Sustav za preporučivanje smještaja u sklopu sjedišta turističke agencije

Muža, Domagoj

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

DOMAGOJ MUŽA

Sustav za preporučivanje smještaja u sklopu sjedišta turističke agencije

Diplomski rad

Pula, Rujan, 2023. godine

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

DOMAGOJ MUŽA

Sustav za preporučivanje smještaja u sklopu sjedišta turističke agencije

Diplomski rad

JMBAG: 0303075978, redoviti student

Studijski smjer: Informatika

Kolegij: Izrada informatičkih projekata

Mentor: doc. dr. sc. Nikola Tanković

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani ***Domagoj Muža***, kandidat za magistra ***Informatike*** ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega
necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

U Puli, 09. rujna 2023. godine

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, ***Domagoj Muža*** dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom ***Sustav za preporučivanje smještaja u sklopu sjedišta turističke agencije*** koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 09. rujna 2023. godine

Potpis

Sadržaj

1. Uvod

U današnjem svijetu bogatom informacijama, preplavljeni smo izborom, bilo da se radi o filmovima, glazbi, proizvodima ili sadržaju. Ovo mnoštvo opcija nas često ostavlja neodlučnima i nesigurnima što sljedeće odabrati. Ovdje leži ključna uloga sustava preporuka. Sustavi preporuka su moćni alati napravljeni kao pomoć prilikom odabira nudeći personalizirane prijedloge s obzirom na potrebe korisnika.

Sustavi preporuka su računalni algoritmi koji analiziraju i predviđaju koje će stavke ili sadržaj korisnik vjerojatno smatrati vrijednima na temelju prošlih interakcija, ponašanja i stavova. Ove interakcije mogu obuhvatiti širok raspon domena, uključujući e-trgovinu, društvene mreže, novinske članke i sl. Učinkovitim korištenjem podataka koje generiraju korisnici, sustavi preporuka pomažu smanjiti razliku između preopterećenosti informacijama i zadovoljstva korisnika.

Neizbježna su komponenta mnogih aplikacija i web stranica. Bilo da kupujemo online, otkrivamo novu glazbu ili odabiremo što sljedeće gledati, sustavi preporuka rade u pozadini kako bi poboljšali naše iskustvo i kao posljedica toga povećavaju prihode tvrtkama.

Turizam je također jedno od poslovanja gdje sustavi mogu pomoći, to jest preporučiti alternativni smještaj u slučaju da je onaj koji korisnika zanima zauzet ili mu dati preporuku na osnovu njegovih prijašnjih rezervacija, uz pretpostavku da ih ima.

Cilj ovoga rada je izrada jednog takvog sustava u svrhu pomoći turističkoj agenciji, odnosno izgradnja sustava koji preporučuje smještaj za korisnika ili pronalazi slične smještaje traženom.

2. Sustavi preporuke

Sustavi preporuka predstavljaju skupinu algoritama čija je zadaća predložiti artikle korisnicima (artikli mogu biti filmovi, knjige, proizvodi za kupnju ili bilo što, drugo ovisno o industriji) [1] [3] [4].

Cilj sustava za preporuku je preporučiti stavke korisnicima ovisno o njihovim prijašnjim interesima, interakciji s proizvodom i sl. Oni zadržavaju korisnike zainteresiranima za ono što stranica nudi. Također, neki od sustava pružaju personalizirano korisničko iskustvo, pomažući pojedinom kupcu da pronađe filmove, TV emisije, digitalne proizvode, knjige, članke, usluge i ostale slične proizvode. Poduzećima pomažu pri povećanju prodaje, a najbitnije je to da potrošači imaju veliku korist od stranica koje koriste sustave preporuka. Amazon koji je jedan od najvećih web trgovina svojim korisnicima neprestano putem sustava preporuka predlaže nove predmete [1].

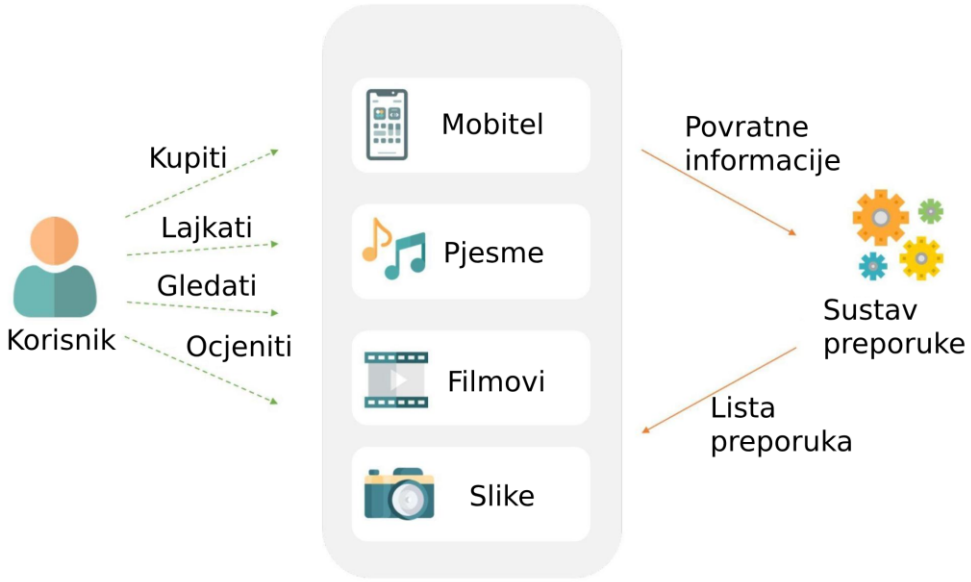
Korisnici se većinu puta suočavaju s problemima pri navigaciji i pronalaženju proizvoda za kupnju, stoga im sustavi preporuka pomažu da lakše pronađu artikle, te im također pružaju jednostavnost korištenja i potiču ih da nastave koristiti stranicu umjesto da odu s nje [3].

Velik rast informacija na internetu i porast e-usluga korisnicima pruža ogroman izbor, koji često dovodi do težeg donošenja odluka. Sustavi preporuka prvenstveno su osmišljeni da pomognu pojedincima koji nemaju dovoljno iskustva ili znanja da se nose s nizom izbora koji im se nude, stoga koriste nekoliko izvora informacija kako bi predvidjeli koje stavke bi korisniku bile zanimljivije [4].

Ovi sustavi prvi su put primijenjeni u e-trgovini za rješavanje problema preopterećenosti informacijama, a brzo su prošireni na personalizaciju e-uprave, e-poslovanja, e-učenja i e-turizma. Oni su danas neizostavni na stranicama kao što su Amazon, YouTube, Netflix, Yahoo i Facebook.
Njihova glavna funkcija je ponuditi preporuke korisnicima (npr. Amazon ili Netflix) ili raditi u pozadini kako bi odabrali sadržaj koji će se sljedeći prikazati korisniku (npr. YouTube).

Bitni su za određene vrste poslovanja jer mogu izložiti korisnika sadržaju koji možda inače ne bi pronašao ili zadržati korisnika dulje nego što bi ostao. Iako izgradnja sustava može biti prilično jednostavna, pravi je izazov izgraditi sustav koji funkcionira i u kojem tvrtka vidi stvarni napredak i vrijednost iz svojih rezultata [6].

Sustavi se mogu izgraditi korištenjem različitih tehnika, od jednostavnijih (npr. onih koji su temeljeni samo na drugim ocijenjenim stavkama istog korisnika) do iznimno složenih. Složeni sustavi preporuka koriste niz različitih izvora podataka (jedan izazov je korištenje nestrukturiranih podataka, posebno slika, kao ulazne podatke) i tehnika strojnog učenja (uključujući duboko učenje). Stoga su vrlo prikladni za umjetnu inteligenciju, točnije za učenje bez nadzora. Kako korisnici nastavljaju pregledavati sadržaj i pružati više podataka, ti se sustavi mogu izgraditi za pružanje sve boljih preporuka [5].



Slika 1. Tijek korištenja sustava preporuka

Prilagođeno prema: https://www.mdpi.com/2076-3417/10/16/5510/htm

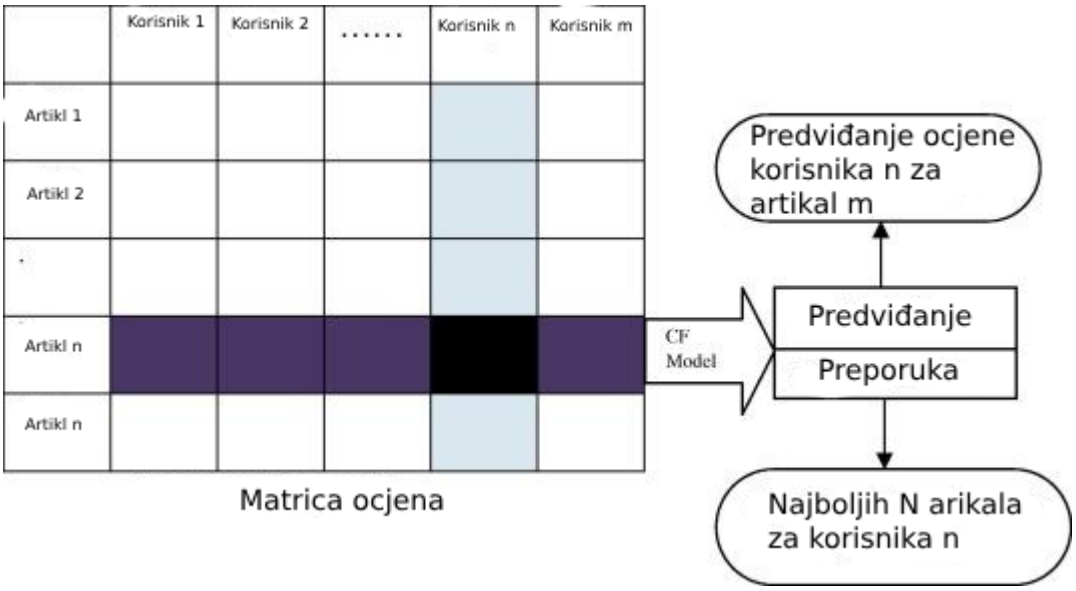
Proizvoditelj sustava	**Cilj sustava**
Amazon.com	Knjige i ostali proizvodi
Netflix	Serije, Filmovi, itd.
Jester	Vicevi
GroupLense	Vijesti
MovieLense	Filmovi
last.fm	Glazba
Google News	Vijesti
Google Search	Oglasi
Facebook	“Prijatelji”, Oglasi
Pandora	Glazba
YouTube	Videa
Tripadvisor	Turistički proizvodi
IMDb	Filmovi

Tablica 1: Najpopularniji sustavi preporuka i njihovi ciljevi

Izvor [1]

Dva su glavna modela na koje se može definirati problem preporuke:

1. Predviđanje ocjene
2. Rangiranje



Slika 2. Proces kolaborativnog filtriranja,
Prilagođeno prema: https://www.researchgate.net/figure/Collaborative-filtering-process_fig4_321808484

Predviđanje ocjene

Pod pretpostavkom da skupljeni podaci prikazuju što se korisniku sviđa kod pojedine stavke ili koje stavke mu se sviđaju. Za m korisnika i n stavki, ovo odgovara nepotpunoj $m * n$ matrici, gdje se navedene (ili opažene) vrijednosti koriste za obuku. Vrijednosti koje nedostaju (ili se ne promatraju) se predviđaju tijekom obuke. Ovaj problem također se naziva i problem popunjavanja matrice jer imamo nepotpuno specificiranu matricu vrijednosti, a preostale vrijednosti su predviđene tijekom učenja modela [3].

Rangiranje

U praksi nije potrebno predvidjeti ocjene korisnika za određene proizvode kako bi dali preporuke korisnicima. Npr. možda se želi preporučiti najbolje proizvode za određenog korisnika ili odrediti potencijalne korisnike za određeni proizvod. Određivanje najboljih stavki je uobičajenije od određivanja potencijalnih korisnika, iako su obje metode potpuno analogne [6].

U slučaju rangiranja, vrijednosti predviđenih ocjena nisu važne. Prva opcija je općenitija jer se iz nje može izvesti rješenje za drugu opciju rješavanjem prve opcije za različite kombinacije korisnik-stavka i zatim ocijeniti predviđanja. Međutim, u

mnogim je slučajevima lakše i prirodnije osmisliti metode za izravno rješavanje problema.

Iako je primarni cilj sustava preporuka povećanje prihoda za tvrtku, to se često postiže na načine koji su manje očiti nego što se na prvi pogled čini [1].
Kako bi se postigao poslovni cilj povećanja prihoda, zajednički operativni i tehnički ciljevi sustava preporuka su sljedeći:

- Relevantnost: najočitiji cilj sustava preporuka je preporučiti stavke koje su relevantne korisniku. Veća je vjerojatnost da će korisnici pregledavati stavke koje im se čine zanimljive. Iako je relevantnost glavni cilj sustava preporuka, ona nije dovoljna sama po sebi [6].

- Novosti: sustavi za preporuku su korisni kada je preporučena stavka nešto s čime se korisnik do sada nije susreo. Na primjer, popularni filmovi određenog žanra rijetko bi bili novi za korisnika. Ponovljena preporuka popularnih artikala također može dovesti do smanjenja raznolikosti [4].

- Slučajnost: preporučene stavke su donekle neočekivane, pa stoga postoji skroman element slučajnosti, za razliku od očitih preporuka. Slučajnost se razlikuje od novosti po tome što su preporuke doista iznenađujuće za korisnika, a ne jednostavno nešto za što prije nisu znali. Često se može dogoditi da određeni korisnik pregledava samo artikle određene vrste, iako može postojati latentno zanimanje za stavke drugih vrsta što bi i samog korisnika moglo iznenaditi. Za razliku od novosti, slučajne metode usmjerene su na otkrivanje takvih preporuka [5].

- Povećanje raznolikosti preporuka: sustavi preporuka obično predlažu popis najboljih stavki. Kada su sve te preporučene stavke vrlo slične, povećava se rizik da se korisniku možda neće svidjeti niti jedna od tih stavki. S druge strane, kada preporučeni popis sadrži stavke različitih vrsta, veća je vjerojatnost da će se korisniku svidjeti barem jedna od tih stavki. Raznolikost ima prednost jer osigurava da korisniku ne dosadi ponovljena preporuka sličnih artikala [6].

2.1. Vrste podataka

Sustavi preporuka rade sa dvije vrste podataka. Prva vrsta su eksplicitni podaci među kojima je najzastupljenije ocjenjivanje. Druga vrsta su implicitni podaci koji su u rasponu od klika na web stranici do raznih događaja u sustavu (npr. Odgledan video do kraja).

2.1.1. Eksplicitni podaci

Eksplicitni podaci, predstavljaju točnu ocjena koju je korisnik dao proizvodu. Neki od primjera eksplicitnih informacija su ocjene filmova od strane korisnika na Netflixu i ocjene proizvoda od strane korisnika na Amazonu [1].

Eksplicitni podaci uzimaju u obzir unos korisnika o tome kako mu se sviđa ili ne sviđa proizvod i mogu se kvantificirati. Trebaju biti kao u obliku ocjena. Teško ih je prikupiti jer zahtijevaju dodatni unos od korisnika, a potreban je poseban sustav za prikupljanje ove vrste podataka. Zatim treba odlučiti želi li se ići s ocjenama ili opcijom sviđa/ne sviđa mi se za prikupljanje podataka. Svaki ima svoje prednosti i mane [2].

Eksplicitne povratne informacije ne uzimaju u obzir kontekst kada ste gledali artikl.
Postoji još jedan problem s eksplicitnim ocjenama. Ljudi različito ocjenjuju filmove/artikle. Neki su popustljivi u svojim ocjenama, dok su drugi ozbiljni u pogledu ocjena koje daju. Također moramo voditi računa o pristranosti u ocjenama korisnika [1].

2.1.2. Implicitni podaci

Implicitni podaci ne odražavaju izravno interes korisnika, ali djeluju kao njegova zamjena [1].

Primjeri skupova implicitnih povratnih informacija uključuju povijest pregledavanja, klikove na poveznice, broj reprodukcije pjesme, postotak web stranice koji se pregledao - 25%, 50% ili 75% ili čak pomicanje miša [2].

