УНИВЕРЗИТЕТ У НИШУ
ЕЛЕКТРОНСКИ ФАКУЛТЕТ
Катедра за рачунарство

Имплементација система за давање препорука коришћењем DuckDB базе података

Завршни рад
Студијски програм: електротехника и рачунарство
Модул: рачунарство и информатика

Студент:
Ненад Павловић, бр. инд. 18319

Ментор:
Проф. др Александар Станимировић

Ниш, октобар 2024. година

Универзитет у Нишу
Електронски Факултет

Имплементација система за давање препорука коришћењем DuckDB базе података

Implementation of a recommendation system using DuckDB database

Завршни рад

Студијски програм: Електротехника и рачунарство

Модул: Рачунарство и информатика

Студент: Ненад Павловић бр. инд. 18319

Ментор: Проф. др Александар Станимировић

Задатак: *Упознати се појмом и основним концептима система за давање препорука. Детаљно проучити могућности примене DuckDB у имплементацији система за давање препорука. У практичном делу рада, коришћењем DuckDB базе података, имплементеирати прототип система за давање препорука на примеру једноставне e-commerce апликације.*

Датум пријаве рада: _____

Датум предаје рада: _____

Датум одбране рада: _____

Комисија за оцену и одбрану:

1. _____
2. _____
3. _____

Ниш, октобар 2024. година

Садржај

Ниш, октобар 2024. година

1. Увод

У данашњем свету где свака врста пословања налази неку примену и везу са интернетом, електронска трговина је постала један од најкоришћенијих услуга које нуди интернет. Уместо да корисници морају да иду лично у продавнице, могу из топлине својих домова да пазаре неопходне производе. E-commerce апликације пружају лак приступ широком спектру производа и услуга. Баш због олакшавања процеса куповине производа све више корисника користе могућност електронске трговине чиме се број корисника и трансакција драстично повећао. Праћењем понашања корисника, генеришу се велике количине искористивих информација у виду претрага, куповина и оцењивања. Велики изазов је постао да се нађе начин да се те информације искористе како би се побољшало корисничко искуство тиме што би се нудиле персонализоване препоруке производа самим корисницима.

Доказано је да уколико се корисницима нуде препоруке на основу њихових интересовања повећава се вероватноћа куповине и повратка корисника на исти сајт, при чему су и корисници много задовољнији. Међутим, имплементација овакве врсте система није лака и доноси велики број изазова, нарочито када је број података који треба обрадити велики. Систем мора да прати корисничко понашање константно како би сакупио што више информација о њему како би повећао прецизност и тачност препорука.

У овом раду биће обрађена тема имплементације система за давање препорука у e-commerce апликацији коришћењем DuckDB. DuckDB је алат који има могућност да ефикасно обрађује велике количине података. Његова специфична архитектура обезбеђује ефикасно извршавање аналитичких упита и обраду података из различитих извора као што су CSV и Parquet датотеке. Поред рада са подацима из датотека омогућена је и директна обрада података из других складишта података (нпр. релационих база података). Због тога је DuckDB добар избор за генерисање препорука у e-commerce апликацијама.

Циљ овог рада је да се прикажу могућности коришћења DuckDB за развој система за генерисање препорука у оквиру e-commerce апликације, са фокусом на персонализацију препорука производа на основу сличности производа и историје куповине односно профила корисника. Практични део рада обухвата имплементацију e-commerce апликације у којој се DuckDB база података у комбинацији са PostgreSQL базом података користи за чување и обраду података о корисницима и трансакцијама. Примењени су неки од популарних алгоритама за давање препорука који корисницима треба да препоруче производе за које постоји велика вероватноћа да су интересантни.

Друго поглавље рада биће посвећено теоријским концептима система за давање препорука, при чему ће се детаљније обрадити алгоритми који се користе приликом генерисања препорука. Треће поглавље ће се бавити DuckDB базом података, њеном архитектуром, предностима и начином коришћења. У четвртом поглављу биће приказана имплементација система за давање препорука у оквиру e-commerce апликације. Пето поглавље ће приказати тест резултате система. Шесто и последње поглавље ће у кратким цртама сумирати рад и нагласити најбитније ставке у њему.

2. Системи за давање препорука

2.1. Увод у системе за давање препорука

2.1.1. Дефиниција система за давање препорука и кратак преглед историје

Системи за давање препорука су информационо-филтрирајући алати који помажу корисницима да пронађу релевантне производе или информације у великим базама података. Главна функција ових система је генерисање садржаја према корисничким преференцијама. Пример таквог система нуди Amazon који аутоматски препоручује производе кориснику базиране на предходним куповинама, оценама и интеракцијом корисника са производима. [2][3]

Такође се систем за давање препорука може посматрати и као страшки алат за доношење одлука у системима који обрађују велику количину података. Генерална идеја је омогућити да систем предпостави шта би се кориснику свидело на основу информација које су складиштене о њему, практично треба препоручити производе које корисник није видео, а требало би. Уколико се као пример узме online књижара, треба препоручити књиге које корисник није купио а јако су сличне са онима које је пазарио. [1]

Порастом корисника и употребе интернета крајем деведесетих година креће и развој првих система за давање препорука. Као и сваки други пројекат који је у развоју, тако и системи за давање препорука нису били савршени, били су јако прости и препоручивали су корисницима најпродаваније производе. Временом су схватили да таква врста препорука не доприноси много повећању продаје и да би систем био много кориснији да се кориснику препоручују њему занимљиви производи, а не глобално занимљиви производи. Због новонасталих потреба имплементиране су три главне технике које се данас користе: колаборативно филтрирање, филтрирање по садржају и хибридне методе. [3][4]

2.1.2. Разлози за имплементацију система за препоруке у e-commerce апликацијама

Имплементација система за препоруке у e-commerce апликацијама нуди бројне предности које директно утичу на раст продаје, унапређење корисничког искуства и оптимизацију пословања. Ове предности укључују повећање прихода, лојалности корисника и ефикасније трошење маркетиншких ресурса.

