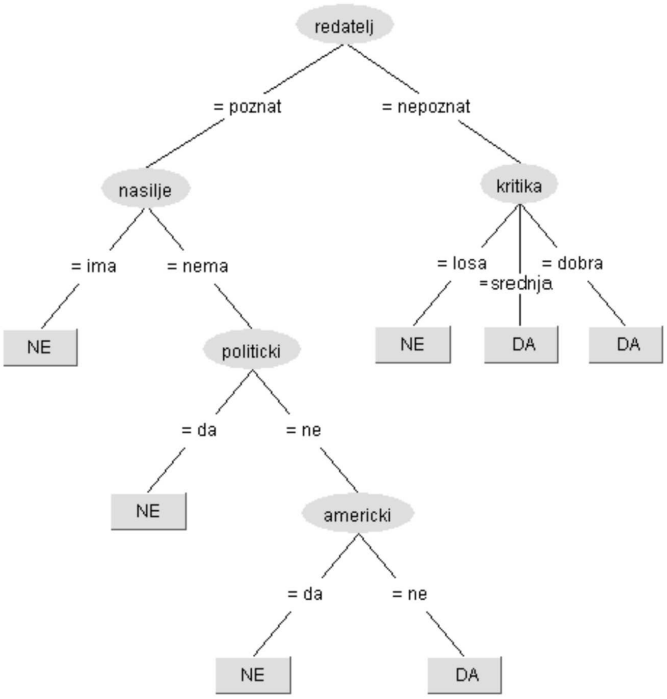
2.4. Model slučajnih šuma

U ovom radu korišten je model slučajnih šuma. Kako bi se shvatio taj model potrebno je prvo objasniti što su stabla odluke.

Stabla odluke su jednostavni, ali moćni algoritmi strojnog učenja koji se koriste za klasifikacijske i regresijske zadatke. Svako stablo sastoji se od čvorova i grana koje predstavljaju odluke na temelju vrijednosti atributa ulaznih podataka. Proces donošenja odluka u stablu odluke počinje od korijenskog čvora i nastavlja se prema dolje kroz unutarnje čvorove sve dok se ne dođe do listova, koji predstavljaju konačne odluke ili predikcije.

U klasifikacijskom stablu odluke, svaki unutarnji čvor predstavlja test na atribut (npr. "Je li dob > 30?"), a svaka grana predstavlja ishod testa (npr. "da" ili "ne"). Listovi predstavljaju klasifikacijske oznake (npr. "Da" ili "Ne"). U regresijskom stablu, listovi predstavljaju kontinuirane vrijednosti.

Na slici (Slika 2.2 [3]) je prikazan primjer stabla odlučivanja u kojem se odlučuje hoće li se gledati neki film. Čvorovi odlučivanja se sastoje od pitanja vezana za pojedini film i ovisno o odgovoru prelazimo na sljedeći čvor i ponavljamo dok ne dođemo do lista stabla. Listovi sadrže rezultat koji su u ovom slučaju „DA“ ili „NE“.



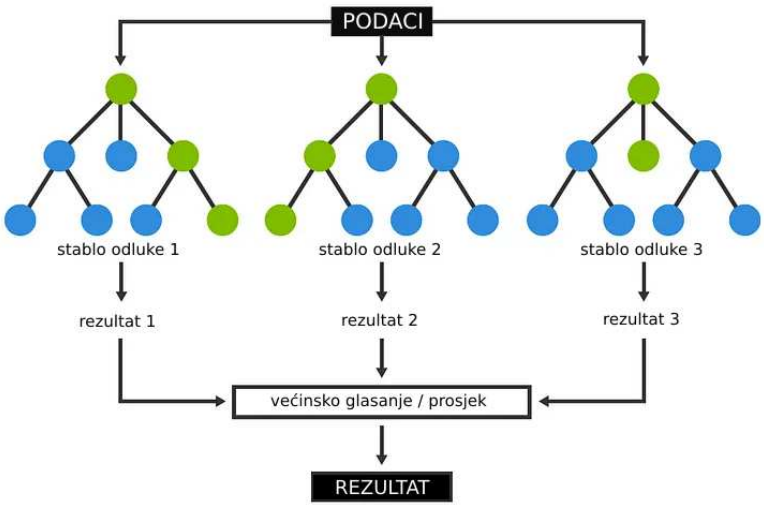
Slika 2.2 Stablo odlučivanja [4]

Prednosti stabala odluke uključuju jednostavnost interpretacije i vizualizacije, no glavni nedostatak je što su sklona prekomjernom učenju, posebno kada su duboka i složena.

Slučajne šume je popularni algoritam strojnog učenja koji su razvili Leo Breiman i Adele Cutler. Algoritam spaja rezultate brojnih stabala odlučivanja kako bi proizveo jedan ishod. Njegova popularnost proizlazi iz jednostavnosti upotrebe i svestranosti, što ga čini prikladnim i za zadatke klasifikacije i regresije. [3]

Svako stablo u šumi trenira se na različitim podskupovima originalnog skupa podataka, koji se stvaraju metodom nasumičnog uzorkovanja s ponavljanjem, poznatom kao *bootstrap sampling*.

Na slici (Slika 2.3) je prikazan primjer modela nasumičnih šuma u kojem možemo vidjeti da se koristi tri stabla odluke.



Slika 2.3 Model slučajnih šuma [5]

Dodatna nasumičnost uvodi se odabirom nasumičnog podskupa značajki za svaki čvor stabla, čime se smanjuje korelacija među stablima i povećava njihova raznolikost. Ovaj pristup omogućava modelu slučajnih šuma da postigne visoku točnost predikcija i da se bolje nosi s varijabilnošću podataka. Konačna odluka modela donosi se glasanjem svih stabala u šumi (u slučaju klasifikacije) ili izračunavanjem prosjeka njihovih predikcija (u slučaju regresije).

Jedna od ključnih prednosti modela slučajnih šuma je njegova robusnost na šum u podacima i stršeće vrijednost. Zbog agregacije brojnih stabala, slučajni šumovi mogu apsorbirati anomalije i smanjiti njihov utjecaj na konačne predikcije. Osim toga, ovaj model automatski procjenjuje važnost značajki, što omogućava uvid u to koji atributi najviše doprinose konačnim odlukama.

3. Aplikacija

Kako bi se demonstrirali neki od navedenih algoritama strojnog učenja i preporučiteljskih sustava kreirana je aplikacija koja pomaže korisnicima prilikom prodaje i kupnje automobila.

Za pomoć pri prodaji automobila koriste se tehnike strojnog učenja za preporučivanje cijene tog oglasa. Nakon što unese sve značajke svojeg automobila u formu, korisniku se prikaže preporučena cijena dobivena iz modela strojnog učenja koji je treniran na svim ostalim oglasima.

Za pomoć kod kupnje automobila aplikacija koristi tehnike preporučiteljskih sustava i strojnog učenja. Korištenjem algoritma preporučiteljskih sustava dobiva se funkcionalnost da prilikom pregledavanja oglasa aplikacija predlaže i prikazuje slične oglase.

Također prilikom registracije korisnik upisuje svoje preference te mu se onda pružaju personalizirane preporuke.

Prilikom pregledavanja oglasa uz stvarnu cijenu prikazuje se i preporučena cijena za taj oglas koja se ponovno računa pomoću modela strojnog učenja. Također pokazuje se i postotak odstupanja stvarne i preporučene cijene.

3.1. Tehnologije

Aplikacija se sastoji od baze podataka te odvojenog poslužiteljskog i klijentskog servera. Korištene tehnologije su: PostgreSQL, FastAPI i Flutter. U nastavku su ukratko objašnjene navedene tehnologije i zašto su odabrane za ovaj projekt.

3.1.1. PostgreSQL

PostgreSQL je moćan, otvoreni objektno-relacijski sustav baza podataka koji koristi i proširuje SQL jezik u kombinaciji s mnogim značajkama koje sigurno pohranjuju i skaliraju najkompliciranije operacije s podacima. PostgreSQL-a nastaje 1986. godinu kao dio POSTGRES projekta na Sveučilištu Kalifornija u Berkeleyju i ima više od 35 godina aktivnog razvoja na osnovnoj platformi.

PostgreSQL radi na svim glavnim operativnim sustavima, ACID-kompatibilan je od 2001. godine i ima moćne dodatke poput popularnog PostGIS proširenja za geoprostorne baze podataka. [6]

Jedan od presudnih faktora za izbor PostgreSQL-a je njegova velika zajednica korisnika i bogat ekosustav dodataka i alata. Uz sve navedeno, PostgreSQL je besplatan za korištenje, što ga čini ekonomičnim izborom za projekte svih veličina.

Za pristup PostgreSQL koristila se besplatna aplikacija pgAdmin 4 koji služi kao GUI sučelje i olakšava pristup podacima.

3.1.2. FastAPI

FastAPI je moderan i brz web okvir za izgradnju API-ja s Pythonom 3.6+ temeljeno na standardnim Python tipovima. Razvio ga je Sebastián Ramírez, a stekao je popularnost zbog svoje jednostavnosti, brzine i robusnosti. FastAPI koristi asinkrone funkcionalnosti Pythona, omogućavajući rukovanje velikim brojem istovremenih zahtjeva, što ga čini idealnim za izgradnju visoko performansnih aplikacija.

Jedna od ključnih prednosti FastAPI-ja je automatska generacija OpenAPI i JSON Schema specifikacija, što olakšava dokumentiranje i testiranje API-ja. Uz to, FastAPI integrira Pydantic za validaciju podataka, osiguravajući da su svi podaci koji ulaze u aplikaciju valjani i u ispravnom formatu. Ovo znatno smanjuje mogućnost grešaka i poboljšava stabilnost aplikacija.