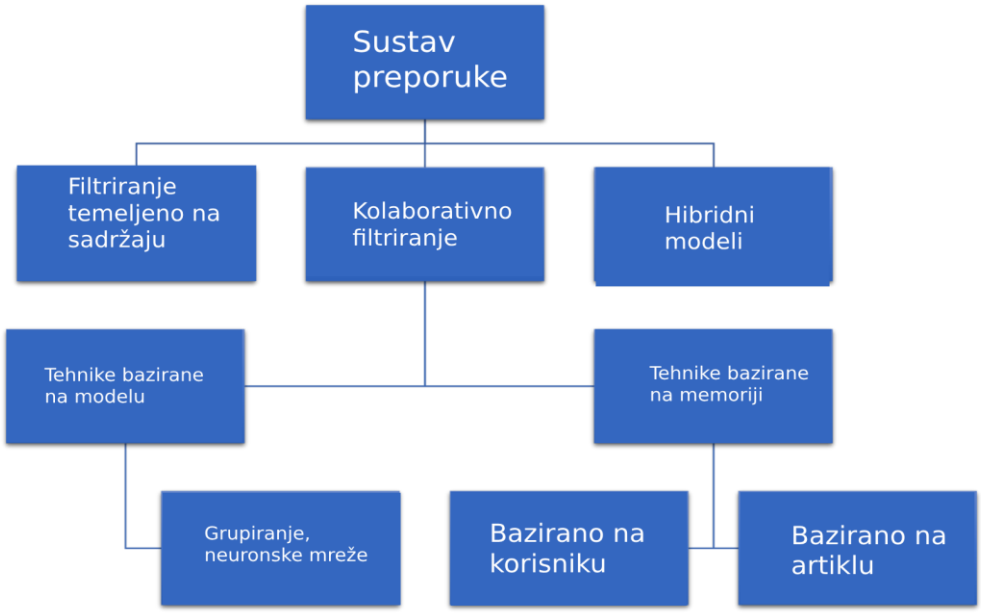
Ako smo upravo pregledali stavku, to ne znači da nam se nužno svidjela, ali ako smo ju pregledali više puta, to nam je bolji pokazatelj zainteresiranosti za tu stavku. Implicitna povratna informacija prikuplja informacije o radnjama korisnika [2].
Implicitnih podataka ima u izobilju i lako ih je prikupiti. Ne trebaju nikakav dodatni unos od korisnika. Nakon što se dobije dopuštenje korisnika za prikupljanje njegovih podataka, ovisnost o korisniku je smanjena [1].

2.2. Vrste sustava preporuke

Osnovni modeli rade s dvije vrste podataka. Prvi sadržavaju ocjene ili ponašanje pri kupnji, dok druga sadrži opise ili relevantne ključne riječi o korisniku ili stavci.
Modeli koji koriste prvu varijantu nazivaju se modeli kolaborativnog filtriranja, dok modeli koje koriste druge nazivaju se modeli temeljenim na sadržaju.
Također, uz prethodno dva navedena modela postoji i hibridan model koji su kombinacija oba modela [1] [3].

Modeli sustava preporuka:

- Kolaborativno filtriranje
- Sustavi temeljeni na sadržaju/znanju
- Hibridni sustavi



Slika 3. Različite vrste sustava preporuka

Prilagođeno prema: https://www.lumenci.com/post/ai-powered-search-and-recommendation-system

2.2.1. Filtriranje temeljeno na sadržaju

Sustavi filtriranja temeljenog na sadržaju odabiru stavke na temelju interakcije između sadržaja artikala i interesa korisnika [7].

S obzirom na važnost značajki artikla u ovom pristupu, važno je razumjeti kako se odlučuje o najbitnijim značajkama korisnika [7].

Ovdje se mogu koristiti dvije metode, koje se potencijalno mogu kombinirati jedna s drugom:

- Korisnici mogu dobiti popis značajki među kojima mogu odabrati ono s čime se najviše poistovjećuju.
- Algoritam može pratiti proizvode koje je korisnik prije odabrao i dodaje te značajke podacima o korisniku.

Slično tome, značajke artikla mogu identificirati sami arhitekti sustava. Štoviše, korisnike se može pitati za koje značajke smatraju da se najviše poistovjećuju s proizvodima [7].

Koraci filtriranja temeljenog na sadržaju [8]:

- Vektorizacija teksta:
    - Brojanje vektora
    - TF-IDF
- Izračun sličnosti

Vektorizacija teksta jedan je od najvažnijih pristupa koji se temelji na obradi teksta, rudarenju teksta i obradi prirodnog jezika. Pristupi kao što je pretvaranje tekstova u vektore i izračunavanje sličnosti-udaljenosti na njima čine osnovu analitičkog svijeta. Ako se tekstovi mogu prikazati vektorima, onda se nad njima mogu izvoditi matematičke operacije [7].
Dva najkorištenija načina za predstavljanje teksta kao vektora su Brojanje vektora i TF-IDF eng. Term Frequency - Inverse Document Frequency.

Svi jedinstveni pojmovi smješteni su u stupce, a svi dokumenti u retke.

	danas	je	lijep	dan	će	biti	sunčan	i
Danas je lijep dan								
Danas će biti sunčan dan								
Danas će biti lijep i sunčan dan								

Slika 4. Dokumenti i raščlanjene svih riječ u dokumentima

Učestalost pojmova u dokumentima nalazi se u ćelijama.

	danas	je	lijep	dan	će	biti	sunčan	i
Danas je lijep dan	1	1	1	1				
Danas će biti sunčan dan	1			1	1	1	1	
Danas će biti lijep i sunčan dan	1		1	1	1	1	1	1

Slika 5. Učestalost pojma u svakom dokumentu

TF-IDF je statistička metoda u obradi prirodnog jezika i traženju informacija. Mjeri koliko je pojam važan u dokumentu u odnosu na zbirku dokumenata. Čini opću standardizaciju vektora riječi, uzimajući u obzir matricu pojmova dokumenta, cijeli korpus, sve dokumente i učestalost pojmova. Tako eliminira neke od pristranosti koje se mogu pojaviti [7].

Prvo se izračuna učestalost svake riječi u svakom dokumentu, a zatim se izračuna učestalost riječi (eng. Term Frequency - TF) čija je formula:

$$TF = \frac{učestalost\ pojma\ t\ u\ dokumentu}{ukupan\ broj\ pojmova\ u\ dokumentu}$$

	danas	je	lijep	dan	će	biti	sunčan	i
Danas je lijep dan	0.25	0.25	0.25	0.25	0	0	0	0
Danas će biti sunčan dan	0.2	0	0	0.2	0.2	0.2	0.2	0
Danas će biti lijep i sunčan dan	0.14	0	0.14	0.14	0.14	0.14	0.14	0.14

Slika 6. TF za svaku riječ u svakom dokumentu

Sljedeće se izračuna inverzna učestalost dokumenta (eng. Inverse Document Frequency - IDF)

$$IDF = log(\frac{broj\ dokumenata}{broj\ dokumenata\ s\ izrazom\ t + 1})$$

danas	je	lijep	dan	će	biti	sunčan	i
0.125	0.602	0.301	0.125	0.301	0.301	0.301	0.125

Slika 7. IDF za svaku riječ

Ako pojam t ima visoku učestalost opažanja u cijelom korpusu, to znači da taj srodan pojam utječe na cijeli korpus. U tom se slučaju vrši normalizacija frekvencija kako unutar pojmova tako i u cijelom korpusu.

Izračuna se TF-IDF $= TF * IDF$

	danas	je	lijep	dan	će	biti	sunčan	i
Danas je lijep dan	0.031	0.151	0.075	0.031	0	0	0	0
Danas će biti sunčan dan	0.025	0	0	0.025	0.06	0.06	0.06	0
Danas će biti lijep i sunčan dan	0.018	0	0.043	0.018	0.043	0.043	0.043	0.018

Slika 8. Izračun TF-IDF-a za svake riječ u svakom dokumentu

Zatim se vrši L2 normalizacija.

Izračuna se korijen zbroja kvadrata redaka i podijele se odgovarajuće ćelije s pronađenom vrijednošću.

Danas je lijep dan	0.174
Danas će biti sunčan dan	0.205
Danas će biti lijep i sunčan dan	0.225

Slika 9. L2 normalizacija za svaki dokument

	danas	je	lijep	dan	će	biti	sunčan	i
Danas je lijep dan	0.178	0.867	0.431	0.178	0	0	0	0
Danas će biti sunčan dan	0.121	0	0	0.121	0.292	0.292	0.292	0
Danas će biti lijep i sunčan dan	0.079	0	0.191	0.079	0.191	0.191	0.191	0.079

Slika 10. TF-IDF nakon normalizacije

L2 normalizacija ponovno ispravlja riječi koje nisu mogle pokazati svoj učinak zbog prisutnosti nedostajućih vrijednosti u nekim redcima.

2.2.2. Kolaborativno filtriranje

Kolaborativno filtriranje je tehnika filtriranja stavki koje bi se korisniku mogle svidjeti na temelju interesa sličnih korisnika [9].

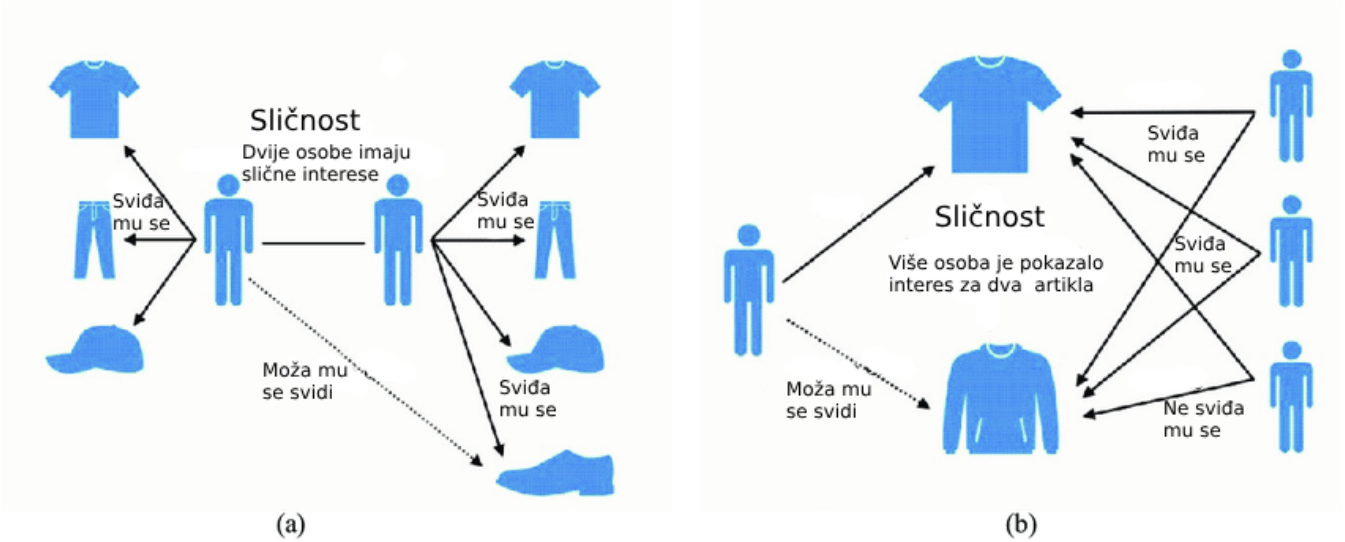
Djeluje tako da pretražuje veliku grupu ljudi i pronalazi manji skup korisnika sa sličnim interesima određenom korisniku. Uzima u obzir stavke koje im se sviđaju i kombinira ih kako bi stvorio rangirani popis prijedloga [10].

Filtrira informacije korištenjem interakcija i podataka koje je sustav prikupio. Temelji se na ideji da će korisnici koji su isto ocijenili iste stavke vjerojatno isto ocijeniti i ostale stavke u budućnosti, odnosno na odnos između korisnika i stavki. Sličnost stavki utvrđuje se sličnošću ocjena tih stavki od strane korisnika koji su ocijenili druge stavke [1].

Koncept je jednostavan: kada želimo pronaći npr. novi film za gledanje, često ćemo pitati naše prijatelje sa sličnim ukusom za preporuke jer imamo više povjerenja u njih od onih preporuka prijatelja koji imaju različite ukuse [12].

Klase kolaborativnog filtriranja:

- Tehnike temeljene na modelima- koje se dalje dijele na:
    - Temeljene na korisniku (mjeri sličnost između korisnika i drugih korisnika).
    - Temeljene na stavci (mjeri sličnost između stavki koje ciljani korisnici ocjenjuju ili s kojima stupaju u interakciju i drugih stavki).



Slika 11. Algoritmi kolaborativnog filtriranja: (a) temeljeni na korisniku, (b) na temelju stavke.

Izvor:https://www.researchgate.net/figure/The-collaborative-filtering-algorithms-a-user-based-b-item-based_fig2_355218515

Za izgradnju sustava koji može automatski preporučiti stavke korisnicima na temelju interesa drugih korisnika, prvi korak je pronaći slične korisnike ili artikle. Drugi korak je predviđanje ocjena stavkama koje korisnik još nije ocijenio. Dakle, trebat će nam odgovori na pitanja [12]:

- Kako odrediti koji su korisnici ili stavke slični jedni drugima?
- S obzirom na to da znamo koji su korisnici slični, kako odrediti ocjenu koju bi korisnik dao artiklu na temelju ocjena sličnih korisnika?
- Kako izmjeriti točnost ocjene koje model izračuna?

Kolaborativno filtriranje je skup algoritama gdje postoji više načina za pronalaženje sličnih korisnika ili stavki i više načina za izračunavanje ocjene na temelju ocjena sličnih korisnika [1].

Važna napomena je da se u pristupu koji se temelji isključivo na kolaborativnom filtriranju, sličnost ne izračunava pomoću faktora kao što su dob korisnika, žanr filma ili bilo koji drugi podaci o korisnicima ili artiklima. Izračunava se samo na temelju ocjene (eksplicitne ili implicitne) koju korisnik daje artiklu. Na primjer, dva se korisnika mogu smatrati sličnima ako daju iste ocjene za deset filmova unatoč velikoj razlici u njihovoj dobi.

Pitanje o tome kako izmjeriti točnost predviđanja također ima višestruke odgovore, koji uključuju tehnike izračuna pogreške koje se mogu koristiti na mnogim mjestima [13]. Jedan od pristupa za mjerenje točnosti rezultata je srednja kvadratna pogreška (RMSE), u kojoj se predviđaju ocjene za testni skup podataka parova korisnik-stavka čije su vrijednosti ocjena već poznate. Razlika između poznate vrijednosti i predviđene vrijednosti bila bi pogreška. Kvadriranjem svih vrijednosti pogreške za skup testova, izračuna se prosjek (ili srednja vrijednost), a zatim se izvuče kvadratni korijen tog prosjeka da bi se dobila tražena vrijednost.

Drugi pristup za mjerenje točnosti je srednja apsolutna pogreška (MAE), u kojoj se veličina pogreške nalazi pronalaženjem njezine apsolutne vrijednosti, a zatim uzimanjem prosjeka svih vrijednosti pogreške.

$$RMSE = \sqrt{\frac{1}{broj\ uzoraka} * \sum_{(i,j)\epsilon R}^{k} (r - \hat{r})^2}$$

$$MAE = \frac{1}{broj\ uzoraka} * \sum_{(i,j)\epsilon R}^{k} (r - \hat{r})$$

2.2.2.1. Filtriranje temeljeno na memoriji

Ključna razlika između filtriranja temeljenog na memoriji i tehnika temeljenih na modelu je u tome što kod pristupa temeljenog na memoriji ne učimo nijedan parametar korištenjem gradijentnog spuštanja (ili bilo kojeg drugog algoritma optimizacije). Najbliži korisnici ili stavke izračunavaju se pomoću jedne od formula za izračun sličnosti, koje se temelje na aritmetičkim operacijama [11].