1. Повећање продаје

Као што је већ наглашено системи за давање препорука се креирају да би се повећала количина продаје производа. Анализе и истраживања тржишта креирали су податак да коришћење квалитетних система за давање препорука повећава број купаца за чак

22,66% [5]. Такође постојећи купци могу докупити додатни производ ако систем препоручи производ за који предпоставља да ће се свидети кориснику. [4]

2. Побољшање корисничког искуства и лојалности

Системи за давање препорука доприносе побољшању корисничког искуства тиме што скраћују време претраге и олакшавају корисницима проналажење жељених производа. Препоруке прилагођене преференцијама корисника не само да повећавају задовољство, већ и подстичу лојалност, јер се корисници враћају на платформе које боље разумеју њихове потребе. Корисник ће пре изабрати платформу на којој ће брже и лакше задовољити своје жеље. [6]

3. Унапређење пословања и смањење трошкова компаније

Обзиром да се обрађују подаци о корисницима и њиховим преференцијама, компаније имају увид у производе које занимају већину корисника. Са тим знањем компаније знају у ком правцу треба да иде њихово пословање како би се продаја повећавала односно који тип производа требају више набављати и продавати а који мање. [5]

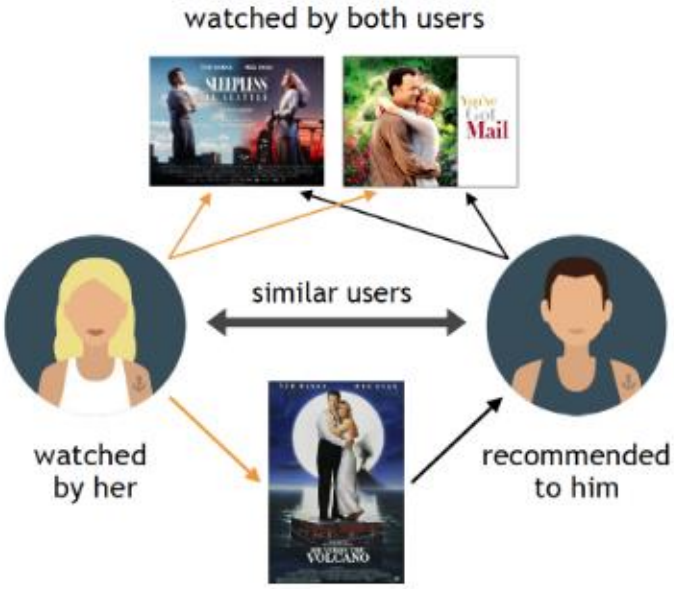
Ови разлози јасно указују на важност имплементације система за препоруке у модерним e-commerce апликацијама, где персонализација и ефикасност постају кључни фактори у одржавању конкурентности и задовољства корисника.

2.2. Типови система за давање препорука

Системи за давање препорука могу бити имплементирани коришћењем различитих алгоритама, у зависност од потреба компаније и од врста података са којима раде. Постоје три главна приступа у имплементацији система за давање препорука, а то су колаборативно филтрирање, филтрирање по садржају и хибридне методе.

1. Колаборативно филтрирање (енгл. Collaborative filtering)

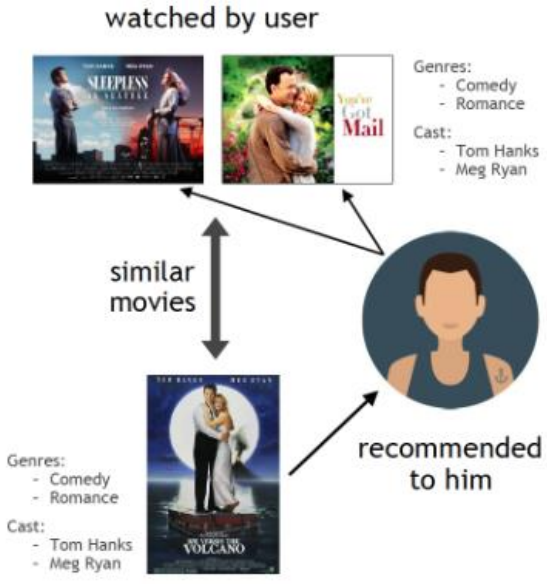
Колаборативно филтрирање је један од најранијих и најчешће коришћених приступа у системима за препоруке. Модел се заснива на идеји да се на основу корисничких преференција и понашања генеришу препоруке. Колаборативно филтрирање (слика 1) се може поделити на два приступа: базиран на меморији (memory-based) и базиран на моделу (model-based). Колаборативно филтрирање базирано на меморији је метод који примењује различите алгоритме над подацима које систем поседује о корисничкој историји, док с друге стране колаборативно филтрирање базирано на моделу је приступ који користи машинско учење за креирање модела који предвиђају преференције корисника. Постоје две врсте memory-based колаборативног филтрирања колаборотивно филтрирање на основу корисника (енгл. user-based filtering) и колаборативно филтрирање на основу производа (енгл. content-based filtering). У даљем раду изоставиће се реч memory-based јер се рад неће освртати на model-based приступ. [1][4]