FastAPI je također vrlo prilagodljiv i proširiv, omogućavajući integraciju s raznim bibliotekama i alatima kao na primjer SQLAlchemy za rad s bazama podataka koji je korišten u ovoj aplikaciji za pristup PostgreSQL. Također su se koristili Python paketi scikit-learn za obradu podataka i modele strojnog učenja, paket Pandas za analizu i manipulaciju seta podataka i paket Pickle za spremanje i učitavanje modela.

3.1.3. Flutter

Flutter je otvorena platforma za razvoj aplikacija koju je razvio Google, omogućavajući izradu nativnih aplikacija za mobilne uređaje, web i desktop s jednom bazom koda. Flutter koristi Dart programski jezik i pruža bogat skup unaprijed definiranih *widgeta* koji pojednostavljuju razvoj privlačnih korisničkih sučelja. Ovi *widgeti* su visoko prilagodljivi i

omogućuju kreiranje složenih i responzivnih dizajna koji se mogu prilagoditi različitim veličinama zaslona i platformama.

Jedna od najistaknutijih značajki Fluttera je njegova arhitektura koja omogućava hot-reload, značajku koja omogućava programerima da odmah vide promjene u kodu bez potrebe za ponovnim pokretanjem aplikacije. Ovo znatno ubrzava razvojni proces i poboljšava produktivnost jer omogućava brze iteracije i testiranje.

Flutter se ističe svojom performansnom optimizacijom, omogućavajući aplikacijama da rade glatko i brzo na različitim platformama. Uz to, Flutter podržava napredne animacije i prijelaze, što omogućava stvaranje vizualno atraktivnih i dinamičnih korisničkih iskustava. Zahvaljujući svojoj modularnoj strukturi, Flutter omogućava ponovnu upotrebu koda i jednostavnu integraciju s različitim API-jevima i servisima, čineći ga fleksibilnim rješenjem za različite projekte.

Zajednica oko Fluttera je vrlo aktivna, pružajući puno resursa, biblioteka i dodataka koji olakšavaju razvoj i proširuju funkcionalnosti platforme. Googleova kontinuirana podrška i redovita ažuriranja dodatno osiguravaju da Flutter ostane moderna i relevantna tehnologija za razvoj aplikacija.

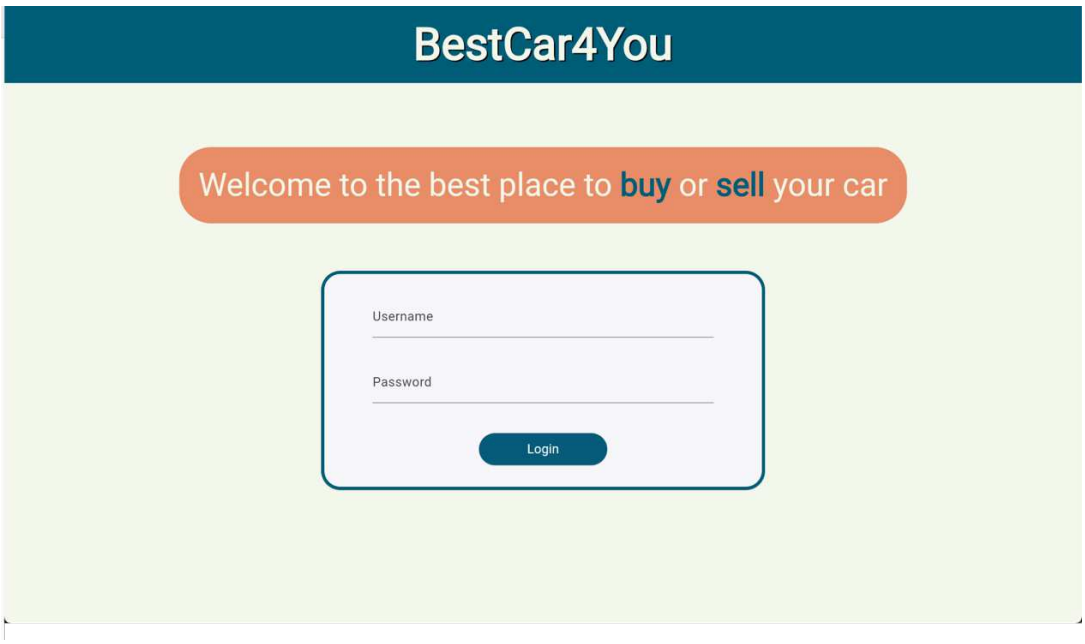
U aplikaciji korištena je i ekstenzija za Flutter **GetX** koja pojednostavljuje upravljanje stanjem, rute i ovisnosti u aplikaciji. Razvijena kao lagana i visoko performansna biblioteka, GetX omogućava programerima da učinkovito organiziraju kod bez potrebe za šablonskim kodom. GetX pruža intuitivne API-je za reaktivno upravljanje stanjem, omogućujući automatsko ažuriranje sučelja kada se podaci promijene. Osim toga, GetX olakšava upravljanje rutama i navigaciju u aplikaciji, pružajući jednostavan način za definiranje i rukovanje različitim ekranima i njihovim prijelazima. Integracija s GetX-om značajno poboljšava produktivnost i održivost Flutter aplikacija, čineći ga popularnim izborom među Flutter programerima.

3.2. Sučelje

Aplikacija je napravljena kao web stranica kako bi ju korisnici mogli najlakše naći i koristiti.

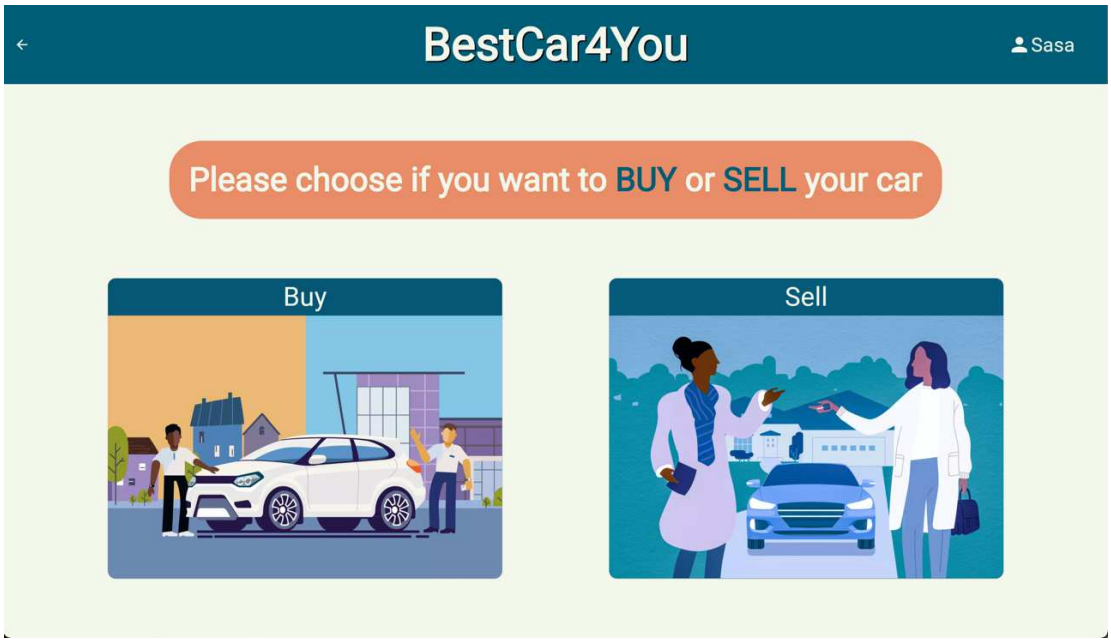
Slijedi pregled sučelja aplikacije i njene funkcionalnosti, a u sljedećim poglavljima detaljno su objašnjene korištene metode kojima je to postignuto.

Dolaskom na aplikaciju korisnika dočeka stranica za prijavu gdje mogu unijeti svoje korisničke podatke koja je prikazana na slici (Slika 3.1).



Slika 3.1 Stranica za prijavu

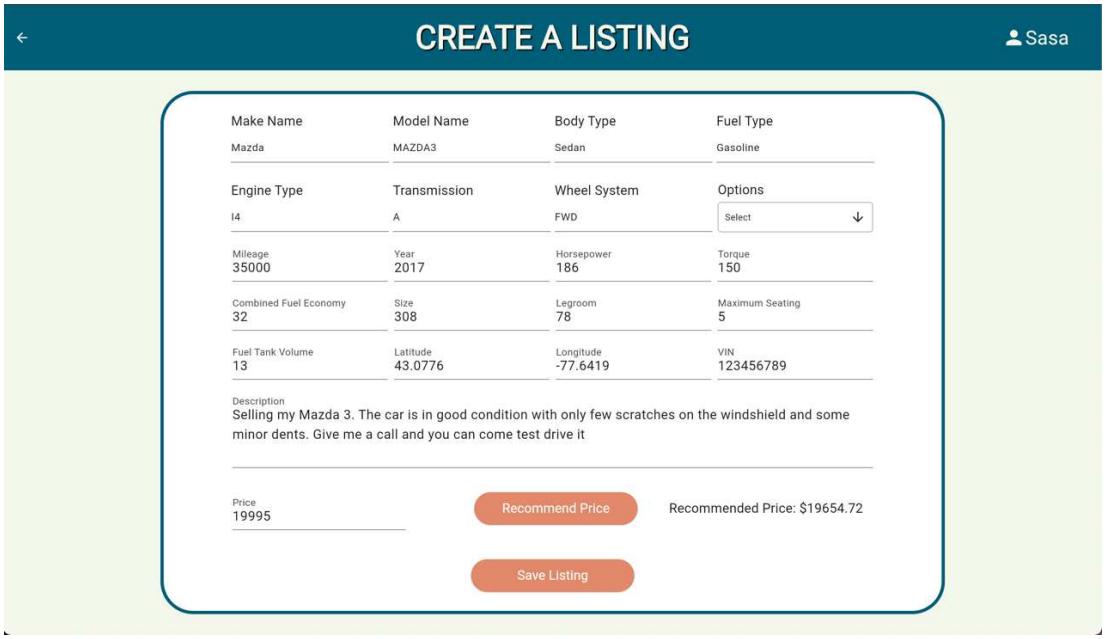
Nakon uspješne prijave prikazuje se početna stranica. U gornjem desnom kutu se prikazuje korisničko ime korisnika. Korisnik može izabrati želi li objaviti oglas za svoj automobil ili pregledavati tuđe oglase. Početna stranica je prikazana na slici (Slika 3.2).