Kosinusna sličnost - može se promatrati geometrijski ako se redak (stupac) određenog korisnika (stavke) matrice ocjena tretira kao vektor. Za kolaborativno filtriranje temeljeno na korisniku, sličnost dvaju korisnika mjeri se kao kosinus kuta između

vektora dvaju korisnika. Za korisnike u i u′, sličnost po kosinusu je:

$$sim(u,u') = cos(\theta) = \frac{\mathbf{r}_u \cdot \mathbf{r}_{u'}}{\|\mathbf{r}_u\| \|\mathbf{r}_{u'}\|} = \sum_i \frac{r_{ui} r_{u'i}}{\sqrt{\sum_i r_{ui}^2} \sqrt{\sum_i r_{u'i}^2}}$$

Slika 12. Formula za sličnost po kosinusu

Izvor: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

Možemo predvidjeti ocjenu korisnika u za artikla i uzimajući zbroj ocjena artikala i od svih drugih korisnika (u') gdje je težina broj sličnosti između svakog korisnika i korisnika-u.

$$\hat{r}_{ui} = \sum_{u'} sim(u,u') r_{u'i}$$

Slika 13. Formula za izračun predviđene ocjene korisnika i artikl

Izvor: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

Također bismo trebali normalizirati ocjene prema ukupnom broju ocjena drugih korisnika.

$$\hat{r}_{ui} = \frac{\sum_{u'} sim(u,u') r_{u'i}}{\sum_{u'} |sim(u,u')|}$$

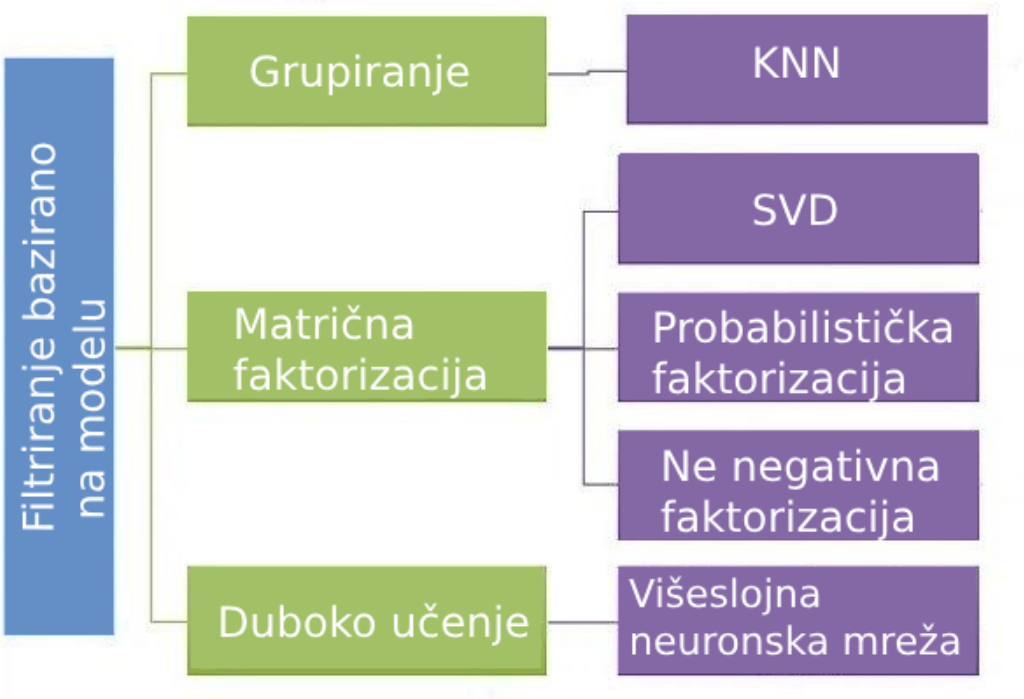
Slika 14. Normalizacija predviđene ocjene

Izvor: https://www.encora.com/insights/recommender-system-series-part-2-neighborhood-based-collaborative-filtering

Budući da nije uključena nikakva obuka ili optimizacija, pristup je jednostavan za korištenje. Ali njegova se točnost smanjuje kada imamo slabo popunjene podatke što narušava skalabilnost ovog pristupa za većinu problema iz stvarnog svijeta.

2.2.2.2. Filtriranje bazirano na modelu

U ovom pristupu, modeli kolaborativnog filtriranja razvijeni su pomoću algoritama strojnog učenja za predviđanje korisnikove ocjene neocijenjenih stavki. Algoritmi u ovom pristupu mogu se podijeliti na tri podvrste [13].



Slika 15. Podjela sustava baziranih na modelu

Prilagođeno prema: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

2.2.2.3. Grupiranje

Algoritam grupiranja (eng. k-nearest neighbors) pretpostavlja da slični artikli postoje u neposrednoj blizini. Drugim riječima, slični artikli su blizu jedna drugoj [13].

Algoritam k-najbližih susjeda, poznat i kao KNN, ne parametarski je klasifikator s nadziranim učenjem, koji koristi blizinu artikala za izradu klasifikacija ili predviđanja o

grupiranju pojedinačne točke. Iako se može koristiti za probleme regresije ili klasifikacije, obično se koristi kao algoritam klasifikacije [13].

Ideja je ista kao kod sustava preporuka temeljenih na memoriji. U algoritmima temeljenim na memoriji koristimo se sličnostima između korisnika i/ili stavki i koristimo ih kao težine za predviđanje ocjene za korisnika i stavku. Razlika je u tome što se sličnosti u ovom pristupu izračunavaju na temelju modela učenja bez nadzora, a ne Pearsonove korelacije ili kosinusne sličnosti. U ovom pristupu također ograničavamo broj sličnih korisnika kao k, što sustav čini skalabilnijim [14].

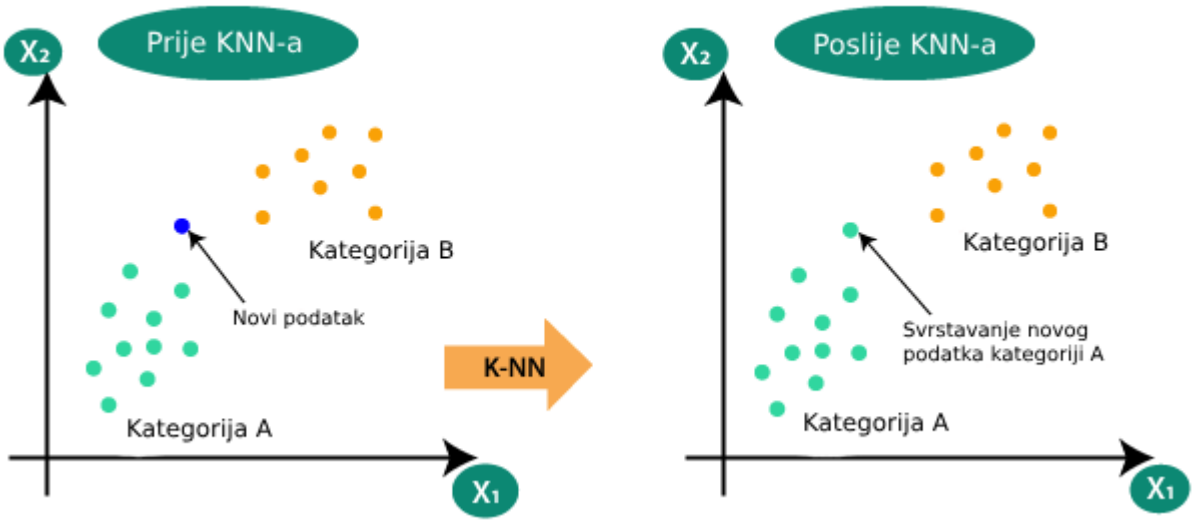
Pohranjuje sve dostupne podatke i klasificira novu točku na temelju sličnosti. To znači da kada se pojave novi podaci, oni se mogu lako klasificirati u kategoriju korištenjem KNN algoritma. Također se naziva i algoritam lijenog učenika jer ne uči iz skupa za obuku, već pohranjuje skup podataka i u vrijeme klasifikacije izvodi radnju na skupu podataka [14].

Da bismo odabrali k koji je pogodan za podatke, pokrećemo KNN algoritam nekoliko puta s različitim vrijednostima k i odabire se k koji smanjuje broj pogrešaka na koje se nailazi, a istovremeno održava sposobnost algoritma da točno predviđa kada mu se daju novi podaci [15].

Za probleme klasifikacije, oznaka klase se dodjeljuje na temelju većine glasova odnosno koristi se oznaka koja je najčešće zastupljena oko dane točke. Dok se to tehnički smatra "glasovanjem većine", u literaturi se češće koristi izraz "glas većine". Razlika između ovih terminologija je u tome što "glas većine" tehnički zahtijeva većinu veću od 50%, što prvenstveno funkcionira kada postoje samo dvije kategorije. Kada imamo više razreda (na primjer četiri kategorije), ne treba nam nužno 50% glasova da bismo donijeli zaključak o razredu, već možemo dodijeliti oznaku klase s glasovima većim od 25% [15].

Regresijski problemi koriste sličan koncept kao problem klasifikacije, ali u ovom slučaju, prosjek k najbližih susjeda se uzima da bi se napravilo predviđanje o klasifikaciji. Glavna razlika ovdje je da se klasifikacija koristi za diskretne vrijednosti, dok se regresija koristi za kontinuirane vrijednosti. Međutim, prije nego što se može

napraviti klasifikacija, udaljenost mora biti definirana. Najčešće se koristi euklidska udaljenost [15].



Slika 16. Primjer procesa grupiranja

Prilagođeno prema: https://patriziacastagnod.medium.com/k-nearest-neighbors-knn-9491f6d684ae

Kako bi se odredilo koje su podatkovne točke najbliže danoj točki upita, morat će se izračunati udaljenost između točke upita i ostalih točaka podataka.

Neke od najbitniji metrika:

Euklidska udaljenost**:** Ovo je najčešće korištena mjera udaljenosti, a ograničena je na vektore stvarnih vrijednosti. Koristeći donju formulu, mjeri ravnu liniju između točke upita i druge točke koja se mjeri [16].

$$d(x,\ y)\ =\ \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Slika 17. Formula za euklidsku udaljenost

Izvor: https://www.geeksforgeeks.org/how-to-calculate-euclidean-distance-in-excel/

Manhattanska udaljenost**:** Još jedna od popularnih metrika za mjerenje udaljenosti, koja mjeri apsolutnu vrijednost između dvije točke. Također se naziva udaljenost taksija ili udaljenost gradskog bloka jer se obično vizualizira pomoću mreže, koja prikazuje kako se može kretati s jedne adrese na drugu gradskim ulicama. To je mjera udaljenosti koja se izračunava uzimanjem zbroja udaljenosti između x i y koordinata [16].

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Slika 18. Formula za Manhattan udaljenost

Izvor: https://medium.com/@tpreethi/introduction-to-k-nearest-neighbors-knn-algorithm-python-implementation-9c387915f31d

Udaljenost Minkowski**:** Ova mjera udaljenosti je generalizirani oblik euklidske i manhattanske metrike udaljenosti. Parametar p u formuli omogućuje stvaranje drugih metrika udaljenosti. Manhattanska udaljenost vrlo dobro funkcionira za visokodimenzionalne skupove podataka. Ne povećava razlike između bilo koje značajke. Također ne zanemaruje nijednu značajku [16].

$$D(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

Slika 19. Formula za Minkowski udaljenost

Izvor: https://walidhadri.medium.com/k-nearest-neighbors-part-1-introduction-a67cfdd601a7

Prednosti grupiranja

- Jednostavan za implementaciju: s obzirom na jednostavnost i točnost algoritma [14].

- Lako se prilagođava: dodavanjem novih uzoraka, algoritam se prilagođava kako bi izračunao sve nove podatke jer se svi podaci o vježbanju pohranjuju u memoriju [14].
- Malo hiperparametara: zahtijeva samo vrijednost k i metriku udaljenosti, što je malo u usporedbi s drugim algoritmima strojnog učenja.

Nedostatci grupiranja

- Ne skalira se dobro - budući da je grupiranje lijen algoritam, zauzima više memorije i pohrane podataka u usporedbi s drugim klasifikatorima. To može biti skupo i iz vremenske i novčane perspektive. Više memorije i prostora za pohranu povećat će poslovne troškove, a računanje više podataka može trajati dulje. Iako su stvorene različite strukture podataka, kao što je Ball-Tree, kako bi se riješila računalna neučinkovitost, drugačiji klasifikator može biti idealan ovisno o poslovnom problemu [14].
- Problem dimenzionalnosti - ne radi dobro s visokodimenzionalnim unosima podataka. Ovo se ponekad naziva i fenomenom vrhunca, gdje nakon što algoritam postigne optimalan broj značajki, dodatne značajke povećavaju količinu grešaka klasifikacije, posebno kada je veličina uzorka manja [14].
- Sklon prenaklonjenosti (eng. overfitting) - zbog "problema dimenzionalnosti", također je skloniji overfitingu. Dok se odabir značajki i tehnike smanjenja količine podataka koriste kako bi se to spriječilo, vrijednost k također može utjecati na ponašanje modela. Niže vrijednosti k mogu pretjerano odgovarati podacima, dok više vrijednosti k imaju tendenciju "izglađivanja" vrijednosti predviđanja budući da se radi o prosjeku vrijednosti na većem području ili susjedstvu [14].

2.2.2.4. Matrična faktorizacija (MF)

Matrična faktorizacija je metoda kolaborativnog filtriranja koja pronalazi odnose između stavki i korisnika. Radi na način da originalnu matricu podjeli u dvije (ili više) matrica čiji umnožak rezultira originalnoj matrici. Matricu R veličine $m * n$ možemo podijeliti u dvije matrice, jedna veličine $m * k$, a druga $n * k$ gdje m i n odgovaraju broju redaka/stupaca originalne matrice, a k broj latentnih faktora koji predstavljaju povezanost između stavki i korisnika [17]. Također, jedna od prednosti matrične faktorizacije je smanjenje memorije potrebne za spremanje podataka sve dok je $k < \min(m, n)$.

Faktorizacija je način za predviđanje matrice kod koje se koristi smanjenje dimenzionalnosti zbog korelacija između stupaca (ili redaka). Većina metoda redukcije dimenzionalnosti također se može izraziti kao matrična faktorizacija [17].
Kao primjer uzmimo slučaj u kojem su svi unosi u matricu ocjena R znani. Ključna ideja je da se bilo koja $m * n$ matrica R uvijek može izraziti pomoću [18]:

$$R = U * V^T$$

Ovdje je U matrica veličine $m * k$, a V je matrica veličine $n * k$. Svaki stupac matrice U može se promatrati kao jedan od k vektora k-dimenzionalnog prostora stupaca matrice R, a $j - ti$ red matrice V sadrži odgovarajuće koeficijente za kombiniranje vektora u $j - ti$ stupac matrice R. Također, stupce matrice V možemo gledati kao vektore redova matrice R, a retke matrice U kao odgovarajuće koeficijente [18].

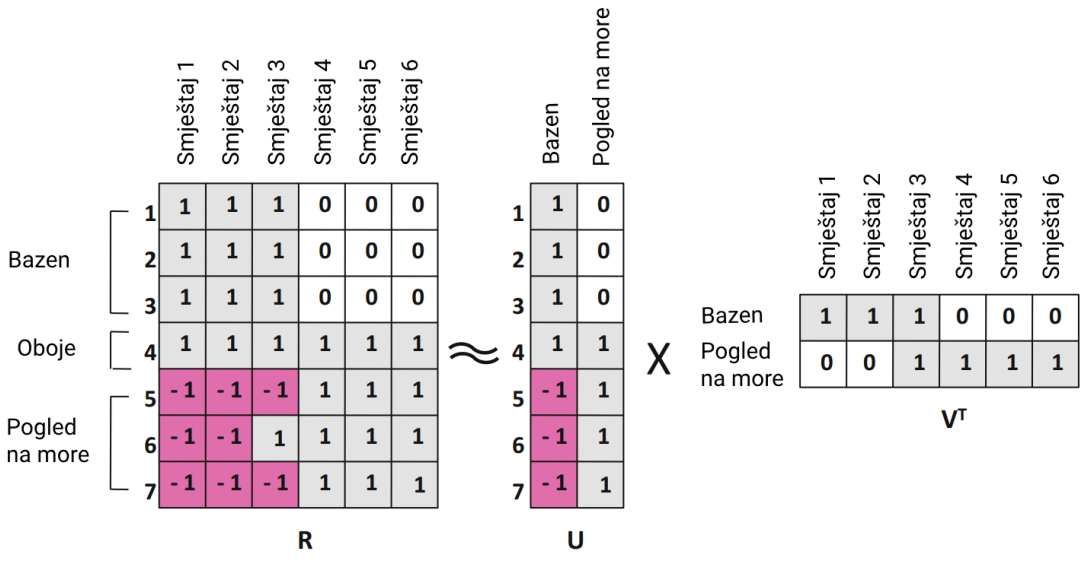
U modelu matrične faktorizacije matrica R veličine $n * m$ se može približno faktorizirati u dvije matrice. Matricu U veličine $m * k$ i matricu V veličine $n * k$ čime dobivamo izraz [18]:

$$R \approx U * V^T$$

U je matrica veličine $m * k$, a V je $n * k$ matrica. Pogreška predviđanja je jednaka $\left|\left| R - U * V^T \right|\right|^2$, gdje $\left|\left| . \right|\right|^2$ predstavlja zbroj kvadrata razlike stvarne cijene i predviđene ocjene čime u konačnici dobivamo matrici razlike $(R - U * V^T)$. Ovo se

također naziva (kvadratnom) Frobeniusova norma matrica razlike. Matrica razlike predstavlja šum odnosno razliku u temeljnoj matrici ocjena [19].

Kako bismo lakše razumjeli matričnu faktorizaciju, proučimo matricu ocjena prikazanu na slici 20. Na njoj je prikazana matrica sa ocjenama veličine 7 × 6, gdje imamo 7 korisnika i 6 stavki. Sve ocjene su u rasponu od $\{-1,0,1\}$, što odgovara sviđanju, ne sviđanju i neutralnosti. Stavke su smještene jedinici i pripadaju kategorijama da li ima bazen, odnosno da li ima pogled na more. Jedna od smještajnih jedinica pod nazivom "Smještaj 3" pripada objema kategorijama. Zbog načina na koji gledamo na kategorije, korisnici također pokazuju naklonjenost svakoj od tih kategorija.