Слика 1. Пример колаборативног филтрирања [5]

2. Филтрирање по садржају (енгл. Content-based Filtering)

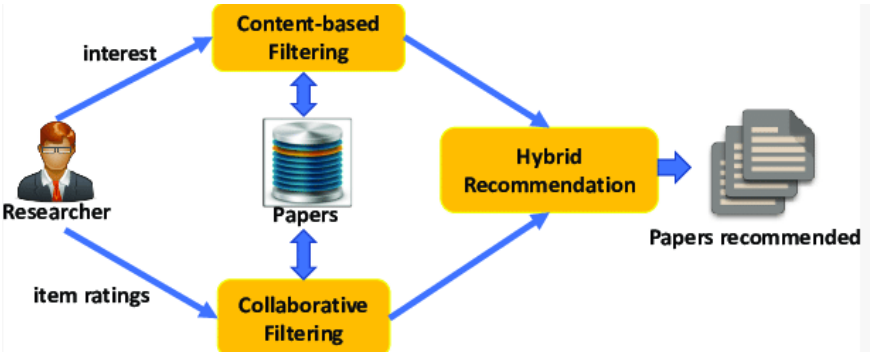
Филтрирање по садржају (слика 2) представља технику која анализира историју корисника, концентришући се на производе са којима је корисник имао интеракцију у виду куповина или оцењивања. Проналази сличне производе анализираним производима на основу сличности у њиховим карактеристикама, описима, атрибутима. Модел није толико сложен за имплементацију али има великуману да нуди само сличне производе већ купљеним и онемогућује корсинику откривање нових типова производа. [4]



Слика 2. Пример филтрирања по садржају [5]

3. Хибридне методе (енгл. hybrid methods)

Хибридне методе (слика 3) настале су као комбинација предходних техника. Комбиновањем метода добијају се прецизније препоруке и повећава се задовољство крајњих корисника. Такође хибридним методама смањују се недостаци појединачних метода и комбинују се предности. [1]



Слика 3. Пример хибридних метода [7]

2.3. Колаборативно филтрирање

Колаборативно филтрирање је један од најефикаснијих и најпопуларнијих приступа у системима за давање препорука, и имају велику примену у e-commerce апликацијама. Ова метода се базира на анализи активности корисника, тачније речено на основу података које систем поседује о кориснику генерише препоруке. За разлику од неперсонализованих препорука које се ослањају на најпопуларније производе, колаборативно филтрирање нуди персонализоване препоруке прилагођене сваком кориснику појединачно. [2]

Главна идеја колаборативног филтрирања је складиштење и анализа података о оценама и куповинама корисника. Систем користи ове податке да пронађе кориснике са сличним интересовањима или производе сличне онима за које је корисник већ показао интересовање. На основу тих информација, генеришу се препоруке које одговарају специфичним потребама и преференцијама корисника. [3]

Колаборативно филтрирање развијено је почетком деведесетих година и брзо је постало стандард у системима за давање препорука. Као стандард у колаборативном филтрирању устаљиле су се две технике: колаборативно филтрирање засновано на кориснику и колаборативно филтрирање засновано на производу. [4]

2.3.1. Колаборативно филтрирање засновано на кориснику

Ова метода се базира на идеји да се корисничке препоруке генеришу на основу сличности међу корисницима. Главна сврха овог приступа је да идентификује кориснике са сличним интересовањима како би се предвидело шта би тренутни корисник могао да воли на основу преференција њему сличних корисника. [2]

Кориснички профил се користи као основа за препоруке. Када корисник користи систем, његов профил се упоређује са профилима других корисника како би се пронашли „суседи" који су имали сличне преференције у прошлости. На основу оцена које су ти слични корисници дали различитим производима, систем може да процени које производе би тренутни корисник могао да воли, а које још није видео. [3]

Техника колаборативног филтрирања се може демонстрирати кроз један једноставан пример. Слика 4 приказује стање демонстративне базе података у којој се памте подаци који показују како су корисници оцењивали различите производе.

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Слика 4. Тест подаци [2]

Табела (слика 4) приказује да корисница Алиса није оценила производ под редним бројем пет. Како би се одредило да ли је потребно препоручити производ пет Алиси неопходоно је наћи сличност између Алисе и осталих корисника. На основу оцена сличних корисника могуће је предпоставити да ли производ треба препоручити Алиси или не.

Рачунање сличности између два корисника зависи од метрике која се користи, што ће детаљно касније бити описано. У описаном примеру користити се Пирсов коефицијент корелације (слика 5).

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}}$$

Слика 5. Формула Пирсовог коефицијента корелације [2]

Ознаке коришћене у формули за Пирсов коефицијент корелације (слика 5) су:

- $U = \{u_1, ..., u_n\}$ – скуп корисника,
- $P = \{p_1, ..., p_m\}$ – скуп производа,
- R оцена r_{ij}, где је $i \in \{1, ..., n\}$, а $j \in \{1, ..., m\}$
- $\overline{r_a}$ и $\overline{r_b}$ одговарају просечним оценама корисника респективно

Применом формуле за Пирсов коефицијент корелације на податке из табеле (слика 4) добија се сличност Алисе са осталим корисницима (сличност за кориснике од User1 do User4 је респективно: 0.85, 0.7, 0.0, -0.79). Овим је установљено да је Алиса слична корисницима User1 и User2. Ако се словом N означе корисници слични Алиси, по формули sa slike 6, се рачуна претпостављена оцена којом би Алиса оценила производ пет, и то би била оцена 4.85.