Slika 3.2 Početna stranica

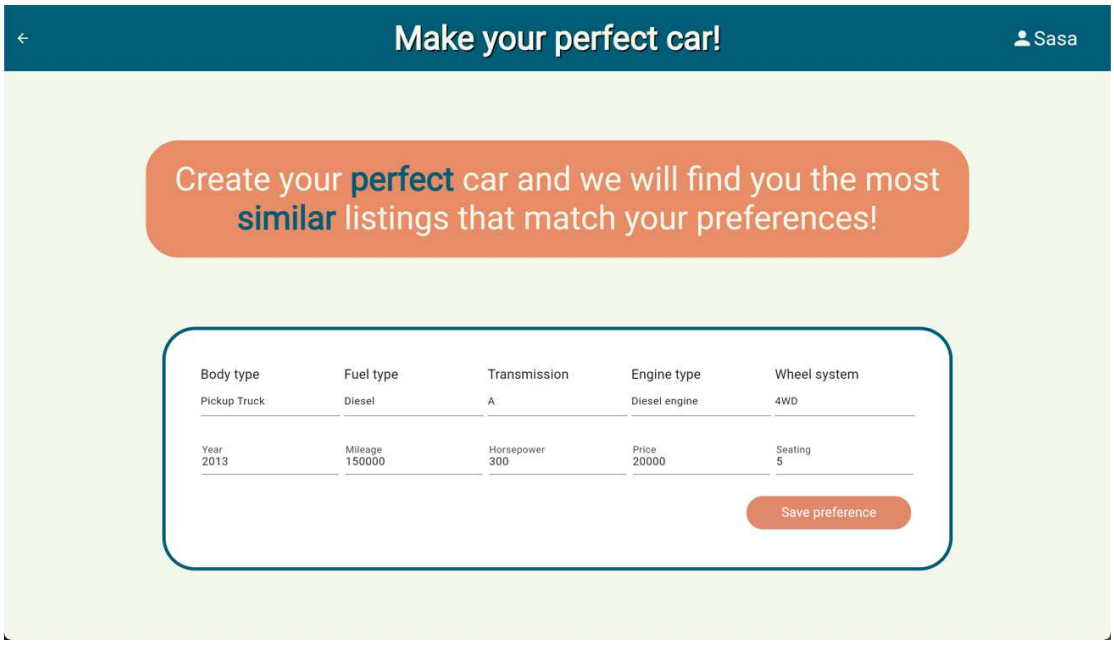
Ako korisnik odabere opciju za kreiranja oglasa otvara mu se forma za unos podataka o automobilu. Za diskretne atribute dohvaćaju se sve moguće vrijednosti s poslužitelja te korisnik bira opciju s padajućeg izbornika. Za odabir dodatnih opcija automobila otvara se dijalog gdje se može označiti više opcija.

Klikom na gumb „Recommend Price“ korisnik dobiva predloženu cijenu za taj automobil, u ovom slučaju to je 19654.72 dolara. Korisnik ne mora staviti tu cijenu već upisuje željenu u tekstualno polje i klikom na gumb „Save Listing“ taj se oglas sprema. Stranica za kreiranja oglasa prikazana je na slici (Slika 3.3).



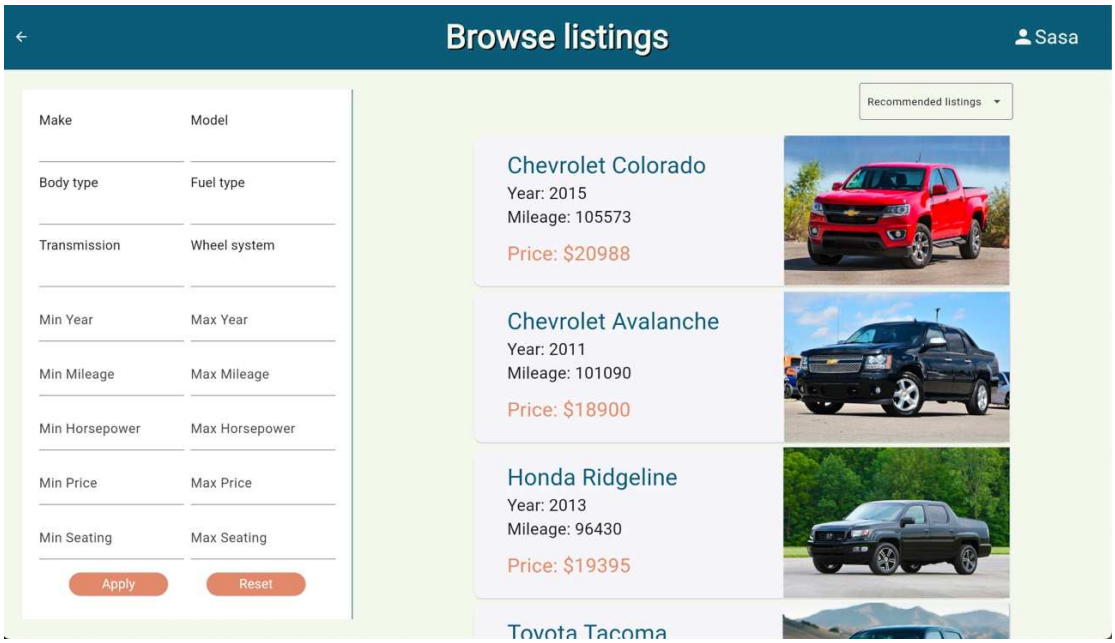
Slika 3.3 Stranica za kreiranje oglasa

Ako korisnik na početnoj stranici odabere opciju za pregledavanje drugih oglasa onda aplikacija prvo provjerava ima li trenutno prijavljeni korisnik spremljene preferencije i ako nema otvara mu se stranica gdje ih može unijeti tako da kreira svoj idealni auto skupa sa cijenom. Na slici (Slika 3.4) se mogu vidjeti preferencije za testnog korisnika „Sasa“.



Slika 3.4 Stranica za ispunjavanje preferencija

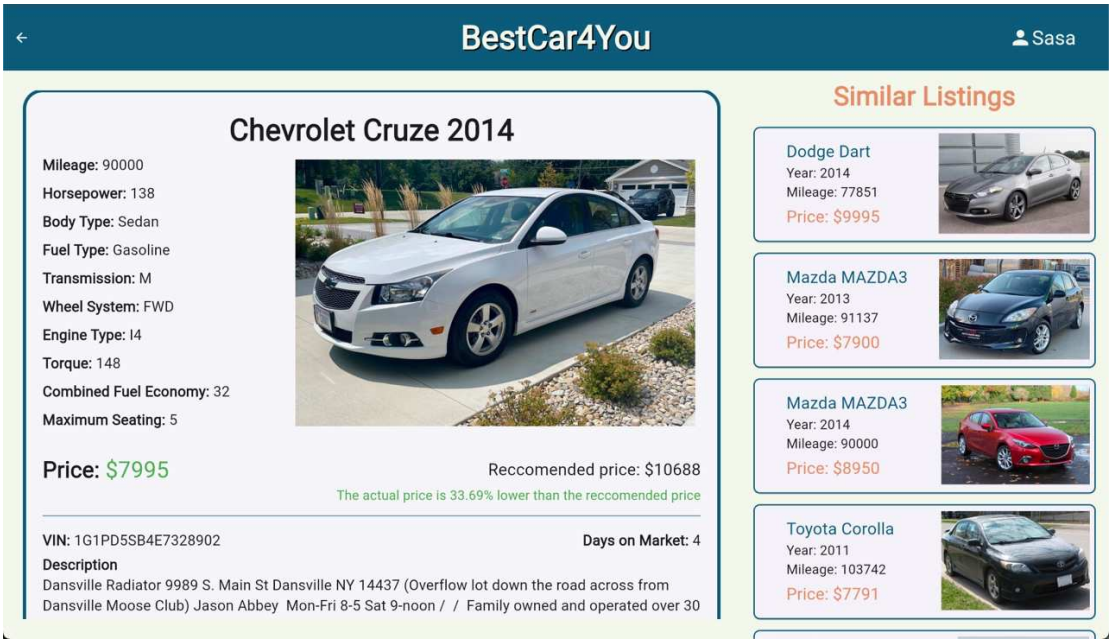
Sa spremljenim preferencijama korisnik može pregledavati oglase. Na desnoj strani nalazi se padajući izbornik gdje korisnik odabire želi li pregledavati sve oglase ili personalizirane oglase. Na slici (Slika 3.5) se prikazuju preporučeni oglasi na temelju preferencija sa slike (Slika 3.4). Na lijevoj strani korisnik ima opciju filtriranja oglasa po jedanaest atributa što mu olakšava pretraživanje.



Slika 3.5 Pretraživanje oglasa

Klikom na neki oglas otvara se njegov detaljni prikaz koji je prikazan na slici (Slika 3.6). Na lijevoj strani su prikazane sve informacije o tom oglasu, a s desne strane je lista sličnih oglasa

koji su poredani silazno po sličnosti. Također prikazuje se preporučena cijena za taj automobil, a ispod nje postotna razlika od stvarne cijene. Ako je stvarna cijena niža onda je tekst u zelenoj boji i ako je viša tekst je u crvenoj boji.