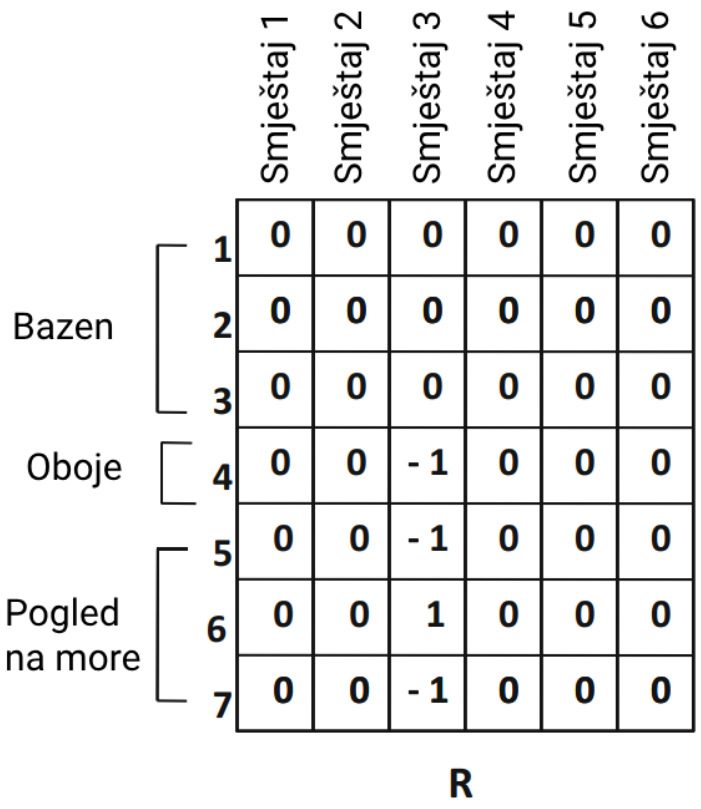
Slika 20. Primjer matrične faktorizacije

Na primjer, korisnici od 1 do 3 preferiraju smještaje koji imaju bazen, ali su neutralni prema smještajima koji imaju pogled na more. Korisnik 4 voli smještaj koji ima oboje. Dok korisnici 5 do 7 vole smještaje koji imaju pogled na more, ali smještaj ne mora imati bazen.

Primijetimo da ova matrica ima značajnu podudarnost između korisnika i smještaja, iako se čini da su ocjene smještaja koji pripada dvjema različitim kategorijama relativno neovisne. Kao rezultat, ova se matrica može približno faktorizirati na faktore ranga 2, odnosno $k = 2$.

Matrica U je veličine 7 × 2, koja pokazuje sklonost korisnika prema prethodno navedenim dvjema kategorijama, dok je matrica V veličine 6 × 2, koja pokazuje pripadnost smještaja u te dvije kategorije. Drugim riječima, matrica U daje osnovu za stupce, dok matrica V daje osnovu za retke matrice R.

Na primjer, kod matrice U vidimo da korisnik 1 voli smještaj s bazenom, dok korisnik 4 voli smještaj s bazenom i pogledom na more. Do sličnog zaključka može se doći pomoću redaka matrice V. Stupci matrice V odgovaraju latentnim vektorima. Odgovarajuća razlikovna matrica prikazana je na slici ispod. Iz razlikovne matrice na slici 21. vidimo da ocjene korisnika za "Smještaj 3" ne prate zadani obrazac.



Slika 21. Razlikovna matrica

U ovom primjeru, matrica R je potpuna, pa stoga faktorizacija nije korisna za predviđanje vrijednosti koje nedostaju. Korisnost pristupa javlja se kada matrica R nije u potpunosti navedena, odnosno imamo vrijednosti koje nedostaju, ali se još uvijek

mogu robusno procijeniti svi latentni faktori matrica U i V. Za niske vrijednosti ranga (vrijednost parametra k), to je još uvijek moguće iz rijetko popunjenih podataka. To je zato što nije potrebna ogromna količina promatranih ocjena za procjenu latentnih faktora. Nakon što su matrice U i V procijenjene, cijela matrica ocjena, to jest matrica R može se procijeniti kao $U * V^T$, što nam daje sve ocjene koje nedostaju.

Svaka ocjena r_{ij} u matrici R može se izraziti kao umnožak faktora i-tog korisnika i faktora j-te stavke [17].

$$r_{ij} \approx u_i * v_j$$

Faktori $u_i = \{u_{i1}, \ldots, u_{ik}\}$ i $v_i = \{v_{i1}, \ldots, v_{ik}\}$ se mogu promatrati kao korisnika za k različitih koncepata, jednostavniji način pisanja formule:

$$\hat{r}_{ij} = \sum_{s=1}^{k} u_{is} * v_{js}$$

$\hat{r}_{ij}$ označava da je to predviđena vrijednost, a r_{ij} označava promatranu vrijednost. Razlika između opažene i predviđene vrijednosti određenog unosa (i, j) je opisana formulom.

$$e_{ij} = (r_{ij} - \widehat{r_{ij}}) = (r_{ij} - \sum_{s=1}^{k} u_{is} * v_{js}).$$

Vrijednost stavke e_{ij} je jednaka razlici između promatrane vrijednosti, to jest vrijednosti koju znamo i predviđene vrijednosti ili drugim riječima razlika promatrane vrijednosti i umnoška i-tog retka matrice U i j-tog stupca matrice V, odnosno matrice V^T.

Zatim funkciju cilja za nepotpune matrice, odnosno matrice u kojima ne znaju sve vrijednosti možemo izračunati samo pomoću elemenata za koje imamo vrijednosti.

$$Min\, J = \sum_{(i,j)\epsilon R, (i,j) > 0}^{k} e_{ij}^2 = \sum_{(i,j)\epsilon R, (i,j) > 0}^{k} (r_{ij} - \widehat{r_{ij}})^2$$

Funkcija cilja J koju želimo minimizirati možemo napisati kao sumu kvadrata razlike promatrane vrijednosti i predviđene vrijednosti uz uvjet da je vrijednost r_{ij} veća od nule.

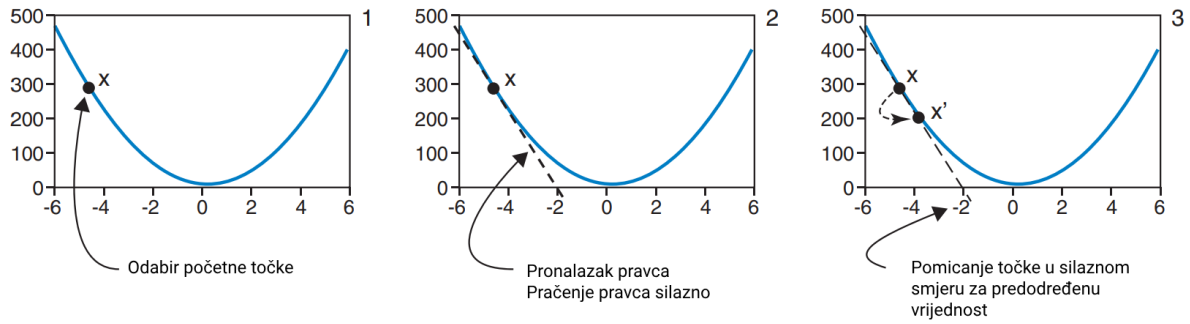
Način da se razlika svede na najmanju moguću vrijednost je korištenje algoritma koji se zove gradijentni spust.
Želimo da jednadžba kažnjava velike pogreške, zato uzimamo kvadrat razlike. No ipak želimo da se pogreška pojavi na istoj ljestvici kao i ocjena pa zbog toga koristimo srednju kvadratnu pogrešku (eng. RMSE).

RMSE možemo napisati kao drugi korijen sume kvadrata razlike promatrane vrijednosti i predviđene vrijednosti podijeljene sa veličinom uzorka.

$$RMSE = \sqrt{\frac{1}{broj\ uzoraka} * \sum_{(i,j)\epsilon R}^{k} (r_{ij} - \widehat{r_{ij}})^2}$$

Gradijentni spust
Gradijentni spust je tehnika pronalaženja optimalnih točaka na grafu, gdje optimalno rješenje može biti najniža ili najviša točka na grafu. Kod gradijentnog spusta potrebno je negdje započeti, a zatim pronaći smjer u kojemu funkcija daje manju vrijednost od trenutne [20].

$$f(x', y') < f(x, y)$$



Slika 22. Koraci kod gradijentnog spusta

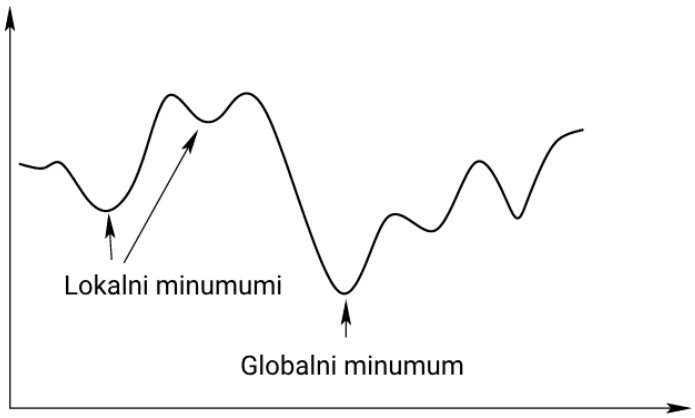
Možemo započeti bilo gdje jer sve funkcije nisu lijepog zdjelastog oblika kao na slici iznad. U većini slučajeva funkcije imaju više od jednog lokalnog minimuma poput onog prikazanog na slici ispod. Često je najbolji prijedlog isprobati različite početne točke i

vidjeti hoće li rezultat biti isti. Da bismo pronašli pravac spusta, pronalazimo derivaciju gradijentne funkcije i zatim sljedeći korak koristeći baznu funkciju [20]:

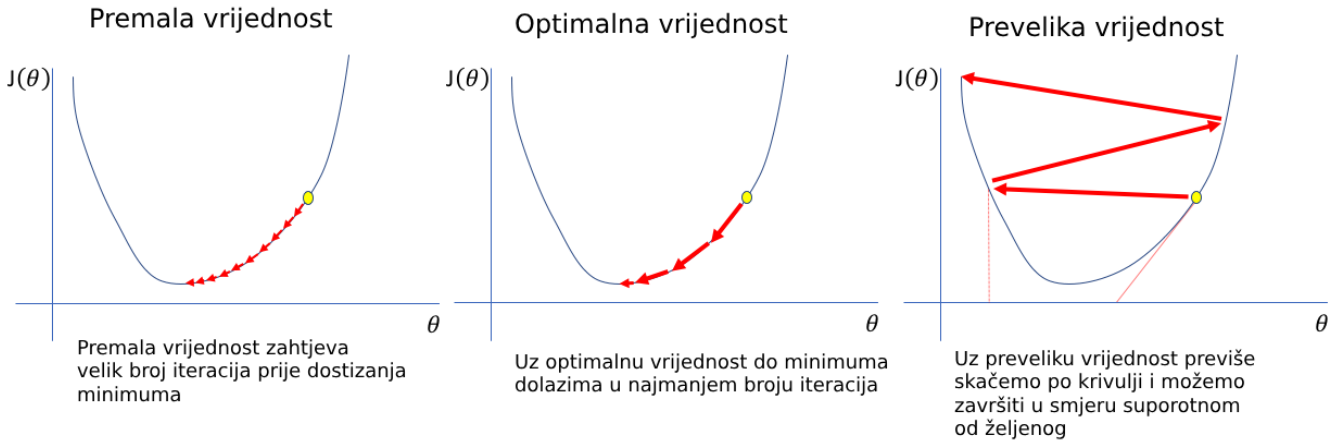
$$x' = x - \alpha * \frac{dy}{dx}$$

x' možemo izračunati kao razliku između trenutne vrijednosti x i umnoška stope učenja α i djelomične derivacije gradijentne funkcije, odnosno $\frac{dy}{dx}$.

Stopa učenja je vrijednost koja predstavlja koliko veliki korak treba napraviti svaki put kada želimo izračunati sljedeću točku. Ne postoje nikakva pravila, osim ako napravimo preveliki korak, mogli bismo preskočiti globalni minimum [20].



Slika 23. Primjer lokalnog i globalnog minimuma



Slika 24. Utjecaj stope učenja na pronalazak minimuma

Ptilagođeno prema: https://www.researchgate.net/figure/Impact-of-learning-rate-on-training-32_fig5_337940311

Prednosti:

- Smjer prema globalnom minimumu uvijek je jednostavan i uvijek je zajamčeno da će doći do optimalnog rješenja.
- Čak i dok je proces učenja u tijeku, stopa učenja može se popraviti kako bi se omogućila poboljšanja.
- Ne stvara šum i daje manju standardnu pogrešku.
- Daje nepristranu procjenu gradijenata.

Nedostaci:

- Sporiji je i proces učenja modela dugo traje.
- Računalno je skup s vrlo velikim broj računskih operacija

Stohastički gradijentni spust

Stohastički gradijentni spust je varijacija gradijentnog spusta. U stohastičkom gradijentnom spuštanju, izračunavamo gradijent koristeći samo nasumični dio promatranja umjesto svih njih. U nekim slučajevima ovaj pristup može smanjiti vrijeme izračuna.

Naš cilj je uzeti sve naše ocjene i stvoriti dvije matrice tako da uzmemo i^{th} redak matrice stavke pomnožen s j^{th} stupcem matrice korisnika čime dobivamo broj koji je blizu stvarnih poznatih ocjena. Ako uvrstimo to u funkciju za minimiziranje kvadratne razlike, možemo reći da želimo pronaći matrice U i V, koje će minimizirati razliku za sve faktore svih ocjena za koje znamo [17].

$$min_{u,v} = \sum_{(i,j)\epsilon R}^{k} e^2 = \sum_{(i,j)\epsilon R}^{k} (r_{ij} - u_i * v_j)^2$$

Kada to uvrstimo u funkciju za izračunavanje sljedeće vrijednosti gradijentnog spusta gdje na $min_{u,v}$ vršimo djelomičnu derivaciju dobijemo sljedeće formule:

$$\frac{d(e^2)}{du_i} = 2 * e * (-v_j) = -2 * (r_{ij} - \widehat{r_{ij}}) * v_j$$

$$\frac{d(e^2)}{dv_i} = 2 * e * (-u_i) = -2 * (r_{ij} - \widehat{r_{ij}}) * u_i$$

Čime na kraju završimo sa sljedećim formulama za izračunavanje sljedeće vrijednosti gradijentnog spusta:

$$u_i = u_i + \alpha * 2(e_{ij} * v_j)$$
$$v_j = v_j + \alpha * 2(e_{ij} * u_i)$$

No s obzirom da je stopa učenja vrlo mali broj možemo izbaciti množenje sa 2 te dobijemo sljedeće:

$$u_i = u_i + \alpha * e_{ij} * v_j$$
$$v_j = v_j + \alpha * e_{ij} * u_i$$

Zatim dobivene funkcije možemo proširiti sa stopom regularizacije (λ) kako bi spriječili prenaučenosti modela.

$$u_i = u_i + \alpha * (e_{ij} * v_j - \lambda * u_i)$$
$$v_j = v_j + \alpha * (e_{ij} * u_i - \lambda * v_j)$$

Ažuriramo vektore u_i i v_j koristeći pogrešku pomnoženu sa stopom učenja, koja je često $\approx 0{,}001$, i oduzimamo vlastiti vektor pomnožen faktorom regulacije (kako bismo zadržali vrijednosti vektora malim) [21].
Točnost našeg modela možemo povećati dodavanjem pristranosti korisnika i stavki.
Dodavanjem pristranosti proširujemo funkciju minimizacije greške [21]:

$$min_{b,u,v} = \sum_{(i,j)\epsilon R}^{k} (r_{ij} - \mu - b_i - b_j - u_i * v_j)^2$$

Gdje je μ prosjek vrijednosti korisnika/smještaja, b_i je pristranost artikla, a b_j je pristranost korisnika.

Tako da na kraju dobijemo:

- $e_{ij} = r_{ij} - u_i * v_j$
- $b_i = b_i + \alpha * (e_{ij} - \lambda * b_i)$
- $b_j = b_j + \alpha * (e_{ij} - \lambda * b_j)$

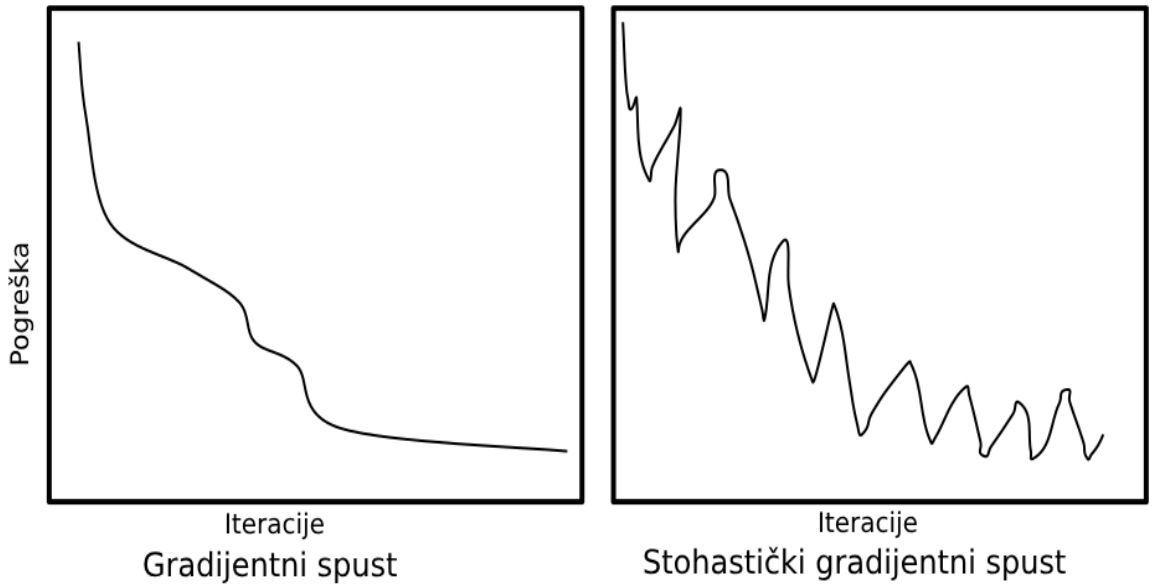
- $u_i = u_i + \alpha * (e_{ij} * v_j - \lambda * u_i)$
- $v_j = v_j + \alpha * (e_{ij} * u_i - \lambda * v_j)$

Prednosti:

- Proces učenja modela je puno brži pošto radi s jednim faktorom po iteraciji.
- Randomizacija pomaže u izbjegavanju ciklusa i ponavljanja primjera.
- Manji računski teret koji dopušta niži standard pogreške.
- Budući da je veličina primjera manja od skupa za vježbanje, ima tendenciju da bude više šuma što omogućuje poboljšanu pogrešku generalizacije.