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

Слика 6. Формула за рачунање предпостављене оцене [2]

На слици 7 дат је пример имплементације колаборативног филтрирања заснованог на кориснику у програмском језику Python. Имплементирани пример даје препоруке филмова кориснику на основу оцена које су филмовима дали слични корисници.

```
avg_ratings = movies_ratings_df.groupby('movie_name').agg(avg_rating = ('rating', 'mean'),
    number_of_ratings = ('rating', 'count')).reset_index()
avg_ratings100 = avg_ratings[avg_ratings['number_of_ratings']>100]
avg_ratings100.sort_values(by='number_of_ratings', ascending=False).head()
```

Слика 7. Најпопуларнији филмови [9]

Применом кода са слике 7 добијају се најпопуларнији филмови, односно филмови који имају највећи број оцена и сортирају се по оцени.

```
merged_df=pd.merge(movies_ratings_df,avg_ratings100[['movie_name']],on='movie_name', how='inner')
merged_df.pivot
```

Слика 8. Најпопуларнији филмови са њиховим оценама [9]

Применом кода са слике 8 добијају се најпопуларнији филмови као и њихове оцене. Функцијом pivot генерише се матрица чији су редови корисници, колоне називи филмова, а у пресеку редова и колона налази се оцена коју је тај корисник дао филму.

```
similarity_matrix = cosine_similarity(movies_ratings_pivot)
similarity_matrix_df=pd.DataFrame(similarity_matrix,index=movies_ratings.pivot.index,
    columns=movies_ratings_pivot.index)
```

Слика 9. Сличност корисника [9]

Применом кода са слике 9 израчуната је сличност између корисника коришћењем косинусне сличности која ће касније бити детаљно објашњена. Такође је направљена матрица чије колоне и врсте представљају кориснике и у њиховим пресецима се налази сличност између корисника.

```
select_userid = 5
similarities = similarity_matrix_df[select_userid].drop(select_userid)
weights = similarities/similarities.sum()
n =10
user_similarity_threshold =  0.5
similar_users=similarity_matrix_df[similarity_matrix_df[select_userid] >
    user_similarity_threshold][select_userid].sort_values(ascending=False)[:n]
```

Слика 10. Слични корисници кориснику пет [9]

Применом кода са слике 10 добијени су слични корисници кориснику са идентификатором пет при чему је граница сличности постављена тако да су корисници слични ако им је косинусна сличност већа од 0.5.

```
not_watched_movies =movies_ratings_pivot.loc[movies_ratings_pivot.index !=
    select_userid, movies_ratings_pivot.loc[select_userid,:]==5]
similarities = similarity_matrix_df[select_userid].drop(select_userid)
weights = similarities/similarities.sum()
weighted_averages= pd.DataFrame(not_watched_movies.T.dot(weights.to_numpy()),
    columns=["weighted_avg"])
```

Слика 11. Препоручени филмови [9]

Применом кода са слике 11 добијени су филмови које корисник са идентификатором пет није гледао, а њему слични корисници јесу и сортирани су по просечним оценама сличних корисника.

2.3.2. Колаборативно филтрирање засновано на производу

Колаборативно филтрирање засновано на производу представља један од најкоришћенијих метода у системима за давање препорука, због своје једноставније имплементације и генерисања довољно тачних препорука. За разлику од колаборативног филтрирања заснованог на кориснику, колаборативно филтрирање засновано на производу препоручује производе на основу сличности међу производима. Главна идеја је да се за сваког корисника предлажу производи који су слични онима са којима је корисник већ имао интеракцију (оценио, купио или коментарисао). Овај приступ се посебно истиче у e-commerce апликацијама јер омогућава оптимизовану обраду великих база података кроз обраду информација о сличностима међу производима. То омогућава системима да дају препоруке у реалном времену, чак и када је у питању огромна количина података. [2][10]

Идеја колаборативног филтрирања заснованог на производу се може показати на једноставном примеру, користе се исти тест подаци као у примеру са колаборативним филтрирањем заснованим на кориснику, односно подаци са слике 5. Обзиром да Алиса није још оценила производ са идентификатором пет, предпоставка је да се она са њим још није сусретала и да се треба испитати да ли је тај производ сличан производима које је Алиса оценила. За утврђивање сличности између произода користи се косинусна сличност која ће касније бити детаљније објашњена. Уколико се са $\vec{a}$ i $\vec{b}$ означе векторе који одговарају оценама два производа, а са $|\vec{a}|, |\vec{b}|$ еуклидске дужине вектора, у том случају косинусну сличност тих два производа рачуна по формули са слике 12.

$$\mathrm{sim}(\vec{a}, \vec{b}) = \frac{\vec{a} * \vec{b}}{|\vec{a}| * \overrightarrow{|b|}}$$

Слика 12. Косинусна сличност [2]

Применом формуле (слика 12) на производе један и пет добија се вредност 0.99, међутим у пракси се поред обичне косинусне сличности користи прилагођена косинусна сличност због проблема у коме неки корисници могу да оцењују много блаже производе, а неки много строже. Идеја прилагођене косинусне сличности је да се пре одређивања