Slika 3.6 Detaljni pregled oglasa

4. Primjene preporučiteljskih sustava u aplikaciji

4.1. Preporuke sličnih oglasa

Prilikom pregledavanja oglasa korisniku se preporučuju slični oglasi s pomoću tehnika preporučiteljskih sustava baziranih na sadržaju. Kako bi se to ostvarilo treba se izračunati sličnost između svih parova oglasa. Zbog količine podataka i broj računskih operacija ovo nije moguće računati u realnom vremenu već se mora periodički osvježivati pozadinskim procesima i spremati izračune u bazu podataka.

4.1.1. Odabir pristupa

Jedan pristup je da se za svaki oglas izračuna sličnosti sa svim drugim oglasima i spremi se samo lista najsličnijih oglasa koji će se kasnije prikazivati. Na ovaj način osiguravamo minimalno trošenje memorije, ali ima i nekih nedostataka. Kao prvo ne može se dobiti proizvoljan broj najsličnijih oglasa već samo onoliko oglasa koliko je spremljeno. Isto tako kod ažuriranja liste najsličnijih oglasa trebalo bi se ponovno računati sličnost sa svim oglasima ili na neki način pratiti koji oglasi su obrađeni, a koji ne. Također za svaki par oglasa bi se sličnost računala dva puta, po jedanput za svaki oglas.

Iz tog razloga odabran je drugi pristup gdje se spremaju sve izračunate sličnosti u bazu. Tako se rješavaju svi nedostaci prošlog pristupa s cijenom veće potrebne memorije. Isto tako filtrira se lista oglasa tako da se računaju sličnosti samo između parova oglasa na kojima je ista vrsta automobila i različiti model vozila kako bi se smanjio broj parova za koji se računa sličnost.

4.1.2. Implementacija

Algoritam za računanje sličnosti skupa sa pomoćnim funkcijama implementiran je u klasi `RecommendationSystem`. Za računanje sličnost između oglasa potrebno je istrenirati TF-IDF model i model za skaliranje značajki. Ti modeli se treniraju tako da se prvo učitaju svi oglasi i onda se mogu koristiti gotove funkcije iz Python paketa skckit-learn i kao rezultat se dobiju gotovi modeli.

Oni se onda spremaju u bazu ili u obliku datoteke na računalu. Također ako postoje već istrenirani modeli mogu se direktno učitati također iz baze ili iz datoteke na računalu.

```
def fit_tfidf_and_scaler(self):
        descriptions = [listing.description if
        listing.description is not None else "" for listing
        in self.listings]
        self.tfidf_matrix =
        self.tfidf_vectorizer.fit_transform(descriptions)

        numerical_data = []
        for listing in self.listings:

            numerical_data.append([getattr(listing, feature)
            for feature in self.numerical_features])

                  numerical_features_df =
                  pd.DataFrame(numerical_data,
                  columns=self.numerical_features)
                  self.scaler.fit(numerical_features_df)
```

Kod 1 Treniranje TF-IDF i modela za skaliranje

Sa svim učitanim oglasima i istreniranim modelima algoritam može krenuti računati sličnosti. Algoritam prolazi kroz sve oglase i sastoji se od četiri dijela.

U prvom dijelu se pronalaze svi kandidati oglasa koji će se uspoređivati s trenutnim oglasom. Iz baze se dohvaća lista oglasa koji već imaju izračunatu sličnost s trenutnim oglasom kako se ne bi više puta računala sličnost između istog para.

```
def get_existing_similarity_listing_ids(self, listing_id):
        result = self.db.query(models.Similarity).filter(
                  (models.Similarity.id1 == listing_id) |
                  (models.Similarity.id2 == listing_id)
                  ).all()
        existing_ids = set()
        for row in result:
            if row.id1 == listing_id:
                existing_ids.add(row.id2)
            else:
                existing_ids.add(row.id1)
        return existing_ids
```

Kod 2 Dohvat oglasa za koje je već izračunata sličnost

Lista oglasa se filtrira tako da se uzmu samo oglasi koji imaju isti tip automobila i različitu marku i model automobila.

```
def update_similarity_scores(self):
      for listing in self.listings:
            existing_ids =
            self.get_existing_similarity_listing_ids
            (listing.listing_id)
            unsaved_pairs = [l for l in self.listings if
            l.body_type == listing.body_type and
            l.make_name != listing.make_name and
            l.model_name != listing.model_name and
            l.listing_id not in existing_ids]
      #...
```

Kod 3 Filtriranje oglasa koji će se uspoređivati

Nakon pronađenih svih parova potrebno je pred procesirati oglase. U ovoj fazi dohvaća se TF-IDF vektor za taj oglas i numeričke značajke se skaliraju. Prednost značajki skaliranja je ta što olakšava interpretaciju modela. Kada su značajke na različitim ljestvicama, postaje izazovno usporediti njihove koeficijente i odrediti njihovu relativnu važnost. Skaliranje značajki na uobičajenu ljestvicu omogućuje nam izravnu usporedbu koeficijenata i procjenu utjecaja svake značajke na regresijski model. [20]

Na konkretnom primjeru to bi značilo da dajemo veću važnost kilometraži od godine modela zato jer je kod kilometraža puno veća razlika nego kod godina.

```
def preprocess_listing(self, listing):
        if listing.description is not None:
            tfidf_vector =
            self.tfidf_matrix[self.listing_id_to_index
            [listing.listing_id]].toarray()[0]
        else:
            tfidf_vector =
            np.zeros(self.tfidf_vectorizer.max_features)

        numerical = [getattr(listing, feature) for feature in
        self.numerical_features]
        numerical_df = pd.DataFrame([numerical],
        columns=self.numerical_features)
        normalized_numerical_features =
        self.scaler.transform(numerical_df)[0]
```

```
        return normalized_numerical_features, tfidf_vector
```

Kod 4 Predprocesiranje oglasa

Sličnost između oglasa se računa tako da se zbrajaju sličnosti pojedinih atributa pomnoženih s njihovim težinama. Težine pojedinih atributa su odabrane proizvoljno, a nakon puštanja sustava u upotrebu te težine se mogu pomoću analiza rada sustava optimizirati. Na primjer može se pratiti na koje preporučene oglase korisnik klikne, a na koje ne i iz zbog kojeg razloga.

Na tablici (Table 1) su prikazane težine numeričkih atributa dok su na tablici (Table 2) prikazane težine diskretnih atributa. Sličnost opisa je postavljeno na težinu 0.06.

Table 1 Težine dodijeljene numeričkim atributima

cijena	godište	kilometraža	snaga motora	potrošnja goriva	broj sjedala	prostor za noge	veličina vozila	okretni moment	spremnik za gorivo
0.15	0.13	0.13	0.1	0.08	0.06	0.03	0.03	0.02	0.01

Table 2 Težine diskretnih atributa

mjenjač	gorivo	pogon	tip motora
0.15	0.13	0.13	0.1

Za računanje sličnosti između dva TF-IDF vektora koristi se mjera kosinus sličnosti.

```
if np.any(tfidf_1) and np.any(tfidf_2):
      tfidf_similarity = np.dot(tfidf_1, tfidf_2) /
      (np.linalg.norm(tfidf_1) * np.linalg.norm(tfidf_2))
      similarity_score = self.coefficients['description'] *
      tfidf_similarity
```

Kod 5 Računanje sličnosti opisa

Za numeričke atribute koristi se formula (4). [7]

$$sličnost = 1 - \frac{|a-b|}{\max(|a|,|b|)} \tag{4}$$

Mogla se je i kod numeričkih atributa koristiti mjera kosinus sličnosti ili neka druga mjera za uspoređivanje vektora, ali na taj način se ne mogu namjestiti pojedinačne težine pojedinog

atributa. Ova formula je odabrana jer daje vrijednosti između 0 i 1 i jer je proporcionalna pa za veće vrijednosti mora biti veća razlika u vrijednostima da bi sličnost bila ista.

```
for i, key in enumerate(numerical_1):
      similarity = 1 - abs(numerical_1[i] - numerical_2[i]) /
      max(abs(numerical_1[i]), abs(numerical_2[i]), 1)
      if similarity > 0:
            similarity_score += self.coefficients[list(
      self.coefficients.keys())[i+1]] * similarity
```

Kod 6 Računanje sličnosti između numeričkih atributa

Za diskretne atribute ukoliko su isti dodaje se iznos težine na finalnu sličnost, a ako su različite dodaje se nula.

```
for key in self.coefficients.keys():
      if key in self. categorical_features:
            similarity = 1.0 if getattr(listing_1, key) ==
            getattr(listing_2, key)
      else 0.0
      if similarity > 0:
            similarity_score += self.coefficients[key] *
            similarity

return similarity_score
```

Kod 7 Računanje sličnosti diskretnih atributa

Važno je napomenuti da su mjere sličnosti za svaki atribut u rangu između nule i jedinice i također zbroj svih težina je jedan tako da će i završna sličnost biti između nule i jedinice.

Zadnji korak je spremanje id-jeva i mjere sličnosti između dva oglasa u bazu podataka kako bi je kasnije mogli dohvatiti.

U klasi `RecommenderSystem` uz funkciju za dohvat proizvoljno mnogo najsličnijih oglasa određenog oglasa postoje još i funkcije za računanje personaliziranih preporuka za korisnike.

4.1.3. Rezultati

Zbog količine potrebnih izračuna i vremena izvršavanja algoritam se testirao na uzorku od 50 000 oglasa. U tablici (Table 3) su prikazana dva najsličnija i dva najrazličitija oglasa od prvog oglasa za Chevrolet Cruze.