Mane:

- Stvara više šuma i to može rezultirati duljim vremenom učenja.
- Kao posljedicu ima veću varijancu jer radi s jednim primjerom po iteraciji.



Slika 25: Krivulja greške kod gradijentnog spusta i stohastičkog gradijentnog spusta
Prilagođeno prema: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

2.2.3. Evaluacija modela

Postoji nekoliko načina testiranja algoritma preporuke. Neki od njih nam neće dati kvalitetan uvid u to kako će algoritam raditi ako ga dodamo na web stranicu. Jedini

pravi način da saznamo koliko je algoritam kvalitetan je da ga aktiviramo. No prije toga je moguće evaluirati model [22].

Da bismo napravili kvalitetnu procjenu, potreban nam je skup podataka koji sadrži informacije o svim kombinacijama korisnika i sadržaja. No ukoliko to imamo, možda nam sustav preporuka ni ne treba jer korisnik već točno zna što misli o svim stavkama. Ako nemamo taj skup, trebamo umjesto toga pretpostaviti da su kombinacije korisnika i artikla prisutne u podacima istinite i reprezentativne za sve korisnike. Evaluaciju možemo testirati u tri vrste scenarija [22]:

- Izvanmrežna evaluacija
- Kontrolirani korisnički eksperimenti
- Mrežna evaluacija

2.3.3.1. Izvanmrežna evaluacija

Ideja izvanmrežne evaluacije je korištenje podataka koje smatramo istinitima, a zatim se podijeli podatke na dva dijela i jedan dio se da sustavu preporuke. Drugi dio podataka upotrijebi se za provjeru predviđanja, odnosno da li sustav preporuke daje točne ocjene za stavke u skupu koje do sada nije vidio, tj. one artikle koji su blizu stvarnih ocjena ili one stavke koje su visoko ocijenjene u skrivenim podacima [22].

Ovo se ne smatra dobrim načinom ocjenjivanja preporuka, ali do sada je bilo teško smisliti bolji način osim ako ne možemo provesti A/B testiranje.
Izvanmrežna evaluacija i dalje je način mjerenja učinkovitosti preporuka. Da se stvari dodatno zakompliciraju, sustav preporuke može proći izvanmrežnu evaluaciju s odličnim ocjenama, ali i dati loše rezultate u produkciji [19].

Često ćemo naići na slučajeve u kojima sustav preporuke ne daje nikakve preporuke ili ih možda nema dovoljno. To može predstavljati problem prilikom procjene rada sustava, tako da često takvi slučajevi popunjavaju jednostavnim algoritmom, kao što je preporuka najpopularniji artikala, prosječna ocjena artikla ili prosjek korisničkih ocjena [19].

Izvanmrežna evaluacija koristi podatke koje imamo i mjeri da li je algoritam dobar. S izvanmrežnim eksperimentima imamo ograničene mogućnosti da saznamo koliko je

sustav kvalitetan. To je zato što će se podaci koje imamo vjerojatno temeljiti na ponašanju korisnika kod kojih nije postojao sustav preporuka. Jedan od načina na koji ćemo promatrati testiranje algoritma je da vidimo hoće li sustav preporuke dobro predvidjeti ocjenu korisnika. Ako sakrijemo dio visoko ocjenjenih artikala, vidjet ćemo koliko tih skrivenih ocjena sustav predvidi [19].

2.3.3.2. Mrežna evaluacija

Nakon što smo se uvjerili da naš sustav za preporuku radi najbolje što može u izvanmrežnom pristupu, vrijeme je da implementiramo sustav i testiramo ga na stvarnim korisnicima. Ali to možemo učiniti u fazama [17].

Prva je provođenje kontroliranih eksperimenata. Onda kada povratne informacije budu dobre, prebacimo se na testiranje sa nasumičnim korisnicima u našem sustavu, radeći ono što se zove A/B testiranje [17].

Kontrolirani eksperimenti se vrše tako da se poziva ljude da izvedu test u kontroliranom okruženju. Možemo ih dati da slijede kontrolni popis [17].

Na primjer, da korisnici unese svoje preference, zatim utvrde jesu li preporuke dane s obzirom na njihov unos dobre, pokažemo dvije različite vrste preporuka korisniku i vidimo koja im se čini boljom. Dobra stvar kod kontroliranih eksperimenata je da možemo pratiti ponašanje korisnika.

Korisnicima možemo postavljati pitanja o tome što misle. Nedostatak je to što bi se korisnici mogli ponašati drugačije u kontroliranom okruženju nego što bi se ponašali da su slučajno došli na web stranicu. Također bi moglo biti dugotrajno i teško postaviti eksperiment [17].

3. Implementacija sustava preporuka na podacima turističke agencije

U sljedećem poglavlju ćemo proći kroz proces izrade sustava preporuka za web stranicu Istriasun.com koja trenutno nema implementiran niti jedan oblik sustava preporuke.
Razlog za implementaciju sustava je došao iz potrebe za poboljšanjem korisničkog iskustva i boljeg plasmana smještaja kupcima, bili oni novi ili već postojeći.
Sustav će biti implementiran koristeći KNN model i model matrične faktorizacije. Razlog za korištenje dva modela dolazi iz činjenice da velik broj korisnika prvi put posjećuje stranicu te nemamo zapise o njemu, odnosno ne možemo personalizirati preporuku.

KNN ćemo upotrijebiti kada nemamo dovoljno podataka o korisniku. Služit će za preporuku baziranoj na artiklu (u našem slučaju to će biti smještajna jedinica), a radit će na način da kada korisnik otvori stranicu o pojedinoj smještajnoj jedinici sustav će dati preporuku baziranu na prijašnjim interakcijama korisnika s drugim smještajnim jedinicama.

Matričnu faktorizaciju ćemo koristiti za personaliziranu preporuku postojećim korisnicima. U slučaju web stranice to su korisnici koji su povezani sa jednom ili više rezervacija. Da bi došli do preporuke model će izračunati latentne faktore te na osnovu njih predvidjeti koje smještajne jedinice bi korisnik mogao rezervirati, a koje pregledati (otvoriti stranicu o smještajnoj jedinici).

Implementacija oba modela i klase za preporuku je napisana u programskom jeziku Python.

3.1. Pregled podataka

Podaci su prikupljeni u rasponu od godine dana, od 01.06.2022 do 01.06.2023 i spremani su u MySql bazu podataka. Događaj koji pokreće spremanje podataka u bazu bilo je otvaranje stranice smještajne jedinice od strane korisnika te su se pratili podaci vidljivi na slici 26. Također, pratio se i događaj unosa rezervacije jer web stranica pojedinog korisnika može identificirati tek kada on napravi rezervaciju, a u našem sustavu preporuke služi kako bi zapisu koji je rezultirao sa rezervacijom mogli dati veću „ocjenu“.

	id	objectId	visitorId	visitorIp	sessionId	timestamp	isMobile	checkIn	checkOut	numberOfGuests	reservationId
0	1	770	628f2de207049	89.164.145.111	k89053u9vh5jdb2lob1crfkipl	2022-05-26 09:46:02	0	None	None	None	None
1	2	770	628f2de207049	89.164.145.111	k89053u9vh5jdb2lob1crfkipl	2022-05-26 09:46:02	0	None	None	None	None
2	3	770	628f2de207049	89.164.145.111	k89053u9vh5jdb2lob1crfkipl	2022-05-26 09:46:02	0	None	None	None	None
3	4	11295	628f2eca4f8d3	193.77.85.79	plfln49378adi57nmpv2njf9og	2022-05-26 09:46:02	1	None	None	None	None
4	5	11703	628f2eca4f8d3	193.77.85.79	plfln49378adi57nmpv2njf9og	2022-05-26 09:46:02	1	None	None	None	None
5	6	770	628f2de207049	89.164.145.111	k89053u9vh5jdb2lob1crfkipl	2022-05-26 09:46:08	0	None	None	None	None
6	7	34	628f306e2bd0e	89.164.145.111	90kuuqojmi2s248oprhkldp6v3	2022-05-26 09:47:08	1	None	None	None	None
7	8	250	628f2de207049	89.164.145.111	k89053u9vh5jdb2lob1crfkipl	2022-05-26 09:49:38	0	None	None	None	None
8	9	378	628f3115bae40	89.164.145.111	jjs1ne3bhbe7bm9v9efnpu7t53	2022-05-26 09:49:50	0	None	None	None	None
9	10	231	628f3150b02bb	193.34.143.20	9frfom15teovehlmg0lnhhjd13	2022-05-26 09:51:41	0	None	None	None	None

Slika 26. Prikaz tablice skupljenih podataka

Iz tablice vidimo da se podaci sastoje od:

- objectId - id objekta/stavke u bazi
- visitorId - identifikator korisnika
- visitorIp - ip adresa korisnika
- sessionId - id trenutne sesije korisnika. Korisnik može kroz nekoliko navrata pregledavati ponudu, a sessionId nam služi kako bi saznali koliko artikala korisnik pregleda po jednoj sesiji, odnosno otvaranju web-a
- timestamp - datum kada je korisnik otvorio stanicu. Pomaže nam kako bi izbacili botov-e ili sveli više istih zapisa na jedan.
- isMobile - da li korisnik pregledava ponuda na mobiltelu ili osobnom računalu.
- checkIn, checkOut, numberOfGuests i reservationId - podaci sa rezervacije ukoliko je došlo do iste

3.2. Statistička obrada podataka

Sa isječka koda 1. vidimo da se podaci sastoje od:

- Sveukupno 197855 zapisa
- 61383 zabilježenih sesija generiranih od strane korisnika
- 35986 Ip adresa/korisnika koji su pregledavali objekte
- 1067 objekata
- 1651 rezervacije

```
mycursor.execute("SELECT count(id) from v_tracking_details");
countOfEntries = mycursor.fetchall()
print('Broj zapisa u tablici:', countOfEntries[0][0]);
```

```
Broj zapisa u tablici: 197855
```

```
mycursor.execute("SELECT count(distinct(visitorIp)) from v_tracking_details");
distinctVisitorIps = mycursor.fetchall()
print('Broj jedinstvenih korisnika:', distinctVisitorIps[0][0]);
```

```
Broj jedinstvenih korisnika: 35986
```

```
mycursor.execute("SELECT count(distinct(sessionId)) from v_tracking_details");
distinctSessions = mycursor.fetchall()
print('Broj jedinstvenih sesija:', distinctSessions[0][0]);
```

```
Broj jedinstvenih sesija: 61383
```

```
mycursor.execute("SELECT count(distinct(objectId)) from v_tracking_details");
distinctObjects = mycursor.fetchall()
print('Broj jedinstvenih objekata/artikala:', distinctObjects[0][0]);
```

```
Broj jedinstvenih objekata/artikala: 1067
```

```
mycursor.execute("SELECT count(*) from v_tracking_details where reservationId is not null");
reservations = mycursor.fetchall()
print('Ukupan broj rezervacija: ', reservations[0][0]);
```

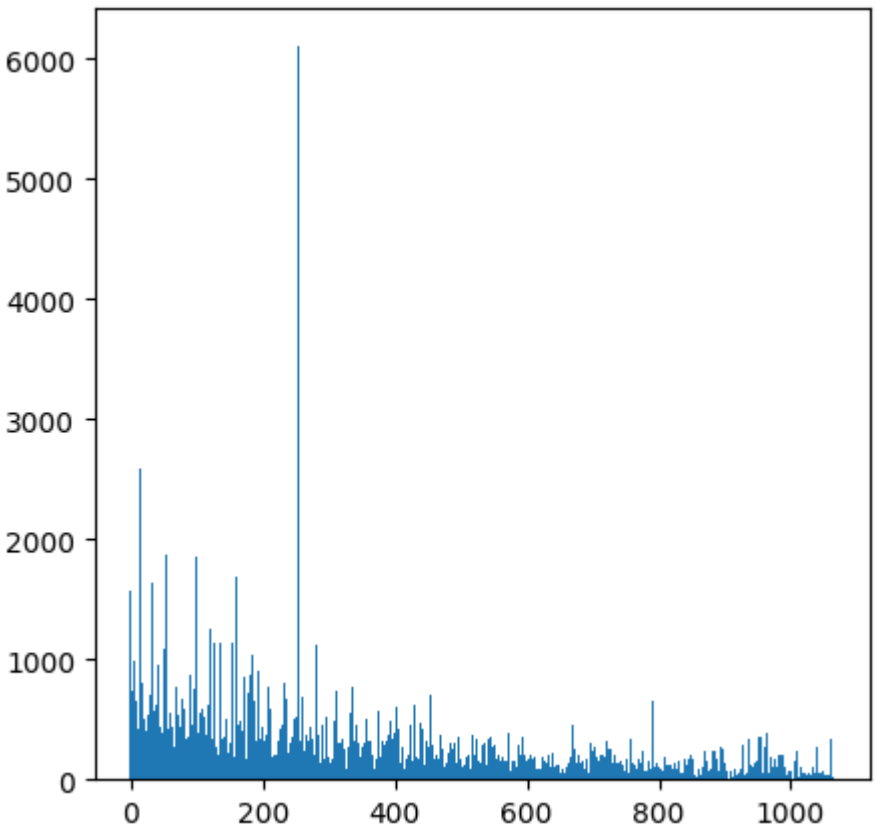
```
Ukupan broj rezervacija:  1651
```

Isječak koda 1. Analiza podataka

Prije same implementacije sustava potrebno je očistiti prikupljene podatke.

Prvo ćemo u stupčastom dijagramu prikazati koliko je puta koji objekt/artikal otvoren na web stranici.

Radi lakšeg prikaza podataka na osi x se nalaze objekti, a na osi y ukupan zbroj pregleda objekta, te se iz dijagrama može vidjeti da među zapisima imamo mogući šum.



Slika 27. Prikaz koliko se puta koji objekt spominje u podacima

3.2.1 Interkvartilni raspon

Pošto nemamo normalnu distribuciju podataka najpogodnija metoda za filtriranje šuma je interkvartilni raspon (eng. Interquartile Range).
Interkvartilni raspon je razlika između gornjeg i donjeg kvartila.

$$IQR = Q_3 - Q_1$$

Donji kvartil dijeli prvu četvrtinu uređenog niza podataka od druge tri četvrtine. Tako dobijemo broj za koji možemo reći da postoji 25% podataka koji su manji od toga broja ili mu jednaki, a istodobno barem 75% podataka su veći od tog broja ili su mu jednaki.

Gornji kvartil dijeli prve tri četvrtine uređenog niza podataka od posljednje četvrtine. Tako dobijemo broj za koji možemo reći da postoji 75% podataka koji su manji od toga broja ili mu jednaki, a istodobno barem 25% podataka su veći od tog broja ili su mu jednaki.

Zatim moramo izračunati gornju i doljnju granicu prihvatljivih podataka.

$$Gornja\ granica = Q_1 - IQR * 1.5$$

$$Donja\ granica\ =\ Q_3 + IQR * 1.5$$

Nakon što izračunamo gornju i doljnju granicu brišemo sve podatke koji su manji od doljnje granice i veći od gornje granice.

Filtriranje podataka je obavljeno na način da smo obrisali sve korisnike s obzirom na to koliko su puta pogledali koji objekt.

3.2.2. Fiiltriranje podataka

Koraci čišćenja podataka:

1. Iz baze podataka uzmemo sve podatke iz tablice
2. Grupiramo podatke po ‘visitorIp’ i ‘objectId’ kako bi dobili broj koliko je puta koji korisnik pregledao pojedini objekt
3. Računamo kvartile koristeći omjere 0.05 i 0.95
4. Računamo gornju i donju granicu
5. Izbrišemo sve korisnike koji spadaju u šum
6. Brišemo sve objekte koji su pregledani manje od sto puta
7. Na kraju iz originalnog skupa podataka obrišemo sve zapise sa prethodno dobivenim Ip adresama.