угла одузме просечна оцена којом је дати корисник оцењивао остале производе. Формула по којој се израчунава прилагођена косинусна сличност приказана је на слици 13.

$$sim(a,b) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}$$

Слика 13. Формула за рачунање прилагођене косинусне сличности [2]

Резултат који се добије када се рачуна прилагођена косинусна сличност између производа један и пет је 0.80. Када се пронађу сви производи који су слични производу, могуће је одредити предпостављену оцену којом би корисник оценио производ. Формула којом се рачуна предпостаљена оцена којом би корисник оценио производ p је приказана на слици 14.

$$pred(u,\ p) = \frac{\sum_{i \in ratedItems(u)} sim(i,\ p) * r_{u,i}}{\sum_{i \in ratedItems(a)} sim(i,\ p)}$$

Слика 14. Формула за рачунање предпостављене оцене [2]

На слици 15 дат је пример имплементирања колаборативног филтрирања на основу производа у програмском језику python.

```
df = pd.read_csv(train_path, delimiter = '\t', names = ['userid', 'itemid', 'rating', 'timestamp'])
utility = df.pivot(index = 'itemid', columns = 'userid', values = 'rating')
utility = utility.fillna(0)
utility.head()
```

Слика 15. Учитавање датотека и креирање матрице[11]

Применом кода са слике 15 учитавају се подаци и креира се матрица чији су редови производи, колоне корисници и у пресецима колона и врста се налазе оцене које је одговарајући корисник дао одговарајућем производу.

```
distance_mtx = squareform(pdist(utility, 'cosine'))
similarity_mtx = 1- distance_mtx
```

Слика 16. Матрица косинусне удаљености [11]

Код са слике 16 генерише матрицу која представља косинусну удаљеност производа и на основу те матрице креира се матрица сличности између производа.

```
user_to_predict = 2
item_to_predict = 3
user2_ratings = utility.iloc[:,user_to_predict-1]
item3_similarity = similarity_mtx[item_to_predict-1]
```

Слика 17. Слични производи проиводу три [11]

Да би се видело коју оцену би корисник са идентификатором два дао проиводу са идентификатором три налазе се оцене које је оцене корисник два дао осталим

производима и налазе се слични производи производу три, што се постиже кодом са слике 17.

```
numerator = np.dot(user2_ratings,item3_similarity)
denom = item3_similarity[user2_ratings > 0].sum()
prediction = numerator / denom
```

Слика 18. Финална предикција [11]

Кодом са слике 18 добија се финална предикција. Резултат се добија када се поделе скаларни производ оцена које је корисник дао другим производима и сличност тих проивода са производом три са сумом сличности проивода које је корисник са идентификатором два већ оценио.

2.3.3. Метрике које се користе у колаборативном филтрирању

Метрике се користе у процесу одређивања сличности између корисника у колаборативном филтрирању. Постоји мноштво метрика, у алгоритмима се бирају на основу конкретног проблема. Три најпознатије метрике су косинусна сличност, већ раније помињани Пеарсонов коефицијент корелације и Јакардов коефицијент.

1. Косинусна сличност

Косинусна сличност се примењује приликом утврђивања сличности два корисника на начин да се од њихових оцена производа креирају вектори након чега се мери косинус угла између тих вектора. Уколико су вектори ортогонални, сличност између кроисника је 0 (не постоји), док ако су вектори паралелни онда је сличност између корисника 1 (максимална вредност). Косинусна сличност се углавном користи код колаборативног филтрирања заснованог на производима. Уколико се са xi и yj означе оцене корисника за исти производ тада се косинусна сличност рачуна по формули са слике 19. [10]

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

Слика 19. Формула косинусне сличности [10]

2. Пеарсонов коефицијент корелације

Пеарсонов коефицијент се примењује приликом утврђивања сличности између два корисника на начин да се мери линеарна зависност између два сета оцена. Идеја је да се мери зависност у променама оцене једног корисника у зависности на промене оцене другог корисника. Користи се да би се утврдиле различитости у оцењивању истог производа од стране два корисника у ситуацијама када један корисник један производ оцени вишом оценом, а други нижом оценом. Код Пеарсоновог коефицијента корелације максимална сличност има вредност 1, док потпуна различитост има вредност -1. Углавном се користи код колаборативног филтрирања заснованом на кориснику. Уколико се са r_{ai} и r_{bi} означе оцене различитих корисника за исти производ а са $\bar{r}_a$ и $\bar{r}_b$

просечне оцене корисника онда се Пеарсонов коефицијент рачуна по формули са слике 20. [10]

$$PCC = \frac{\sum (r_{ai} - \bar{r}_a)(r_{bi} - \bar{r}_b)}{\sqrt{\sum (r_{ai} - \bar{r}_a)^2} \sqrt{\sum (r_{bi} - \bar{r}_b)^2}}$$

Слика 20. Формула Пеарсоновог коефицијента сличности [10]

3. Јакардов коефицијент

Јакардов коефицијент се примењује приликом утврђивања сличности између два корисника када у систему оцене корисника нису нумеричке већ бинарне (уместо оцене 5, оцена је описана текстуално). Главна идеја је да се мери сличност између два асиметрична бинарна вектора (мерење сличности између два скупа). Резултат Јакардовог коефицијента је исти као код косинусне симетричности односно 1 представља потпуну сличност, а 0 потпуну различитост. Уколико се са А и Б означе скупови елемената које су два различита корисника оценили позитивно, тада се Јакардов коефицијент рачуна по формули са слике 21. [26]