Table 3 Tablica sličnosti između oglasa

sličnost	-	0.837	0.834	0.068	0.072
model	Chevrolet Cruze	Dodge Dart	Mazda 3	Cadilac XTS	Dodge Charger
tip automobila	sedan	sedan	sedan	sedan	sedan
cijena	7995	9995	7900	57539	43370
kilometraža	90000	77851	91137	0	0
godište	2014	2014	2013	2019	2020
snaga motora	138	160	155	304	370
mjenjač	ručni	ručni	ručni	automatski	automatski
potrošnja	32 mpg	28.5 mpg	33 mpg	21.5 mpg	20.5 mpg
broj sjedala	5	5	5	5	5
gorivo	benzin	benzin	benzin	benzin	benzin
pogon	prednji	prednji	prednji	svi kotači	stražnji
prostor za noge	77.5 in	77.4 in	78.2 in	82.1 in	81.9 in
veličina	309.8 in	313.6 in	307.9 in	333.2 in	341.3 in
motor	I4	I4	I4	V6	V8
okretni moment	148	148	148	264	395
spremnik goriva	15.6	14.2	14.5	20	18.5

4.2. Personalizirane preporuke za korisnika

Prilikom pregledavanja oglasa korisniku se nudi opcija za prikaz svih oglasa ili preporučenih oglasa na temelju njegovih preferencija, a oni su sortirani po sličnosti.

4.2.1. Implementacija

Za prikaz personaliziranih preporuka potrebno je riješiti cold-start problem. Nakon prijave korisnika provjerava se ima li taj korisnik spremljene preferencije, ako nema korisniku se prikazuje forma u kojoj može opisati svoje idealno vozilo što uključuje cijenu, kilometražu i ostale karakteristike samog automobila koje se onda spremaju u tablicu korisnika.

Personalizirane preporuke nisu dostupne odmah već se računaju pozadinskim procesima, a u međuvremenu korisnik može pregledavati sve oglase. Preporuke se računaju tako da se pretvore u virtualni oglas koji se onda uspoređuje s ostalim oglasima, računa sličnost te sprema u bazu podataka. Oglasi se također filtriraju tako da se samo uspoređuju oni s istim tipom automobila.

Za dohvatiti preporuke koristi se funkcija `get_user_recommendations` koja prima dva parametra: korisničko ime i broj oglasa te vraća oglase sortirane po sličnosti.

```
def get_user_recommendations(self, username, n):
      user =
      self.db.query(models.User).filter(models.User.username
      == username).first()
      if not user:
            return []

      recommended_listings = (
        self.db.query(models.Listing)
        .join(models.UserListingSimilarity,
        models.Listing.listing_id ==
        models.UserListingSimilarity.listing_id)
        .filter (models.UserListingSimilarity.username ==
        username)
        .order_by
        (models.UserListingSimilarity.similarity.desc())
        .limit(n)
        .all())

        return recommended_listings
```

Kod 8 Dohvat personaliziranih preporuka za korisnika

4.2.2. Rezultati

Za provjeru rada algoritma napravljen je testni korisnik te njegove preference se mogu vidjeti na tablici (Table 4) skupa sa dva najsličnija oglasa.

Table 4 Tablica s preporukama za testnog korisnika "Sasa"

sličnost	-	0.525	0.516
model	-	Chevrolet Colorado	Chevrolet Avalanche
tip automobila	kamionet	kamionet	kamionet
cijena	20 000	20 988	18 900
kilometraža	150 000	105 573	101 090
godište	2013	2015	2011
snaga motora	300	305	320
mjenjač	automatski	automatski	automatski
broj sjedala	5	5	5
gorivo	dizel	benzin	benzin
pogon	svi kotači	svi kotači	svi kotači
motor	dizelski motor	V6	V8

5. Primjena strojnog učenja u aplikaciji

U izradi aplikacije korišteni su algoritmi strojnog učenja kako bi se mogla na temelju podataka o vozilu preporučiti cijena. Pomoću toga može se korisniku koji stavlja oglas preporučiti cijenu za njegovo vozilo i također korisniku koji pretražuje oglase prikazati koliko se cijena na oglasu razlikuje od preporučene za vozilo s tim karakteristikama. Ovaj problem spada pod nadzirano učenje zato što se model uči na podacima koje sadrže ulazne i izlazne varijable odnosno karakteristike vozila i njihovu cijenu.

Kako se računa kontinuirana varijabla potrebno je koristiti regresiju. Za pronalazak optimalnog modela za učenje potrebno je prvo očistiti i pripremiti podatke, istrenirati nekoliko modela na dijelu podataka za učenje te ih onda usporediti na dijelu podataka za testiranje. Obradu podataka rađena je u Jupyter bilježnicama koristeći Python pakete. Skup podataka se sastoji od 66 stupaca i 3 milijuna zapisa. Za učitavanje i manipulacijom podataka koristio se paket „Pandas“.

5.1. Čišćenje podataka

Najprije je potrebno provjeriti podatke i očistiti kako bi ih model mogao koristiti. Čišćenje podataka se sastoji od nekoliko koraka: brisanje duplikata, popunjavanja null vrijednosti i brisanje nepotpunih ili nepotrebnih stupaca.

Duplikate je lako pronaći provjerom podataka te se ispostavilo da ima 40 zapisa koji se pojavljuju više puta tako da ih je bilo potrebno izbrisati.

5.1.1. Sređivanje null vrijednosti

Većina modela strojnog učenja ne može raditi s null vrijednostima te ih onda moramo na neki način izbaciti. To se može napraviti tako da se potpuno izbriše cijeli zapis koji sadrži null vrijednosti ili da se popune s nekom vrijednosti. Za popunjavanje vrijednosti potrebno je znati strukturu podataka. Deset stupaca je imalo preko 50 % praznih vrijednosti pa se oni mogu jednostavno izbrisati. Ostalo je još 38 stupaca koji sadrže prazne vrijednosti koje se mogu riješiti na nekoliko načina: postavljanjem najčešće vrijednosti, postavljanjem srednje vrijednosti, postavljanjem srednje vrijednosti po atributu, dodavanjem novih vrijednosti i hibridni pristup.

Postavljanje najčešće vrijednosti se koristi kod diskretnih varijabli i nad ovim podacima je bila najbolji izbor za atribut fuel_type jer 87 % vozila iz podataka koristi benzin kao gorivo pa se može pretpostaviti da će i vozila bez ispunjene vrijednosti koristiti benzin.

Postavljanje srednje vrijednosti se koristi kod kontinuiranih varijabli, ali mora se paziti na distribuciju podataka kako ne bi neki ekstremi puno utjecali na srednju vrijednost. Ova metoda je primijenjena nad stupcem front_legroom koji sadrži podatak o veličini prednjeg prostora za noge.

Postavljanjem srednje vrijednosti po nekom atributu se također koristi kod kontinuiranih varijabli, ali može biti bolji izbor za one atribute koji više ili manje ovise o nekim drugim atributima. Tako na primjer za atribut length, odnosno dužinu vozila, može se staviti prosječna vrijednost za taj tip vozila jer su to dvije ovisne varijable.

Dodavanje novih vrijednosti se koristi kod diskretnih varijabli kada ne postoji neka prevladavajuća vrijednost pa može biti teško odlučiti kojom vrijednosti zamijeniti prazne podatke. Taj problem se javlja kod atributa engine_type jer postoji veliki broj različitih motora. Iz tog razloga dodane su tri nove vrijednosti „Gasoline engine“, „Diesel engine“ i „Electric motor“ koje su dodijeljene ovisno o tipu goriva koje vozilo koristi.

Kod nekih atributa može se koristiti i **hibridni pristup** kao na za fuel_tank_volume, odnosno veličina spremnika za gorivo. Za električna vozila taj atribut nije primjenjiv pa su za njih prazne vrijednosti popunjene s 0, dok su ostale popunjene s prosječnom vrijednošću.

5.1.2. Odabir stupaca

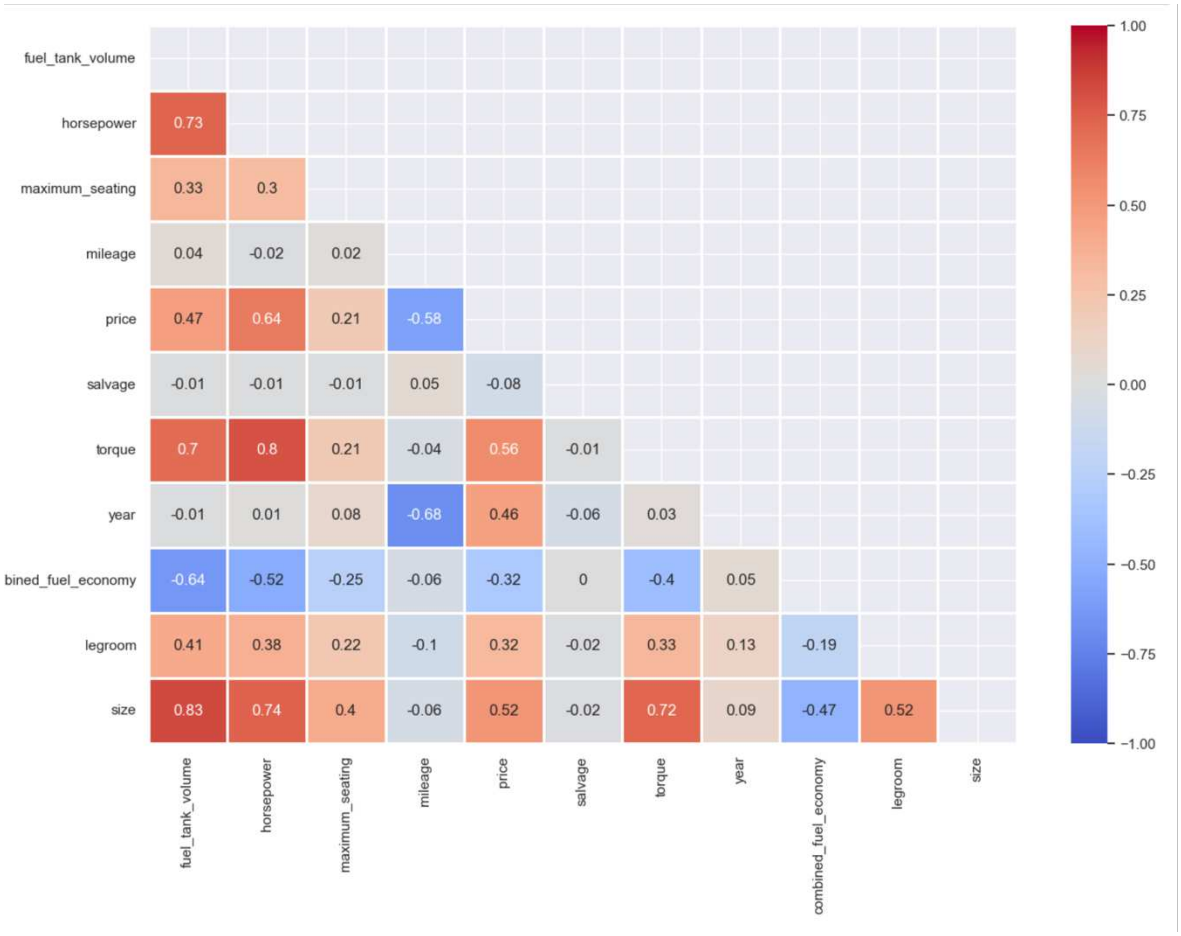
Skup podataka sadrži 66 stupaca, ali neki nisu potrebni za svrhu aplikacije kao na primjer poštanski broj trgovca. Također ima nekih stupaca koji su redundantni poput wheel_system i wheel_system_display gdje je jedan samo skraćena verzija drugoga za vrstu pogona. Nakon eliminacije nepotrebnih stupaca ostao je samo 21 stupac.