Nakon prethodno opisanog postupka sa isječka koda 2. vidimo da su nam granice -5 (doljnja) i 11 (gornja) te da osnovu njih ih skupa podataka brišemo 928 korisnika te da smo na kraju ostali na 156635 zapisa naspram originalnih 1977855, odnosno izbrisali smo 21.83% uzorka.

```
vtd = pd.read_sql('Select * from v_tracking_details;', mydb)
```

```
counts = vtd.groupby(['visitorIp', 'objectId'])['objectId'].count().reset_index(name='counts')
_counts = counts.sort_values('counts', ascending=True)['counts']
```

```
quartile1 = _counts.quantile(0.05)
quartile3 = _counts.quantile(0.95)
iqr = quartile3 - quartile1
upper_limit = quartile3 + 1.5 * iqr
lower_limit = quartile1 - 1.5 * iqr
lower_limit, upper_limit
```

```
(-5.0, 11.0)
```

```
ipsToDrop = counts.loc[(counts['counts'] > upper_limit) | (counts['counts'] < lower_limit)]['visitorIp']
print('Broj ip adresa za obrisati:', ipsToDrop.count())
```

```
Broj ip adresa za obrisati: 928
```

```
vtd_c = vtd[~vtd['visitorIp'].isin(ipsToDrop)]
print('Broj uzoraka brije brisanja:',vtd.shape[0])
print('Broj uzoraka poslije brisanja:',vtd_c.shape[0])
print('% uzoraka koji je ostao:', np.round( (vtd_c.shape[0]/vtd.shape[0])*100, 2 ), '%' )
```

```
Broj uzoraka brije brisanja: 197855
Broj uzoraka poslije brisanja: 156635
% uzoraka koji je ostao: 79.17 %
```

Isječak koda 2. Čišćenje podataka pomoću interkvartilnog raspona

3.3. Grupiranje

U sljedećem poglavlju ćemo implementirati model preporuke koristeći KNN.
Za implementaciju ćemo napraviti tri seta podataka.
Prvi set podataka će biti matrica u kojoj je zapisano koliko je puta koji korisnik gledao koji objekt.
Drugi set podataka se sastoji od tablice da li je korisnik gledao objekt, odnosno ako je korisnik gledao objekt u tablici će biti upisan broj 1 neovisno o tome koliko je puta korisnik pogledao objekt.
I treći set podataka koji je nadograđeni drugi model s time da je u tablicu upisan broj 2 ako je kupac rezervirao objekt.
Na kraju ćemo usporediti rezultate modela ovisno o tome koji set podataka je korišten.

3.3.1 Primjer rada modela

Radi lakšeg razumijevanja algoritma, proučimo sljedeću tablica koja se sastoji od šest korisnika i jednako toliko objekata i pokazuje da li je korisnik pogledao objekt.
Ukoliko se prethodno opisanoj metodi proslijedi objekt 3, model grupiranja uspoređuje naš objekt (redak) sa svim ostalim redcima preko formule za kosinus.
Na primjer za izračun udaljenosti između trećeg (proslijeđenog retka) i drugog retka
Izračunamo R3 * R2 (R je oznaka za redak):
R3 * R2 = (0*1) + (1*1) + (1*0) + (0*1) + (0*1) + (1*1) = 2
Zatim izračunamo magnitude redaka R3 i R2
|R2| = $\sqrt{0^2 + 1^2 + 1^2 + 0^2 + 0^2 + 1^2} = \sqrt{3}$
|R3| = $\sqrt{1^2 + 1^2 + 0^2 + 1^2 + 1^2 + 1^2} = \sqrt{5}$
sim(R3, R2) = $\frac{2}{\sqrt{3} * \sqrt{5}} \approx 0.52$
Za parove sim(R3, R) i dobijemo sljedeće vrijednosti:
sim(R3, R1) ≈ 0.52
sim(R3, R3) ≈ 0.99
sim(R3, R4) ≈ 0.8
sim(R3, R5) ≈ 0.32
sim(R3, R6) ≈ 0.45

Korisnik/ Objekt	1	2	3	4	5	6
1	0	1	1	1	0	0
2	0	1	1	0	0	1
3	1	1	0	1	1	1
4	0	1	1	1	1	1
5	0	0	1	1	0	0
6	1	0	0	0	0	0

Tablica 2. Prikaz da li je korisnik pogledao objekt

Iz prethodno dobivenih vrijednosti zaključuje se da je četvrti redak najsličniji trećem retku uz zanemarivanje vrijednosti *sim(R3, R3),* pošto se tu usporedio redak sam sa sobom i tu će vrijednost uvijek biti ≈ 1.

Nakon što su podaci očišćeni možemo krenuti u izradu prvog modela preporuke koji će biti implementiran pomoću grupiranja.

3.3.2. Definiranje modela

Za izradu modela potrebna su nam dva paketa. Prvi je *sklearn* iz kojeg ćemo upotrijebiti modul *NearestNeighbors*, a drugi je *scipy* iz kojeg nam je potreban modul *csr_matrix*.

Prije inicijalizacije samog modela potrebno je generirati prethodno spomenute setove podataka.

Prvo smo generirali *vtd_matrix* matricu koja se sastoji od objekata, korisnika i broja koliko je korisnik puta pregledao koji objekt. Iz njega zatim generiramo *vtd_matrix_1* i *vtd_matrix_2* koje se također sastoje od objekta i korisnika, ali umjesto broja pregleda sadrži zapis da li je korisnik pregledao objekt.

Zatim u sve tri matrice podatke koje nemamo popunimo sa nulom čime dobivamo dvije nove matrice *vtd_matrix_full*, *vtd_matrix_full_1* i *vtd_matrix_full_2*.

```
vtd_matrix = vtd.groupby(['visitorIp', 'objectId'])['objectId'].count().reset_index(name='counts')

vtd_matrix_1 = vtd_matrix.assign(counts=1)
vtd_matrix_2 = vtd_matrix.assign(counts=1)

vtd_matrix_full = vtd_matrix.pivot(index='objectId', columns='visitorIp', values='counts').fillna(0)
vtd_matrix_full_1 = vtd_matrix_1.pivot(index='objectId', columns='visitorIp', values='counts').fillna(0)
vtd_matrix_full_2 = vtd_matrix_2.pivot(index='objectId', columns='visitorIp', values='counts').fillna(0)

mapirani_objekti = {
    object: i for i, object in
    enumerate(list(oc.set_index('objectId').loc[vtd_matrix_full.index].index))
}

reservations = vtd[vtd['reservationId'].notnull()]

for i, x in reservations.iterrows():
    vtd_matrix_full_2.loc[x['objectId'], x['visitorIp'] ] = 2;

object_user_matrix = csr_matrix(vtd_matrix_full.values)
object_user_matrix_1 = csr_matrix(vtd_matrix_full_1.values)
object_user_matrix_2 = csr_matrix(vtd_matrix_full_2.values)
```

Isječak koda 3. Manipulacija osnovnih podataka kako bi dobili tri potreba seta.

Sljedeće što nam je potrebno je rječnik *mapirani_objekti* u kojemu se nalaze podaci na kojem mjestu (indeksu) se nalaze zapisi unutar matrica *vtd_matrix_full* ,*vtd_matrix_full_1* i *vtd_matrix_full_2*.

Zatim od matrica kreiramo rijetke matrice (eng. Sparse Matrix, modul *csr_matrix*) iz razloga što pohranjuju samo elemente koji nisu nula i njihove indekse redova. Zbog toga korištenje rijetkih matrica može značajno smanjiti količinu memorije potrebne za pohranu podataka.

U konkretnom slučaju nam to možda nije ni potrebno no kada imamo velike količine podataka značajno nam smanjuje količinu memorije, posebice kada imamo rijetku matricu, odnosno kada većina zapisa nema vrijednosti.

Nakon procesa pripreme inicijaliiramo *knn_model* pomoću *NearestNeighbors* metode iz *sklearn* paketa. Postavljamo joj *euclidean* kao metriku računanja jer računa relativnu udaljenost između točaka.

```
def preporuci(knn_model, mapirani_objekti, object_user_matrix, objectId, broj_preporuka):
    knn_model.fit(object_user_matrix);
    print('Odabari objectId:', objectId);
    index = mapirani_objekti.get(objectId);
    if (index is None):
        print('Objekt se ne nalazi u danim podacima');
        return;
    udaljenosti, indeksi = knn_model.kneighbors(object_user_matrix[index], n_neighbors=broj_preporuka+1);
    preporuke = sorted(list(zip(indeksi.squeeze().tolist(), udaljenosti.squeeze().tolist())), key=lambda x: x[1])[:0:-1];
    for i, (idx, dist) in enumerate(preporuke):
        print('{0}. {1}, sa udaljenosti {2}'.format(i+1, idx, math.ceil(dist*100)/100))
```

Isječak koda 4. Metoda preporuči

Na isječku koda 4. je prikazana metoda koja preporuča slične objekte na osnovi predanog *objectId-a* i uspoređuje koje su objekte drugi korisnici pregledali, a da su gledali predani objekt.

3.3.3. Rezultati modela

Metoda *preporuci* prima 5 parametara. Model (prethodno inicijalizirani model), rječnik u kojemu je mapirano u kojem retku se nalazi koji objekt (objectId), matricu korisnika i objekata veličine korisnici*objekti, id objekta za koji tražimo slične te broj koji označuje koliko objekata želimo da nam metoda preporuči.

Na isječku koda 5. imamo prikaz rezultata oba modela, prvom modelu su proslijeđeni podaci o tome koliko je puta korisnik pogledao koji objekt, a drugi da li je korisnik pogledao objekt.
Iz rezultata vidimo da prvi i drugi modeli daju različite rezultate, točnije za objekt 770 modeli nemaju presijecanja, a drugi i treći imaju za testirani objekt stopostotno preklapanje.

```
print('Podaci koliko je puta koji korisnike gledao koji objekt.')
preporuci(knn_model, mapirani_objekti, object_user_matrix, 770, 3)
✓ 0.0s

Podaci koliko je puta koji korisnike gledao koji objekt.
408, sa udaljenosti 40.963398296528084
810, sa udaljenosti 42.37924020083418
24873, sa udaljenosti 43.197222132910355


print('Podaci da li je korisnik gledao objekt.')
preporuci(knn_model, mapirani_objekti, object_user_matrix_1, 770, 3)
✓ 0.0s

Podaci da li je korisnik gledao objekt.
36723, sa udaljenosti 6.557438524302
11849, sa udaljenosti 6.557438524302
351, sa udaljenosti 6.928203230275509


print('Podaci da li je korisnik gledao objekt i da li je rezevirao.')
preporuci(knn_model, mapirani_objekti, object_user_matrix_2, 770, 3)
✓ 0.0s

Podaci da li je korisnik gledao objekt i da li je rezevirao.
11849, sa udaljenosti 6.782329983125268
36723, sa udaljenosti 6.782329983125268
10053, sa udaljenosti 7.54983443527075
```

Isječak koda 5. Rezultati s obzirom na koji se podataka se koristio

Kada pokrenemo preporuku za sve objekte sa brojem preporuka postavljenim na 3, prva dva modela se poklapaju u $\approx 20\%$ preporuka što nam pokazuje koliko je model osjetljiv na promjene u obliku podataka koje primi.
Dok se drugi i treći preklapaju u $\approx 89\%$ preporuka, a razlog tomu je što je udio rezervacija u odnosu na ukupan broj elemenata tablice $\approx 1{,}5\%$.

```
poklapanja = 0;
ukupno = 0;
for k,v in mapirani_objekti.items():
    p_b = preporuci(knn_model, mapirani_objekti, object_user_matrix, k, 3);
    p_v = preporuci(knn_model, mapirani_objekti, object_user_matrix_1, k, 3);
    poklapanja += (len(interlist(p_b[0], p_v[0])) - 1);
    ukupno += 3;
print(math.ceil( (poklapanja/ukupno)*100), '%' );
```

```
20 %
```

Isječak koda 6. Izračun preklapanja

Kod prvog modela možemo imati naklonjenost modela prema nekim objektima koji su puno puta pregledani, a potencijalno nemaju veće značenje za korisnika.
Kod drugog modela možemo imati suprotan učinak, svi podaci imaju istu “težinu” i ne možemo uhvatiti značenje pojedinog objekta.
Dok treći model u teoriji smanjuje problem “težine” podataka, ali imamo nedovoljan broj rezervacija da bi mogli reći kako to utječe na model.

Tu dolazimo do problema KNN jer kao što je vidljivo, kod ovog pristupa model ne možemo testirat model pošto nema što predvidjeti niti klasificirati jer model samo pronalazi sličnosti među objektima.

Kako bi znali koji je model točniji za korisnike trebalo bi provesti testiranje na korisnicima.

3.4. Matrična faktorizacija

U ovome poglavlju ćemo vidjeti model preporuke implementiran pomoću matrične faktorizacije i stohastičkog gradijentnog spusta.
Za trening i testiranje koristiti ćemo tablicu *vtd_matrix_full_2* iz prethodnog poglavlja koja sadrži podatke o tome da li je korisnik pogledao ili rezervirao smještaj.

3.4.1. Primjer rada modela

Prije prikaza modela i analize rezultata treniranja proći ćemo kroz primjer rada modela, odnosno način na koji model ažurira vrijednosti latentnih faktora i pristranosti.
Model optimizira svaki latentni faktor za nasumično odabrani uzorak podataka po faktoru, u našem slučaju model prilikom optimizacije odabire pola podataka.
Za primjer ćemo koristiti tablicu 3 dok tablica 4 predstavlja inicijalne latentne faktore korisnika i objekata, odnosno matrice U i V u kojima su sve vrijednosti postavljenje na 0.5.

Korisnik/ Objekt	1	2	3
1	1	1	1
2	0	1	1
3	1	0	0

Tablica 3: Podaci da li je korisnik pregledao objekt

Faktor/ Korisnik	1	2	3
1	0.5	0.5	0.5
2	0.5	0.5	0.5
3	0.5	0.5	0.5

Tablica 4: Latentni faktori korisnika

Faktor/ Objekt	1	2	3
1	0.5	0.5	0.5
2	0.5	0.5	0.5
3	0.5	0.5	0.5

Tablica 5: Latentni faktori objekata

Uzimamo prvi stupac iz matrica U i V, zatim nasmično odaberemo % uzorka, te za svaki uzorak koji je pozitivan (ne želimo trenirati model na podacima koje ne znamo) računamo predviđenu vrijednosti množeći matrice U i V^T.
Predviđamo vrijednost iz ćelije korisnika 2 i objekta 2 koja iznosi 1.
Vrijednost dobijemo tako da pomnožimo vrijednost latentnog faktora 1 za korisnika 2 koja iznosi 0.5 i latentnog faktora 1 za objekt 2 koji također iznosi 0.5 čime dobijemo da je predviđena vrijednosti 0.75.
Sljedeće računamo razliku između stvarne i predviđene vrijednosti koja iznosi 0.25.
Zatim ažuriramo vrijednosti faktora u matricama U i V koristeći formule iz prethodnih poglavlja sa stopom učenja od 0.03 i vrijednosti regularizacije od 0.05.

$$u_i = u_i + \alpha * (e_{ij} * v_j - \lambda * u_i)$$
$$v_j = v_j + \alpha * (e_{ij} * u_i - \lambda * v_j)$$

$$u_i = 0.01 + 0.03 * (0.25 * 0.5 - 0.05 * 0.5) \ = 0.53$$
$$v_j = 0.01 + 0.03 * (.025 * 0.5 - 0.05 * 0.5) \ = 0.53$$

Što rezultira u sljedećim matricama

Faktor/ Korisnik	1	2	3
1	0.5	0.5	0.5
2	0.53	0.5	0.5
3	0.5	0.5	0.5

Tablica 6: Ažurirani latentni faktori korisnika

Faktor/ Objekt	1	2	3
1	0.53	0.5	0.5
2	0.5	0.5	0.5
3	0.5	0.5	0.5

Tablica 7: Ažurirani latentni faktori objekata (označeni žutom bojom)

Zatim dalje predviđamo vrijednosti za ostatak podataka iz uzorka za faktor 1.

Recimo da su to vrijednosti za parove korisnik 2 objekt 1 te korisnik 1 objekt 3.

Faktor/ Korisnik	1	2	3
1	0.5	0.5	0.5
2	0.53	0.5	0.5
3	0.5	0.5	0.5

Tablica 8: Ažurirani latentni faktori korisnika nakon optimizacije 1 faktora

Faktor/ Objekt	1	2	3
1	0.55955	0.5	0.5
2	0.5	0.5	0.5
3	0.52984399	0.5	0.5

Tablica 9: Ažurirani latentni faktori objekata nakon optimizacije 1 faktora

Nakon što prođemo sve uzorke računamo RMSE, a postupak za trenutni faktor se ponavlja dok ne dosegnemo predefinirani broj iteracija ili dok razlika između RMSE trening i test podataka nije manja od predefinirane vrijednosti.

Ako prođemo optimizaciju svih faktora dobijemo matrice faktora slične tablicama 10 i 11.