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Слика 21. Формула Јакардовог коефицијента [26]

2.3.4. Проблеми са којима се суочава колаборативно филтрирање

И поред мноштво врлина које колаборативно филтрирање има, такође пати и од мана са којима мора да се суочава. Те мане и проблеми утичу на тачност и ефикасност препорука које колаборативно филтрирање пружа крајњим корисницима. Главни проблеми су:

1. Проблем хладног старта

Проблем хладног старта јавља се када систем не може да генерише ефикасне препоруке због недостатка података. Овај проблем је нарочито изражен код нових корисника или нових производа, где систем нема довољно информација (корисничких преференција и оцена) како би дао одговарајуће препоруке. Постоји неколико начина да би се пребродили ови проблеми [8]:

- Једно од решења је да се од корисника приликом регистрације тражи да унесе одређене преференције или да оцени неке производе како би систем имао довољно података на основу којих може да кориснику понуди одговарајуће препоруке.

- Прикупљање демографских података, као што су године, пол или интересовања корисника, може помоћи у давању одговарајућих препорука у почетним фазама коришћења система.

2. Пробем реткости података

Проблем реткости података јавља се када активни корисници оцене само мали број производа. У великим e-commerce системима, где корисници обично оцењују само мали проценат производа, систем може давати неквалитетне препоруке услед недостатка података. Неке од техника за превазилажење овог проблема укључују [8]:

- Демографско филтрирање – коришћење информација о корисницима као што су старост, пол или интересовања, како би се употпунили подаци.
- Singular Value Decomposition (SVD) – техника смањења димензионалности која омогућава боље коришћење доступних података у случају малог броја оцена.
- Model-based технике – алгоритми који користе моделе, попут кластеровања, за груписање корисника или производа ради генерисања прецизнијих препорука чак и у условима ретких података.

3. Проблем скалабилности

Са порастом броја корисника и производа, долази до проблема скалабилности. У системима са великим бројем корисника и производа, генерисање препорука у реалном времену постаје тешко због потребе за обрадом велике количине података, па су самим тим алгоритми спори. Два уобичајена приступа за решавање овог проблема су [8]:

- Смањење димензионалности – методе попут SVD могу се користити за смањење комплексности података, омогућавајући бржу обраду.
- Технике засноване на кластеровању – уместо да се претражује цела база података, корисници се могу груписати у мање кластере како би се брже пронашле препоруке за специфичне групе.

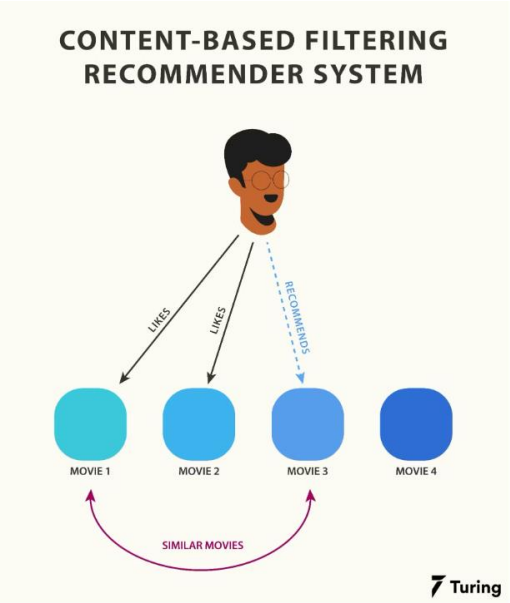
2.4. Филтрирање по садржају

Филтрирање по садржају је техника која се фокусира на карактеристике производа са којима је корисник имао интеракцију. Филтрирање по садржају се базира на идеји да ако је корисник показао интересовање за производ који садржи одређену вредност неког атрибута (на пример ако су у питању филмови, и ако је корисник љубитељ акција) да се кориснику нуде производи који такође садрже исту вредност датог атрибута (слика 22). [2]

За разлику од колаборативног филтрирања, филтрирање по садржају се базира искључиво од корисничког профила и његових преференција, не узимају се у обзир други корисници. Кориснички профил се креира на основу историје активности корисника, тј. креира се на основу производа које је корисник на неки начин означио као

фаворите (купио их је, фаворизовао…). Због изолованости само на личне преференције филтрирање по садржају нема проблема, уколико не постоје информације о активностима других корисника и потпуно правилно функционише и уколико у систему постоји само један корисник. Такође је отпоран и на проблеме хладног старта у односу на нове производе, јер уколико су они слични фаворизованим производима корисника, могу бити препоручени кориснику. [12]

Карактеристике производа могу бити издвојени атрибути, али исто тако могуће да производ садржи само опис. Технике филтрирања по садржају се често примењују на текстуалне препоруке, као што су чланци или web странице. Систем анализира кључне речи и друге релевантне карактеристике предмета како би пронашао сличности са предметима које је корисник раније високо оценио. Иако ова метода захтева познавање карактеристика предмета, она пружа флексибилност у персонализацији и прилагођавању препорука према преференцијама корисника које се временом мењају. [2]



Слика 22. Пример филтрирања по садржају [13]