5.2. Priprema podataka

Nakon čišćenja skupa podataka ostaje pripremiti podatke za treniranje modela. Ovu pripremu je potrebno odraditi svaki put prije ponovnog treniranja modela. Najprije treba izbrisati značajke koje se neće uzimati u obzir kod procjene cijene kao na primjer: broj dana koliko je oglas aktivan, koordinate, datum objave oglasa, itd...

5.2.1. Spajanje značajki

Multikolinearnost je snažna linearna ovisnost dvije ili više značajki. Modeli strojnog učenja ne rade dobro s multikolinearnosti te može dovesti do neželjenih rezultata. Kako bi se to izbjeglo mogu se više povezanih značajki spojiti u jednu. Prvo se mogu spojiti očite ovisne značajke kao na primjer potrošnja goriva u gradu i na autocesti tako što se te dvije vrijednosti zbroje i spreme u novu kolonu. Isto se može napraviti za veličinu prednjeg i stražnjeg prostora za noge i za dužinu, širinu i visinu kako bi se dobila jedna vrijednost za veličinu vozila. Ostalo je još provjerit ovisnost ostalih značajki za što je korištena funkciju `corr` iz paketa Pandas. Na slici (Slika 5.1 Tablica) je prikazana ovisnosti za svaki par značajki i traži se visoka apsolutna vrijednost.

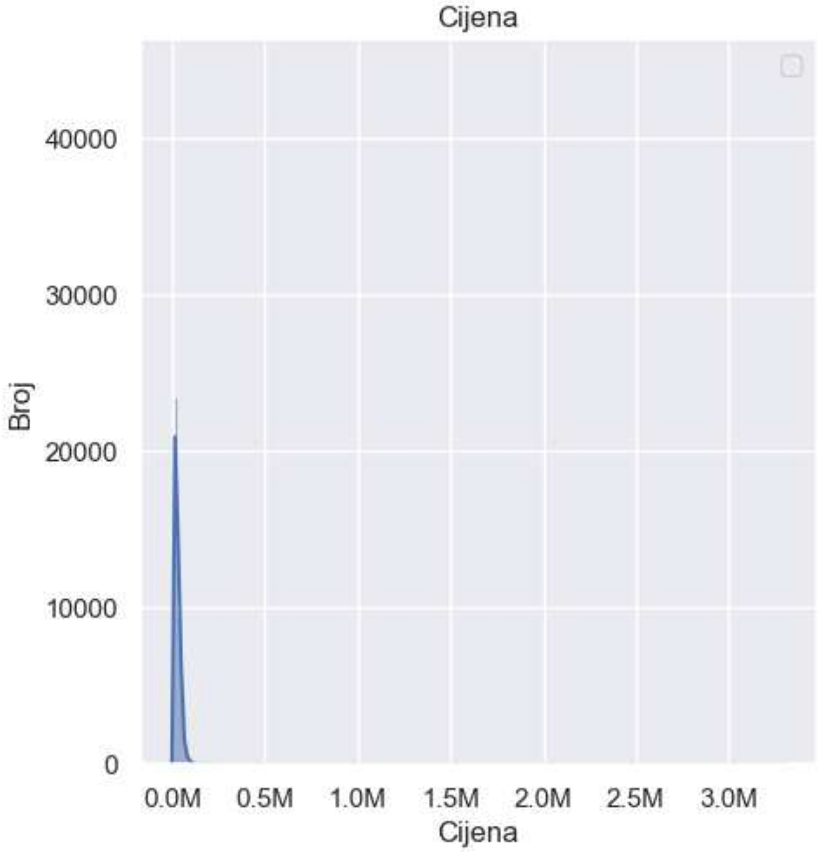


Slika 5.1 Tablica multikolinearnosti

Najvišu apsolutnu ovisnost imaju značajke veličina vozila i veličina spremnika za gorivo. Te dvije značajke se ne mogu spojiti u jednu jer električna vozila imaju veličinu spremnika postavljenu na 0 pa bi nakon spajanja električna vozila ispala manja, a koeficijent od 0.83 je još u prihvatljivim granicama.

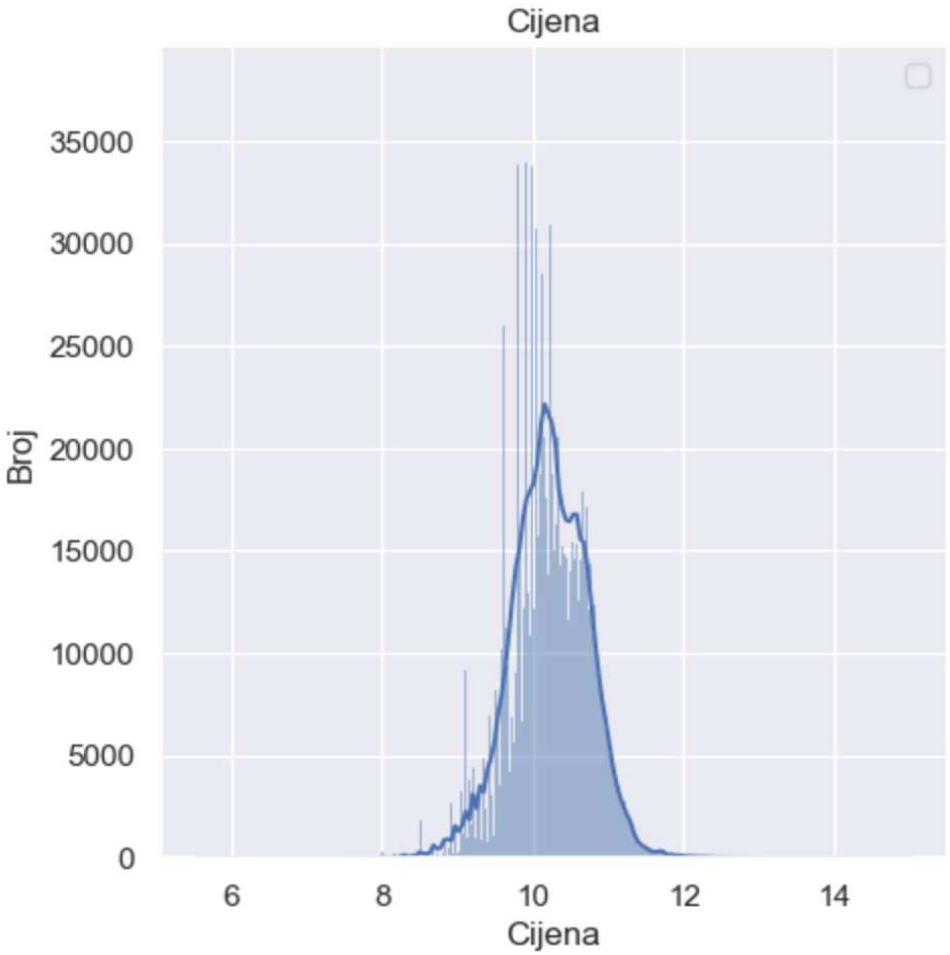
5.2.2. Logaritam cijene

Puno je veći omjer vozila s nižom cijenom nego ovih skupljih što je vizualno prikazano graf na slici (Slika 5.2). Stoga potrebno je normalizirati atribut cijene.



Slika 5.2 Graf cijene prije normalizacije

Kako bi se normalizirala cijena koristiti se logaritam vrijednosti. Na slici (Slika 5.3) može se vidjeti normalizirani graf cijene. Na x osi je vrijednost nakon logaritmiranja cijene u bazi e, dok je na y osi broj automobila sa tom cijenom.



Slika 5.3 Graf cijene nakon normalizacije

5.2.3. Kodiranje diskretnih vrijednosti

One-hot enkodiranje (*eng. one-hot encoding*) je tehnika pretvaranja kategorijskih podataka u numerički format koji se može koristiti u algoritmima strojnog učenja. Ova metoda pretvara kategorijske vrijednosti u binarne vektore gdje svaki vektor ima duljinu jednaku broju jedinstvenih kategorija u originalnom skupu podataka. Svaki vektor sadrži sve nule osim na poziciji koja odgovara kategoriji koju predstavlja, gdje je vrijednost jedan.

One-hot enkodiranje se široko koristi jer mnogi algoritmi strojnog učenja očekuju numeričke ulazne podatke i ne mogu direktno raditi s tekstualnim ili kategorijskim vrijednostima. Tako se osigurava da svaka kategorijska varijabla bude pravilno predstavljena u obliku koji algoritmi mogu obraditi.