Faktor/ Korisnik	1	2	3
1	0.5593475	0.52301769	0.53694113
2	0.53	0.52718901	0.5
3	0.5	0.52775	0.52338519

Tablica 10: Vrijednosti latentnih faktora korisnika nakon optimizacije svih faktora

Faktor/ Objekt	1	2	3
1	0.55955	0.5	0.53778766
2	0.5	0.55269875	0.5
3	0.52984399	0.52532575	0.51843911

Tablica 10: Vrijednosti latentnih faktora objekata nakon optimizacije svih faktora

Nakon što model završi proces treninga uspoređujemo mu konačne vrijednost odstupanja predviđanja te ili smo našli optimalno rješenje ili pokrećemo model ponovo sa novim vrijednostima.

3.4.2. Definiranje modela

Nakon što smo prošli primjer rada modela možemo prijeći na definiranje modela.

Na slici 29. je prikazana metoda *meta_parameter_train* koja prolazi kroz kombinacije stope učenja i broja latentnih faktora i računa korijen srednje kvadratne pogreške (RMSE) kako bi dobili optimalnu kombinaciju za naš model, odnosno optimizirali hiperparametre. Kao što je vidljivo sa slike model smo testirali sa vrijednostima stope učenja 0.0005, 0.001, 0.005, 0.01, 0.05 i 0.1 te sa 25, 50, 75, 100 i 150 latentnih faktora. Također tokom svih iteracija imamo fiksiranu vrijednost regularizacije koja iznosi 0.002.

Podaci su podijeljeni na trening i test u omjernu 7:3 s time da je podjela napravljena na podacima koji imaju vrijednosti.

Također, tokom optimizacije hiperparametara i treninga, model je radio samo na podacima koje znamo, jer ne želimo model optimizirati za vrijednosti koje kasnije predviđamo.

```
import os
import random
import sys
import pickle
import math
import json

import numpy as np
import pandas as pd
from decimal import Decimal
from collections import defaultdict
from datetime import datetime
```

Slika 28: Potrebni paketi

```
def meta_parameter_train(self, ratings_df):

    for lr in [0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]:
        self.Regularization = Decimal(0.002)
        self.BiasReg = Decimal(0.002)

        self.BiasLearnRate = Decimal(lr)
        self.LearnRate = Decimal(lr)

        for k in [25, 50, 75, 100, 150]:
            self.rmse_values[k] = {}
            self.rmse_values[k]['train'] = []
            self.rmse_values[k]['test'] = []
            self.initialize_factors(ratings_df, k)
            print("Treniranje na {} faktora".format(k))
            print(str(k), "faktor, iteracija, train_mse, test_mse, vrijeme")

            test_data, train_data = self.split_data(10, ratings_df)
            columns = ['visitorIp', 'objectId', 'counts']
            ratings = train_data[columns].to_numpy()
            test = test_data[columns].to_numpy()

            iterations = 0
            index_randomized = random.sample(range(0, len(ratings)), (len(ratings) - 1))

            for factor in range(k):
                factor_iteration = 0
                factor_time = datetime.now()

                last_err = sys.maxsize
                last_test_mse = sys.maxsize
                finished = False

                indexes = random.choices(index_randomized, k=math.floor(len(index_randomized) * 0.5))
                while not finished:
                    train_mse = self.stocastic_gradient_descent(factor, indexes, ratings)

                    iterations += 1
                    test_mse = self.calculate_rmse(test, factor)
```

Slika 29: Metoda meta_parameter_train

Na slici 30. je prikazana metoda *stocastic_gradient_descent* koja ažurira vrijednosti za pristranost korisnika (*user_bias*), pristranost objekta (*item_bias*) te latentnih faktora korisnika i objekata (*user_factors*, *item_factors*) koristeći formule opisane u prethodnim poglavljima:

- $b_i = b_i + \alpha * (e_{ij} - \lambda * b_i)$
- $b_j = b_j + \alpha * (e_{ij} - \lambda * b_j)$
- $u_i = u_i + \alpha * (e_{ij} * v_j - \lambda * u_i)$
- $v_j = v_j + \alpha * (e_{ij} * u_i - \lambda * v_j)$

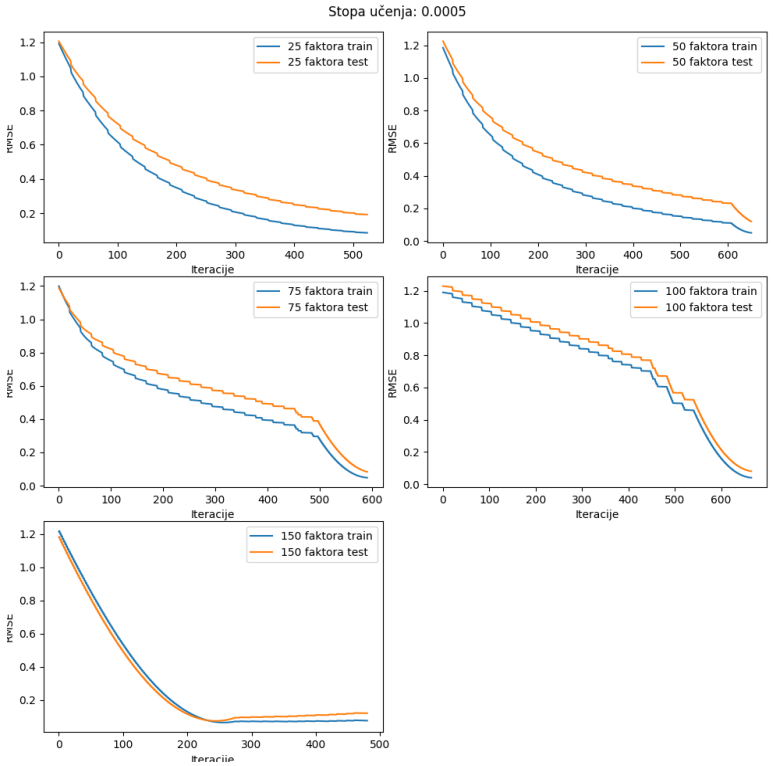
Gdje je b_i *item_bias*, b_j *user_bias*, u *user_factors*, v *item_factors*

```
def stocastic_gradient_descent(self, factor, index_randomized, ratings):

    lr = self.LearnRate
    b_lr = self.BiasLearnRate
    r = self.Regularization
    bias_r = self.BiasReg
    self.test = index_randomized

    for inx in index_randomized:


        rating_row = ratings[inx]

        u = self.u_inx[rating_row[0]]
        i = self.i_inx[rating_row[1]]
        rating = Decimal(rating_row[2])

        if(rating == 0.0):
            continue

        pred = self.predict(u, i)
        err = (rating - pred)

        self.user_bias[u] += b_lr * (err - bias_r * self.user_bias[u])
        self.item_bias[i] += b_lr * (err - bias_r * self.item_bias[i])

        user_fac = self.user_factors[u][factor]
        item_fac = self.item_factors[i][factor]

        self.user_factors[u][factor] += lr * (err * item_fac - r * user_fac)
        self.item_factors[i][factor] += lr * (err * user_fac - r * item_fac)

    return self.calculate_rmse(ratings, factor)
```

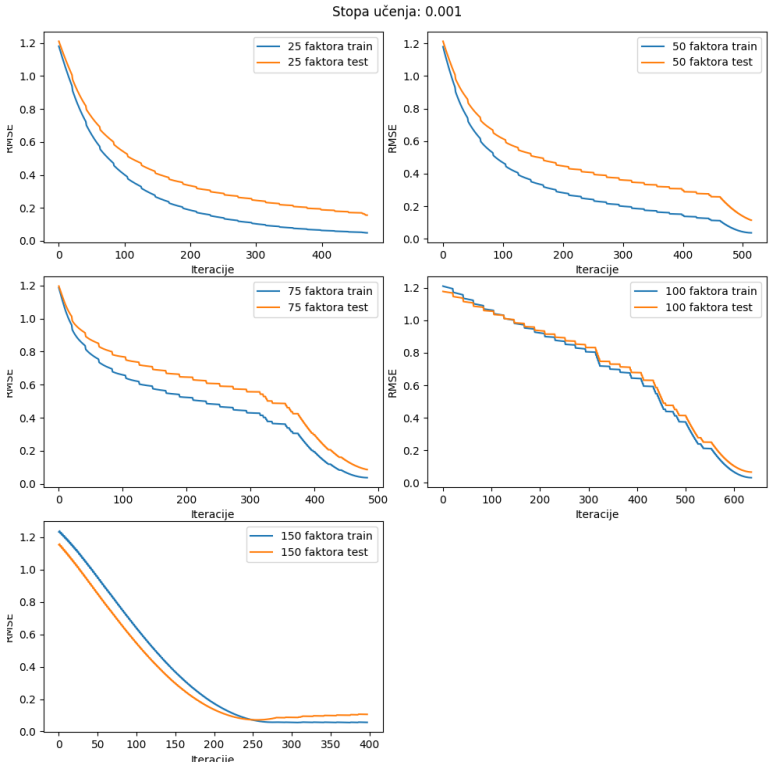
Slika 30: Metoda stocastic_gradient_descent

Te se nakon ažuriranja vrijednosti model evaluira pomoću RMSE-a na podacima za test (validaciju).

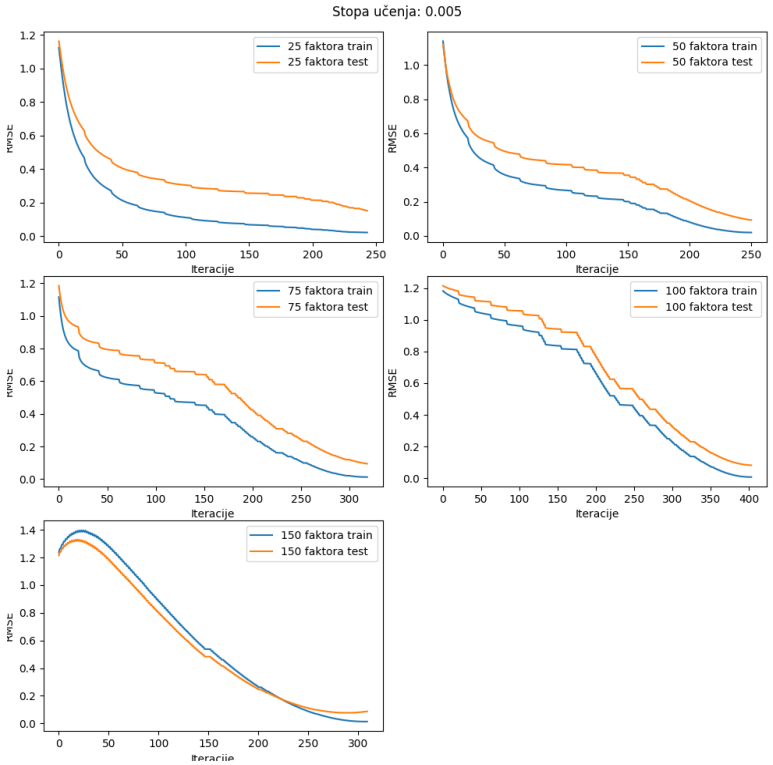
3.4.3. Rezultati optimizacije hiperparametara



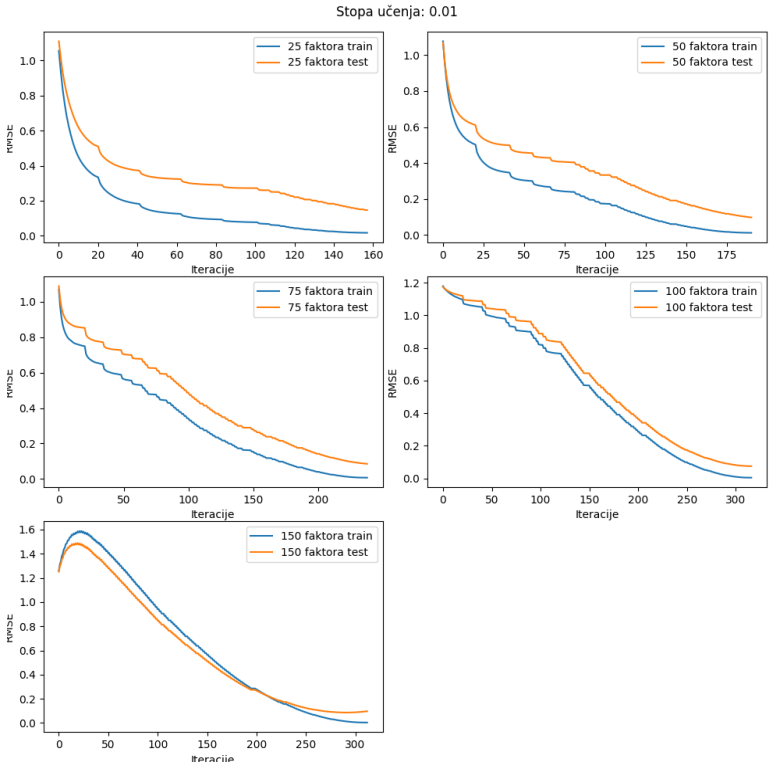
Slika 31: RMSE za stopu učenja od 0.0005



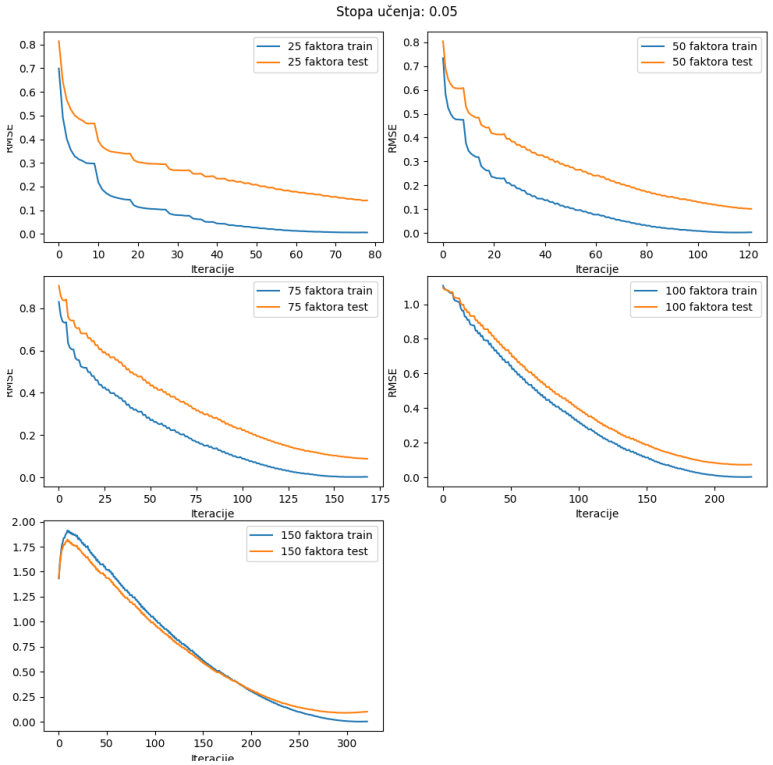
Slika 32: RMSE za stopu učenja od 0.001



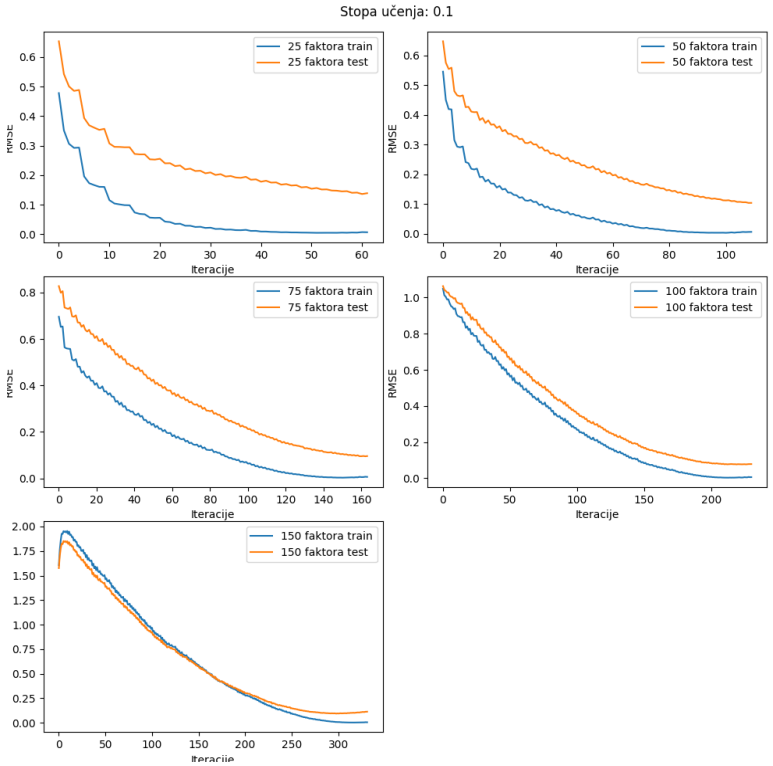
Slika 33: RMSE za stopu učenja od 0.005



Slika 34: RMSE za stopu učenja od 0.01



Slika 35: RMSE za stopu učenja od 0.05



Slika 36: RMSE za stopu učenja od 0.1

Na prethodnim slikama vidimo grafove koji prikazuju rezultate optimizacije hiperparametara, odnosno usporedbu RMSE-a na trening i test podacima te sa grafova može primijetiti da u svim slučajevima modelima sa 150 faktora u vrijednost RMSE-a povećava kroz broj iteracija.
Nadalje, modeli sa 25, 50 i 75 faktora imaju značajno veću razliku u odstupanju trening i test podataka, čime nam ostaju modeli sa 100 faktora.
Od preostalih modela najbolji se pokazao model sa stopom učenja 0.001 iz razloga što mu je razlika između trening i test podataka najmanja (0.034) te ima najmanju vrijednost RMSE-a testnim podacima (0.66).
Nakon što smo pronašli optimalne parametre (stopa učenja: 0.001, broj latentnih faktora: 100), s njima smo pokrenuli trening modela koji ćemo koristiti za preporuku.