2.4.1. Представљање садржаја и рачунање сличности садржаја

Најједноставнији начин описивања производа била би да се сваки производ описује скупом атрибута. Вредности би могле да се складиште у базама података и на основу њих би се проналазили слични производи. Препоруке се затим креирају анализом купљених производа од стране корисника и упоређивањем вредности њихових атрибута са вредностима осталих производа. Креирање корисничког профила могуће је обавити на два начина. Први начин би представљао ручно уношење преференција корисника приликом регистровања, што се користи да би се решио проблем хладног старта. Други начин био би аутоматизовано генерисање преференција корисника на основу његове историје. [2]

За утврђивање сличности између предмета користе се различите функције. Једноставан приступ би био проверавање да ли је производ описан неким атрибутом

задовољава корисничке преференције односно да ли се проверавани и фаворизовани производ поклапају по вредности атрибута (да ли су оба филма акције, уколико је реч о филмовима). Сличност би могла бити 0 (не постоји преклапање) или 1 (потпуно преклапање). Међутим, систем може користити и сложеније метрике, као што је Dice коефицијент, који мери сличност на основу преклапања скупа кључних речи између различитих предмета [2]. За два производа описаних скупом кључних речи Dice кеофицијент сличности рачуна се на основу формуле са слике 23.

$$\frac{2 \times |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

Слика 23. Формула Dice-овог коефицијента сличности [2]

Међутим у неким случајевима није могуће описати производ скупом атрибута, на пример уколико се креира систем за препоруке књига информације попут аутора и издавача нису део директног садржаја књиге, али су значајни за описивање садржаја. Па се због тога садржај често представља као скуп кључних речи које су аутоматски генерисане из текста или описа. На основу кључних речи документ се може анализирати у облику вектора у вишедимензионалном простору. Сваки документ може бити представљен бинарним вектором, где 1 означава присуство одређене кључне речи, а 0 њено одсуство. Мана овог приступа је игнорисање учесталости појављивања речи и дужину документа. [2]

Да би се превазишли ти проблеми, користи се TF-IDF метода. TF (енгл. term frequency) означава колико често се одређена реч појављује у документу, при чему се у обзир узима и дужина документа. IDF (енгл. inverse document frequency) смањује тежину оних речи које се често појављују у свим документима (везника, речца, прилога…). Комбиновањем ова два параметра, TF-IDF трансформише документ у вектор који описује његов садржај, смањујући утицај уобичајених речи и истичући оне које су специфичне и битне за препоруку. Као резултат, систем за давање препорука може упоредити векторе корисничког профила и векторе производа са којима корисник није имао додир како би генерисао препоруке базиране на сличностима у садржају истих. [2]

На слици 24 приказан је пример имплементације филтрирања по садржају у програмском језику python.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
item_data = pd.read_csv('item_data.csv')
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(item_data['description'])
```

Слика 24. Креирање вектора садржаја производа [14]

Код са слике 24 објашњава учитавање података о производима и њихово трансформисање у векторе садржаја. Користе се библиотеке које су потребне за учитавање података и креирење вектора садржаја производа на основу TF-IDF методе споменуте раније.

```
user_interactions = [(101, 5), (102, 4), (105, 3)]
user_profile = np.zeros(tfidf_matrix.shape[1])
for item_id, rating in user_interactions:
    item_index = item_data.index[item_data['item_id'] == item_id][0]
    user_profile += tfidf_matrix[item_index].toarray()[0] * rating
```

Слика 25. Креирање корисничког профила [14]

У коду са слике 25 објашњава се симулација корисничког оцењивања и креирање корисничког профила тако што се карактеристике оцењених производа комбинују са оценом тог производа.

```
from sklearn.metrics.pairwise import cosine_similarity
similarities = cosine_similarity([user_profile], tfidf_matrix)
```

Слика 26. Косинусна сличност корисничког профила и вектора садржаја [14]

Коришћењем sklearn.metrics.pairwise библиотеке рачуна се косинусна сличност корисничког профила и вектора садржаја свих производа (слика 26).