Iz tog razloga treba biti oprezan kod kodiranja diskretnih vrijednosti kako se ne bi dobilo previše stupaca i na taj način došlo do problema visoke dimenzionalnosti (*eng. curse of dimensionality*).

Problem visoke dimenzionalnosti odnosi se na činjenicu da s povećanjem broja značajki prostor podataka eksponencijalno raste, što čini podatke sve rjeđima i otežava modelima strojnog učenja da pronađu uzorke i generaliziraju na nove podatke. Kao rezultat, performanse modela često se pogoršavaju jer je potrebna veća količina podataka za pouzdano učenje u visokodimenzionalnim prostorima.

Kako bi se to spriječilo potrebno je izdvojiti one diskretne varijable koje mogu poprimiti veći broj mogućih vrijednosti. U ovom slučaju to su: proizvođač vozila, model vozila i dodatna oprema.

Za proizvođače i modele vozila ostavljeno je samo 20 najčešćih vrijednosti koje se kasnije mogu dalje filtrirati po tome koliko utječu na cijenu.

Kod dodatne opreme problem nije bio toliko trivijalan jer za svako vozilo sprema se lista dodatne opreme koja može sadržati više vrijednosti. Stoga se za svaku opciju dodatne opreme izračunalo koliko puta se pojavljuje na skupu podataka te se ostavilo samo 35 najčešćih vrijednosti. Neke od najčešćih vrijednosti su: kamera za vožnju unatrag, bluetooth, aluminijske felge, grijanje sjedala, sustav za navigaciju, itd…

Nakon kodiranja dobivena je nova tablica sa 125 stupaca. Kako bi se smanjio broj značajki izbrisane su sve one o kojima ne ovisi puno cijena, a to može lako izračunati koristeći funkciju iz biblioteke Pandas. Nakon brisanja svih stupaca koji su imaju faktor ovisnosti s cijenom manji od 0.05 ostalo je 88 značajki.

5.3. Odabir modela

Postoji mnogo različitih modela strojnog učenja i niti jedan nije apsolutno najbolji za bilo koju vrstu problema, već odabir najboljeg modela ovisi o podacima i problemu pa se postavlja pitanje kako uspoređivati više modela. To se radi tako da podijelimo skup podataka na skup za učenje i skup za testiranje. Praksa je da se za učenje modela uzme 70 % - 90 %. Zbog velikog broja podataka u ovom slučaju je odvojeno 75 % podataka za učenje i 25 % za testiranje modela.

Prvo se model trenira na skupu za učenje te se onda predviđa izlazna varijabla za podatke iz skupa za treniranje. Nakon toga možemo raznim metrikama mjeriti odstupanja predviđene vrijednosti od stvarne. Za usporedbu modela strojnog učenja u regresijskim problemima koristi se nekoliko ključnih metrika.

R^2 *(Coefficient of Determination)* mjeri koliko dobro regresijski model objašnjava varijabilnost ovisne varijable, s vrijednostima od 0 do 1, gdje 1 označava savršeno prilagođen model.

MSE *(Mean Squared Error)* je prosječna kvadrirana razlika između stvarnih i predviđenih vrijednosti, gdje manji MSE ukazuje na bolji model.

RMSE *(Root Mean Squared Error)*, koji je kvadratni korijen MSE-a, interpretira se u istim jedinicama kao i stvarne vrijednosti, što olakšava tumačenje pogreške modela. MAPE *(Mean Absolute Percentage Error)* predstavlja prosječni apsolutni postotak pogreške između stvarnih i predviđenih vrijednosti, izražen u postotcima, omogućavajući razumijevanje relativne pogreške modela.

Iako se **Točnost** *(Accuracy)* obično koristi u klasifikacijskim problemima, može se primijeniti i u regresijskim problemima na specifičan način, primjerice tako da se definira prag unutar kojeg se predviđanja smatraju točnima.

Korištenjem ovih metrika zajedno, može se dobiti sveobuhvatna procjena performansi modela.

Za ovaj skup podataka odabrano je pet modela strojnog učenja koje se međusobno uspoređuju ne bi li se našao odgovarajući model. Testirani modeli su: linearna regresija, algoritam slučajnih šuma, LightGBM, XGBoost i umjetna neuronska mreža. Prije treniranja modela linearne regresije i umjetne neuronske mreže potrebno je skalirati značajke.

Skaliranje značajki je važno kod treniranja modela linearne regresije i neuronskih mreža jer pomaže da sve ulazne vrijednosti budu na sličnoj skali. To znači da jedna značajka neće imati više utjecaja zbog većih brojeva. Time modeli brže i bolje uče, jer izbjegavaju probleme s računanjima velikih i malih brojeva istovremeno i na taj način modeli postaju precizniji i pouzdaniji.

Usporedba modela

U tablici (Table 5) su prikazane izračunate mjere za sve testirane modele. Zelenom bojom je označen najbolji rezultat za svaku mjeru.

Table 5 Usporedba modela strojnog učenja

Model	R2	MSE	RMSE	MAPE	Točnost
linearna regresija	0.8401	0.0416	0.2039	1.4021	0.9856
slučajne šume	0.9601	0.0104	0.1019	0.6624	0.9933
LightGBM	0.9331	0.0174	0.1319	0.9302	0.9906
XGBoost	0.9467	0.0139	0.1177	0.8271	0.9917
neuronska mreža	0.9558	0.0115	0.1073	0.7403	0.9926

Nakon testiranja svih modela i usporedbe njihovih rezultata ustanovljeno je da slučajne šume postižu najbolje rezultate prema svim mjerama, što je značajno olakšalo odabir najboljeg modela.

5.4. Implementacija modela

Implementacija modela slučajnih šuma nalazi se u klasi `RandomForestModel`. Klasa sadržava funkcije za pripremu podataka, treniranje modela, spremanje modela u bazu podatka ili kao datoteka na računalu i također učitavanje modela iz baze podataka ili datoteke. Isto tako nalazi se funkcija koja za neki oglas predviđa iznos cijene.

Pozivanjem funkcije `train_model` kreće treniranje modela. Dohvaćaju se svi oglasi iz baze i poziva funkcija za obradu podataka na prijašnje opisan način.

```
def prepare_data_db(self):
        listings = self.get_listings_db()
        df = pd.DataFrame([listing.__dict__ for listing in
      listings])
```

Kod 9 Dohvat oglasa

Brišemo stupce koje ne koristimo za predviđanje cijene

```
df = df.drop(['daysonmarket', 'description', 'latitude',
'listed_date', 'listing_id', 'longitude'], axis=1)
```

Kod 10 Brisanje nepotrebnih stupaca

Ostavljaju se samo dvadeset najčešćih marka i modela automobila dok sve ostale se pretvaraju u 'Other'. Ovo je potrebno kako bi se smanjio broj stupaca nakon kodiranja diskretnih atributa.

```
top_20_make_names = df['make_name'].value_counts()[:20].index
df['make_name'] = df['make_name'].apply(lambda x: x if x in
top_20_make_names else 'Other')

top_20_model_names =
df['model_name'].value_counts()[:20].index
df['model_name'] = df['model_name'].apply(lambda x: x if x in
top_20_model_names else 'Other')
```

Kod 11 Filtriranje marki i modela vozila

Slična stvar se radi i kod dodatne opreme automobila gdje se prvo pronalazi 35 najčešće opreme i onda se one ostavljaju dok se druge brišu.

```
df['options'] = df['options'].apply(lambda x: x if
isinstance(x, list) else [])

all_options = [opt for sublist in df['options'] for opt in
sublist]
option_counts = pd.Series(all_options).value_counts()
top_values = option_counts.nlargest(35).index.tolist()

df['options'] = df['options'].apply(lambda x: ', '.join([opt
for opt in x if opt in top_values]))
```

Kod 12 Filtriranje dodatne opreme

Primjenjuje se logaritamska funkcija nad atributom cijena kako bi se normalizirala.

```
df['price'] = np.log(df['price'])
```

Kategorički atributi i atribut dodatna oprema se kodiraju s pomoću one-hot-encoding.

```
cat_col =
df.select_dtypes(include='object').columns.drop('options')
for col in cat_col:
      dummies = pd.get_dummies(df[col], prefix=col,
      drop_first=True)
      df = pd.concat([df, dummies], axis=1)
      df = df.drop(col, axis=1)

options_split = df['options'].str.get_dummies(sep=', ')
```

```
df = pd.concat([df, options_split], axis=1)
df = df.drop('options', axis=1)
```

Kod 13 Kodiranje kategoričkih atributa

Izdvajaju se svi atributi koji imaju koeficijent kolinearnosti s cijenom za manje od 0.05. Ti atributi se onda izbacuju iz seta podataka zato što malo utječu na predviđanje cijene, a model postaje jednostavniji.

```
columns_to_drop = [col for col in df.columns if
df[col].corr(df['price']) < 0.05 and
df[col].corr(df['price']) > -0.05]
df = df.drop(columns_to_drop, axis=1)
```

Kod 14 Brisanje stupaca koji ne daju puno informacija

Dalje koristi se paket DictVectorizer kako bi učinkovito transformirao listu rječnika u numeričku matricu pogodnu za modele strojnog učenja. Ovaj alat automatski enkodira kategoričke varijable, pretvarajući ih u numeričke vrijednosti, čime se pojednostavljuje priprema podataka. Time se osigurava da se svi podaci pravilno reprezentiraju u formatu koji modeli mogu obraditi, povećavajući efikasnost i točnost treniranja modela.

```
dict_data = df.drop('price',axis=1).to_dict(orient='records')
vec = DictVectorizer(sparse=False)
X = vec.fit_transform(dict_data)
y = df['price'].values
```