3.5. Klasa za preporuku

U sljedećem poglavlju ćemo objasniti rad klase za preporuku, paketi potrebni za njemu implementaciju te parametre koje ona prima.
Klasa koristi modele definirate u prethodna dva poglavlja.
Na slici 37. vidimo pakete potrebne za implementaciju klase.

```
from sklearn.neighbors import NearestNeighbors
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
```

Slika 37. Paketi potrebni za implementaciju klase za preporuku

Klasa kao parametre prima *objectId, visitorIp* te *k* koji predstavlja koliko objekata se preporučuje. Postoje dva tipa preporuke, prvi je s obzirom na objekt, a drugi s obzirom na korisnika. Prije same preporuke potrebno je učitati originalne podatke (tablicu *vtd_matrix_full_2* iz poglavlja 3.3 i 3.4), to jest podatke na kojima se vršio trening modela iz razloga kako bi prilikom preporuke mogli zanemariti objekte koje je korisnik već pogledao, također potrebno je učitati model, te izračunati matricu koja predstavlja predviđene vrijednosti.
Na slici 38. vidimo opisano. Prilikom inicijalizacije klase postavljaju je vrijednosti *objectId, visitorIp* i *k.* Nakon čega metoda *loadOrginalData()* učitava originalne podatke, metode *loadItemDataAndFactors()* i *loadUserDataAndFactors()* učitavaju model, odnosno istrenirane faktore za objekte i korisnike, a metoda *loadBiases()* učitava naklonjenosti.
Nakon čega imamo metodu *get()* koje provjerava da li se u treniranom model nalazi proslijeđeni objekt ili korisnik, a ako se ne nalaze vraća se poruka, a u suprotnom se odvijaju metode *basedOnItem()* ili *basedOnUser().*

```
class Recommend(object):

    def __init__(self, objectId = False, visitorIp = False, k = 3):
        self.objectId = objectId
        self.visitorIp = visitorIp
        self.k = k

        self.orginal_data = loadOrginalData()
        self.item, self.item_enum = loadItemDataAndFactors()
        self.user, self.user_enum = loadUserDataAndFactors()

        self.user_bias, self.item_bias = loadBiases()

        self.ratings = np.dot(self.item, self.user.T)

        return None

    def get(self):
        if (self.objectId and self.objectId not in self.item_enum):
                return "Nedovoljan broj zapisa za traženi smještaj"
        if (self.visitorIp and self.visitorIp not in self.user_enum):
                return "Nedovoljan broj zapisa o traženom korisniku"
```

Slika 38. Klasa Recommend

Metoda *basedOnItem()* se pokreće u slučaju kada je proslijeđen *objectId*, a unutar nje se odvija grupiranje, model iz poglavlja 3.3, koji vraća najbližih *k* objekata s obzirom na matricu predviđenih vrijednosti.

Dok se metoda *basedOnUser()* pokreće kada je proslijeđen *visitorIp* unutar koje koristimo istrenirane podatke iz poglavlja 3.4, a metoda pronalazi predviđene vrijednosti za proslijeđenog korisnika uvećane za naklonjenosti pojedinog objekta koje zatim sortiramo slijedno, od najveće prema najmanjoj te vratimo prvih *k* objekata.

3.5.1. Prikaz rada klase za preporuku povezane s korisničkim sučeljem turističke agencije

Za potrebe prikaza rada modela za preporuku napravljena je aplikacija unutar koje se prikazuju objekti te korisnici.

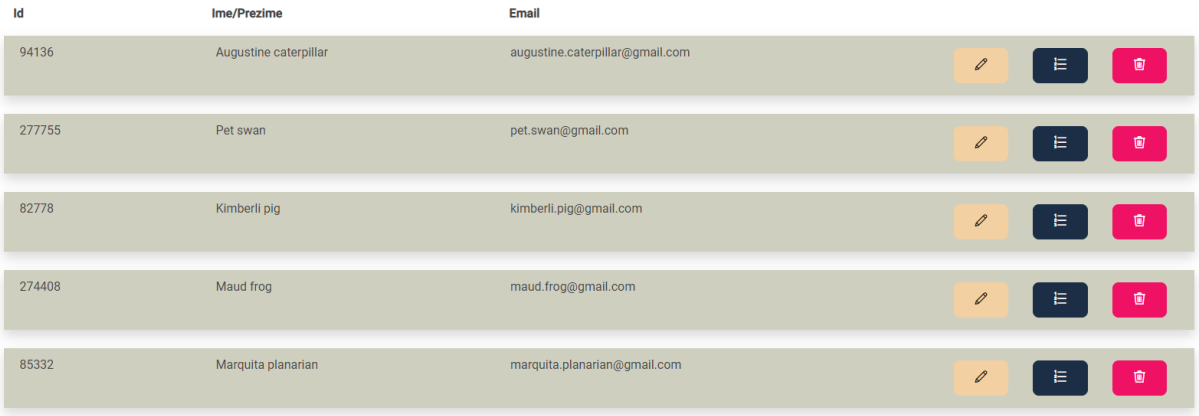
Na slici 39. vidimo prikaz sučelja i ispis objekata odnosno smještaja. Kod svakog smještaja sa desne strane se nalazi gumb kojim se šalje *objectId* na osnovu kojeg se radi preporuka opisana u prethodnom poglavlju, a kao odgovor se dobiva lista od *k* objekata ili poruka da nema dovoljno zapisa o traženom objektu.

Na slici 41. je su prikazana oba slučaja.

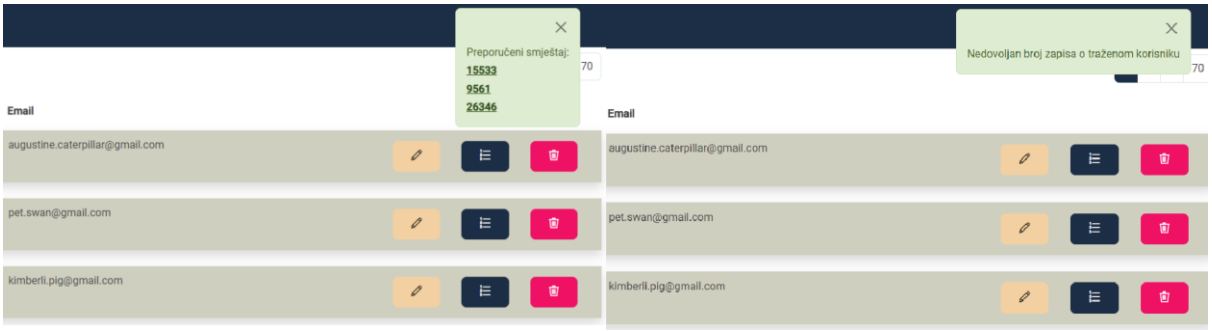
Slika 39. Lista smještaja

Slika 40. Prikaz odgovora klase za preporuku u slučaju smještaja

Na slici 41. vidimo popis kupaca. Kod svakog kupca se s desne strane nalazi gumb čije je funkcija identična kad i kod smještaja uz iznimku da se šalje *visitorIp.* Postoje dva moguća odgovora, jedan je lista od *k* objekata ili poruka da nema dovoljno zapisa o traženom korisniku.
Na slici 42. je su prikazana oba slučaja.



Slika 41. Lista kupaca



Slika 42.Prikaz odgovora klase za preporuku u slučaju korisnika

3.6 Usporedba grupiranja i matrične faktorizacije

U ovome poglavlju ćemo usporediti model implementiran grupiranjem (KNN) i model implementiran matričnom faktorizacijom, odnosno dva modela korištena u prethodnom poglavlju. Također ćemo objasniti kada je najbolje koristiti koji model.

Iako oba izvršavaju istu funkciju (daju preporuku od *k* objekata), načini na koji dođu do preporuke se znatno razlikuju.

Kod KNN-a do rješenja se dolazi rješavanjem problema rangiranja, odnosno iz skupa podataka pronađi artikle koji se poklapaju u najviše značajki. U našem slučaju značajke predstavlja podatak da li je korisnik pregledao ili rezervirao objekt.
Prednost modela je da ga nije potrebno trenirati, jednostavan je za implementaciju te brzo možemo doći po rezultata i testirati različite funkcije za izračun sličnosti.
Pogodan je za korištenje kada nemamo podatke o interakciji korisnika sa artiklima, nego samo informacije o artiklima. Također, najbolji je za korištenje u poslovnim okruženjima gdje je interakcija korisnika i artikala minimalna, odnosno nedovoljna za izvući dodatke informacije iz nje.

Kod matrične faktorizacije do rješenja se dolazi predviđanjem ocjene korisnika za artikle koje nije ocijenio. Podaci ne moraju nužno sadržavati ocjene korisnika, nego mogu biti izvučeni iz same interakcije korisnika sa artiklom.
Model je potrebno trenirati kako bi pronašao povezanosti između korisnikovog odabira korištenjem latentnih faktora koje model nastoji optimizirati.
Prednost mu je što je puno skalabilniji, odnosno daje bolje rezultate kada imamo puno i malo zapisa o interakcijama u odnosu na KNN.
Iako je kompleksniji zahtjeva manje računalne snage jer prilikom svake preporuke ne mora računati sve kombinacije kao što to radi prvi model, nego iz treniranih podataka izvuče predviđanja.

4. Zaključak

Izrada sustava je proces koji prije svega zahtjeva poznavanje aplikacije domene, to jest poslovnu logiku sustava za koji se sustav izrađuje te na kakav način korisnici vrše interakciju sa sustavom i kakve podatke možemo dobiti iz interakcije.

Također, bitno je razumjeti podatke koje imamo na raspolaganju te ih filtrirati i preoblikovati tako da model iz njih može pronaći povezanosti.

Generalno, matrična faktorizacija je bolji pristup za izradu sustava preporuke jer je fleksibilnija za rad sa puno i malo zapisa o interakcijama, zahtjeva manje računalne snage i pronalazi „skrivene“ povezanosti između korisnika i artikla.

No, ako i napravimo sve od prethodno navedenog, to nam ne osigurava kvalitetu preporuke. Da bi saznali stvarnu kvalitetu sustava potrebno je provesti A/B testiranje.

Čak i nakon korisničkog testiranja i utvrđivanja kvalitete sustava potrebno ga je kontinuirano održavati, unaprjeđivati i pokušati izvući dodatne čimbenike koji utječu na povećanja točnosti.

5. Literatura

[1] Aggarwal, C. C. (2016). Recommender systems. Springer.

[2] Jian Zhao, Christopher Collins, Fanny Chevalier, Ravin Balakrishnan. Interactive Exploration of Implicit and Explicit Relations in Faceted Datasets. IEEE Transactions on Visualization and Computer Graphics, 2013, 19 (2), pp.2080-2089. ff10.1109/TVCG.2013.167ff. ffhal-01558714

[3] Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* 9, 59 (2022). Dostupno na: https://doi.org/10.1186/s40537-022-00592-5 . Pristupljeno: 25.02.2023

[4] Beel, J., Gipp, B., Langer, S. *et al.* Research-paper recommender systems: a literature survey. *Int J Digit Libr* 17, 305–338 (2016). Dostupno na: Dhttps://doi.org/10.1007/s00799-015-0156-0 . Pristupljeno: 02.03.2023

[5] ICML '04: Proceedings of the twenty-first international conference on Machine learningJuly 2004. Dostupno na: https://doi.org/10.1145/1015330.1015394 . Pristupljeno: 10.03.2023

[6] Zhang, Q., Lu, J. & Jin, Y. Artificial intelligence in recommender systems. *Complex Intell. Syst.* 7, 439–457 (2021). Dostupno na: https://doi.org/10.1007/s40747-020-00212-w . Pristupljeno: 11.03.2023

[7] Kim, SW., Gil, JM. Research paper classification systems based on TF-IDF and LDA schemes. Hum. Cent. Comput. Inf. Sci. 9, 30 (2019). Dostupno na: https://doi.org/10.1186/s13673-019-0192-7 . Pristupljeno: 25.02.2023

[8] Ma, K. (2016). Content-based Recommender System for Movie Website (Dissertation). Dostupno na: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-188494 . Pristupljeno: 27.03.2023

[9] Amin, S.A., Philips, J., Tabrizi, N. (2019). Current Trends in Collaborative Filtering Recommendation Systems. In: Xia, Y., Zhang, LJ. (eds) Services – SERVICES 2019. SERVICES 2019. Lecture Notes in Computer Science(), vol 11517. Springer, Cham. Dostupno na: https://doi.org/10.1007/978-3-030-23381-5_4 . Pristupljeno: 14.04.2023

[10] Wang, D., Liang, Y., Xu, D., Feng, X., & Guan, R. (2018). A content-based recommender system for computer science publications. Knowledge-Based Systems, 157, 1–9. doi:10.1016/j.knosys.2018.05.001

[11] Zarei, M. R., & Moosavi, M. R. (2019). A Memory-Based Collaborative Filtering Recommender System Using Social Ties. 2019 4th International Conference on Pattern Recognition and Image Analysis (IPRIA). doi:10.1109/pria.2019.8786023

[12] Pujahari, A., & Sisodia, D. S. (2020). Model-Based Collaborative Filtering for Recommender Systems: An Empirical Survey. 2020 First International Conference on Power, Control and Computing Technologies (ICPC2T). doi:10.1109/icpc2t48082.2020.9071454

[13] Do, M. P. T., Nguyen, D. V., & Nguyen, L. (2010, August). Model-based approach for collaborative filtering. In 6th International conference on information technology for education (pp. 217-228).

[14] Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN Model-Based Approach in Classification. Lecture Notes in Computer Science, 986–996. doi:10.1007/978-3-540-39964-3_62

[15] Taunk, K., De, S., Verma, S., & Swetapadma, A. (2019). A Brief Review of Nearest Neighbor Algorithm for Learning and Classification. 2019 International Conference on Intelligent Computing and Control Systems (ICCS). doi:10.1109/iccs45141.2019.9065747

[16] Abu Alfeilat HA, Hassanat ABA, Lasassmeh O, Tarawneh AS, Alhasanat MB, Eyal Salman HS, Prasath VBS. Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. Big Data. 2019 Dec;7(4):221-248. doi: 10.1089/big.2018.0175. Epub 2019 Aug 14. PMID: 31411491.

[17] Zhang, Y. (2022). An Introduction to Matrix factorization and Factorization Machines in Recommendation System, and Beyond. ArXiv, abs/2203.11026.

[18] Bell, R. M. and Koren, Y. (2007). Lessons from the netflix prize challenge. Acm Sigkdd Explorations Newsletter, 9(2):75–79.

[19] Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). Recommender systems: an introduction. Cambridge University Press.

[20] Jiawei, Zhang. (2019). Gradient Descent based Optimization Algorithms for Deep Learning Models Training.

[21] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263.

[22] Bokde, D., Girase, S., & Mukhopadhyay, D. (2015). Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey. Procedia Computer Science, 49, 136–146. doi:10.1016/j.procs.2015.04.237

6. Popis slika

7. Popis tablica

8. Popis isječka koda

8. Sažetak

Tema ovog rada bila je izrada sustava preporuke za turističku agenciju sa svhom unapriđenja poslovanja i poboljšanja usluge koji pruža.

Kroz rad su objašnjene vrste sustava za preporuku, kako rade te vrste podakata s kojima se susrečemo. Objašnjene su tehnike koje svaka vrsta sustava koristi kako bi došli do rezultata. Također prođene su metode testiranja kvalitete modela.
Na kraju je detaljno objašnjena implementacija jednog takvog sustava, a njegov rad je demonstriran kroz korisničko sučelje.

Ključne riječi: sustav preporuke, kolaborativno filtriranje, matrična faktorizacija, stohastični gradijentni spust, turizam

9. Abstract

The topic of this paper was the creation of a recommendation system for a travel agency with the aim of business and service improvements.

The paper explains the types of recommendation systems, how they work and the types of data we work with. Also, we explain techniques that each type of system uses to achieve results and methods of testing the quality of the model.
At the end, the implementation of such a system is explained in detail, and its operation is demonstrated through the user interface.

Keywords: recommendation system, collaborative filtering, matrix factorization, stochastic gradient descent, tourism