Kod 15 Pretvaranje kategoričkih varijabli u numeričke vrijednosti

Važno je napomenuti kako se njegov objekt mora negdje spremiti kako bi se kod korištenja modela svi oglasi na isti način kodirali. U ovom slučaju sprema se kao datoteka na računalu. Također se spremaju liste najčešćih marka, modela automobila i dodatne opreme da se ne moraju računati svaki put ponovo.

```
with open('dict_vectorizer.pickle', 'wb') as f:
      pickle.dump(vec, f)

preprocessing_info = {
      'top_20_make_names': top_20_make_names.tolist(),
      'top_20_model_names': top_20_model_names.tolist(),
       'top_values': top_values
}
with open('preprocessing_info.pickle', 'wb') as f:
```

```
pickle.dump(preprocessing_info, f)
```

Kod 16 Spremanje modela

Funkcija na kraju vraća obrađeni skup atributa i vektor cijena

Sa spremnim podacima se može istrenirati model s pomoću klase RandomForestRegressor iz paketa `sklearn.ensemble`.

```
self.model = RandomForestRegressor()
self.model.fit(X, y)
```

Gotov model ponovno spremamo na željeno mjesto te prilikom prvog pokretanja servera on se učitava i sprema u memoriju kako bi se mogao koristiti. Za računanje predikcije oglasa mora se prvo obraditi na isti način kao i set za treniranje.

```
def predict(self, listing: models.ListingDTO):
        data = {
            'make_name': listing.make_name if
            listing.make_name in
            self.preprocessing_info['top_20_make_names'] else
            'Other',
            …
            'year': listing.year,
            'options': ', '.join([opt for opt in
            listing.options if opt in
            self.preprocessing_info['top_values']])
        }
        df = pd.DataFrame([data])
        dict_data = df.to_dict(orient='records')
        X = self.vec.transform(dict_data)
```

Za kraj se poziva funkcija koja računa cijenu, no zbog primjene logaritamske funkcije prilikom treniranja modela, dobivena vrijednost se mora potencirati kako bi se dobila završna vrijednost.

```
log_predictions = self.model.predict(X)
predictions = np.exp(log_predictions)
```

6. Zaključak

U ovom radu istraživali su se preporučiteljski sustavi i modeli strojnog učenja te se istaknula važnost personaliziranih preporuka u suvremenom digitalnom okruženju. Preporučiteljski sustavi igraju bitnu ulogu u poboljšanju korisničkog iskustva na internetu pružajući korisnicima relevantne informacije i preporuke na temelju njihovih preferencija, ponašanja i povijesti interakcija.

Tijekom istraživanja detaljno su razmatrane različite metode preporučiteljskih sustava, uključujući filtriranje temeljeno na sadržaju, suradničko filtriranje, filtriranje temeljeno na znanju i hibridne pristupe. Za svaku od ovih metoda objašnjen je princip rada, kada se koja koristi te je dan primjer iz stvarnog svijeta.

Proučavani su izazovi s kojima se suočavaju preporučiteljski sustavi, poput problema hladnog starta, gdje sustavi imaju poteškoća s generiranjem preporuka za nove korisnike ili proizvode zbog nedostatka povijesnih podataka.

Istražene su različite metode strojnog učenja, uključujući nadzirano, nenadzirano i učenje pojačanjem. Detaljnije je obrađeno nadzirano učenje i model nasumičnih šuma kao jedan od popularnih modela strojnog učenja.

U radu je također izrađena aplikacija koja demonstrira praktičnu primjenu preporučiteljskih sustava. Cilj je bio kroz aplikaciju pokazati kako teorijska saznanja mogu biti implementirana u stvarnom svijetu da bi korisnicima pružili personalizirano iskustvo.

Detaljno su opisani algoritmi preporučiteljskih sustava korišteni u aplikaciji kako bi se omogućile funkcionalnosti preporuka sličnih oglasa prilikom pregledavanja oglasa i personalizirane preporuke za korisnike ovisno o njihovim unesenim preferencijama. Za preporuke sličnih oglasa koristila se metoda filtriranja temeljena na sadržaju, dok se za personalizirane preporuke koristila metoda filtriranja temeljena na znanju.

Također su opisani algoritmi strojnog učenja korišteni u aplikaciji kako bi se ostvarila funkcionalnost prikaza procijenjenog iznosa cijene nekog oglasa te koliko ta cijena odstupa od stvarne cijene. Isto tako prilikom kreiranja oglasa aplikacija predlaže cijenu temeljenu na podacima o automobilu. U tu svrhu korišten je model slučajnih šuma koji spada pod modele nadziranog učenja.

Literatura

[1] Arsenault, M., The Amazon Recommendations Secret to Selling More Online, Rejoiner, (2021, srpanj). Poveznica: https://www.rejoiner.com/resources/amazon-recommendations-secret-selling-online; pristupljeno 15. lipnja 2024.

[2] Classification, Datu. Poveznica: https://da-tu.ca/courses/info/1; pristpljeno 16. lipnja 2024.

[3] Sruthi, E. R., Understand Random Forest Algorithm With Examples, Analytics Vidhya, (2024, lipanj). Poveznica: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#:~:text=Random%20Forest%20is%20a%20widely,both%20classification%20and%20regression%20problems; pristupljeno 18. lipnja 2024.

[4] Žitko, B., Upotreba stabla odlučivanja kod testiranja znanja metodom kviza, Seminarski rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2003.

[5] Gunay, D., Random Forest, Medium, (2023, rujan). Poveznica: https://medium.com/@denizgunay/random-forest-af5bde5d7e1e; pristupljeno 18, lipnja 2024.

[6] About, PosgreSQL. Poveznica: https://www.postgresql.org/about/; pristupljeno 24. svibnja 2024.

[7] Kushi, How to compute similarity between two numbers, StackExchange, (2015, kolovoz). Poveznica: https://math.stackexchange.com/questions/1481401/how-to-compute-similarity-between-two-numbers#:~:text=This%20measure%20can%20be%20based,x%2Dy%7C%2F(x%2By); pristupljeno 10. svibnja 2024.

[8] Gelo, A., Algoritmi strojnog učenja primijenjeni na zadanom skupu podataka, Završni zadatak. Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje, 2023.

[9] Betancourt, Y. i Sergio, I., Use of Text Mining Techniques for Recommender Systems, Universidad de Zaragoza.

[10] Elham Asani, E., Restaurant recommender system based on sentiment analysis, (2021, prosinac). Poveznica: https://www.sciencedirect.com/science/article/pii/S2666827021000578; pristupljeno 13. svibnja 2024.

[11] Osman, N. A., Integrating contextual sentiment analysis in collaborative recommender systems, (2021, ožujak). Poveznica: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0248695#:~:text=Sentiment%20analysis%20is%20relatively%20a,be%20processed%20by%20CF%20algorithm; pristupljeno 29. travnja 2024.

[12] What are today's top recommendation engine algorithms?, Medium, (2020, ožujak). Poveznica: https://itnext.io/what-are-the-top-recommendation-engine-algorithms-used-nowadays-646f588ce639; pristupljeno 2. svibnja 2024.

[13] Shi, W., Recommendation Systems: A Review, (2020, veljača). Poveznica: https://towardsdatascience.com/recommendation-systems-a-review-d4592b6caf4b; pristupljeno 19. svibnja 2024.

[14] Arora, A. , Content Based Recommender using Text Mining, Kaggle, (2018). Poveznica: https://www.kaggle.com/code/ashish95arora/content-based-recommender-using-text-mining/notebook; pristupljeno 29. svibnja 2024.

[15] Payne, M., Recommender Systems For Business - A Gentle Introduction, Width, (2021, studeni). Poveznica: https://www.width.ai/post/recommender-systems-recommendation-systems; pristupljeno 29. svibnja 2024.

[16] Terra, J. , Regression vs. Classification in Machine Learning for Beginners, Simpli learn, (2023, kolovoz). Poveznica: https://www.simplilearn.com/regression-vs-classification-in-machine-learning-article; pristupljeno 17. svibnja 2024.

[17] Sarkar, D., Categorical Data, (2018, siječanj). Poveznica: https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63; pristupljeno 24. travnja 2024.

[18] Foo, F., How to compute the similarity between two text documents?, (2012, siječanj). Poveznica: https://stackoverflow.com/questions/8897593/how-to-compute-the-similarity-between-two-text-documents; pristupljeno 23. travnja 2024.

[19] Maklin, C., TF IDF | TFIDF Python Example, (2019, svibanj). Poveznica: https://towardsdatascience.com/natural-language-processing-feature-engineering-using-tf-idf-e8b9d00e7e76; pristupljeno 23. travnja 2024.

Sažetak

U današnjem digitalnom dobu, internet je postao ogromno skladište informacija koje neprestano raste. Tradicionalne fizičke trgovine ograničene su prostorom za skladištenje, dok online platforme nude nevjerojatnu raznolikost proizvoda, filmova, glazbe i knjiga. Beskonačna ponuda predstavlja izazov i priliku za korisnike. Omogućuje pronalazak specifičnih proizvoda i sadržaja bez obzira na geografsku lokaciju ili lokalnu dostupnost, ali također zahtijeva efikasne alate za filtriranje i preporuke. Preporučiteljski sustavi i modeli strojnog učenja ključni su u takvom kontekstu. Koriste složene algoritme za analizu korisničkog ponašanja, preferencija i povijesti interakcija kako bi predvidjeli što će korisnicima biti zanimljivo ili korisno. Ovi modeli primjenjuju se kako bi poboljšali korisničko iskustvo i nudili personalizirane preporuke u online trgovinama i streaming servisima.

Summary

In today's digital age, the internet has become an enormous repository of information that continues to expand. Traditional physical stores are constrained by storage space, while online platforms offer incredible diversity of products, movies, music, and books. The limitless supply presents both a challenge and an opportunity for users, necessitating efficient tools for filtering and recommendations. Recommender systems and machine learning models play a crucial role in analyzing user behavior and preferences to enhance user experience with personalized recommendations in online stores and streaming services.