```
recommended_item_ids = item_data['item_id'][np.argsort(similarities[0])[::-1]]
```

Слика 27. Препоручени производи [14]

На основу косинусне сличности између корисничког профила и вектора садржаја свих производа добијају се идентификатори препоручених производа, уређени у опадајућем редоследу од најсличнијих ка мање сличним производима, као што је приказано на слици 27.

2.4.2. Недостаци филтрирања по садржају

Филтрирање по садржају као и колаборативно филтрирање није савршено и има своје недостатке. Недостаци филтрирања по сарджају утичу на прецизност и ефикасност филтрирања. У наставку су наведени главни проблеми филтрирања по садржају.

1. Плитка анализа садржаја

Системи који користе филтрирање по садржају често се ослањају на текстуални опис производа, што може бити недовољно за прецизно одређивање да ли тај производ треба препоручити или не. Систем не може разликовати квалитетно написане описе од лошијих, јер се ослања само на кључне речи. Поред упитног квалитета описа производа такође могу представљати проблеме и кратки описи или предмети са мало атрибута због тога што отежавају алгоритму да направи одговарајуће корисничке преференције. Такође проблем представљају и нетекстуални подаци јер системи обично не узимају у разматрање мултимедијалне податке, што може бити кључно за препоруке. [2]

2. Превише специјализоване препоруке

Велики проблем филтрирања по садржају је што се нуде производи који су слични производима с којима је корисник већ имао интеракцију. Самим тим препоруке постају монотоне и из препорука се изостављају проиводи који нису слични досадашњим префериранim производима корисника, али који такође могу да буду занимљиви.

Технике које се користе како би се у некој мери ублажио наведени проблем су „тематске диверсификације" или убацивања „насумичних" препорука. [2]

3. Хладан старт на нивоу корсисника

Иако хладан старт на нивоу нових производа не представља проблем филтрирању по садржају, хладан старт на нивоу новог корисника негативно утиче на алгоритам. Проблем настаје кад нови корисник нема довољно оцењених производа, па је тешко креирати одговарајући кориснички профил што резултира не тако добрим препорукама. Решење овог проблема је као и код колаборативног филтрирања да се приликом регистровања новог корисника, самом кориснику наметне да оцени неколико производа како би се креирао почетни кориснички профил који ће се временом усавршити. [2]

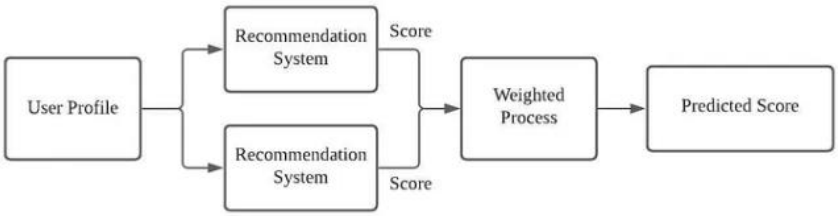
2.5. Хибридни системи за давање препорука

Хибридни системи за давање препорука су комбинација различитих техника, најчешће филтрирања по садржају и колаборативног филтрирања. Главна предност хибридног система је његова способност да искористи добре стране обе технике, чиме се постиже побољшање прецизности препорука и обезбеђује већи степен прилагодљивости потребама корисника. На пример, систем може користити филтрирање по садржају за препоруку производа сличних онима које је корисник већ оценио и колаборативно филтрирање за препоруку производа које су слични корисници оценили позитивно и да комбинацијом та два резултата генерише најадекватније препоруке. Поред идеје да се искористе само врлине обе технике, хибридни системи су настали да би се смањили проблеми које доноси свака техника појединачно. Хибридни модели за давање препорука комбинују ове две технике како би надокнадили недостатак података и побољшали перформансе система. Комбинација наведених техника у хибридном моделу може се остварити на неколико начина. На пример, систем може засебно имплементирати алгоритаме и комбиновати резултате, резултат једног алгоритма проследити као улаз другог или обрнуто. Хибридни системи за давање препорука омогућавају оптимизацију проблема попут „хладног старта", недовољне количине података и давања превише специјализованих препорука. [3]

Постоји неколико врста хибридних система, сваки од тих система има своје специфичне карактеристике и избор типа хибридног система зависи од конкретног проблема. У наставку су дати најважнији типови хибридних система за давање препорука.

1. Хибридни системи за давање препорука засновани на тежинама

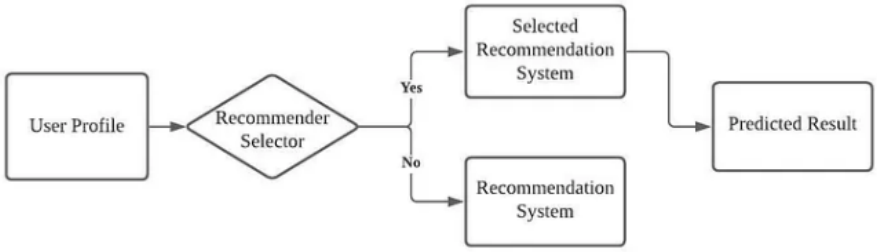
Главна идеја ове технике је комбиновање резултата различитих система за давање препорука путем специјализоване формуле која комбинује оцене које су настале као резултат појединачних техника. Тежине се могу кориговати током времена у зависности од тачности предикција. Шема хибридног система за давање препорука заснованог на тежинама приказана је на слици 28. [3]



Слика 28. Тежински хибридни систем [17]

2. Хибридни систем за давање препорука заснован на пребацивању

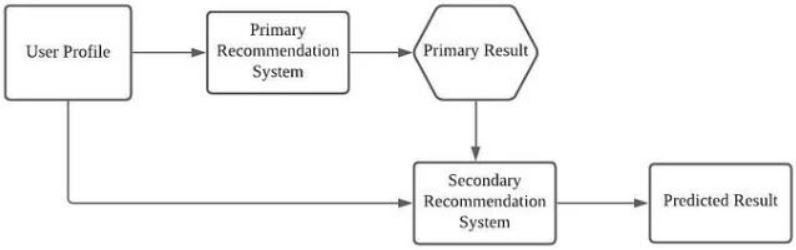
Овај приступ се базира на идеји да систем одређује када се пребацује са једне технике на другу, у зависности од тога која од појединачних техника тренутно може да пружи боље препоруке. На пример, DailyLearner [3] прво примењује филтрирање по садржају, а затим прелази на колаборативно када филтрирање по садржају не пружи довољно података. Шема хибридног система за давање препорука заснованог на пребацивању приказана је на слици 29.



Слика 29. Хибридни систем базиран на пребацивању [17]

3. Каскадни хибридни систем за давање препорука

Главна идеја ове технике је континуално побољшање препорука. Један алгоритам креира основну листу препорука, а затим се другом алгоритму прослеђује иницијална листа препорука коју он додатно унапређује. Пример система који користи ову технику је EntreeC [3], који комбинује знање и колаборативно филтрирање. Шема каскадног хибридног система за давање препорука приказана је на слици 30.

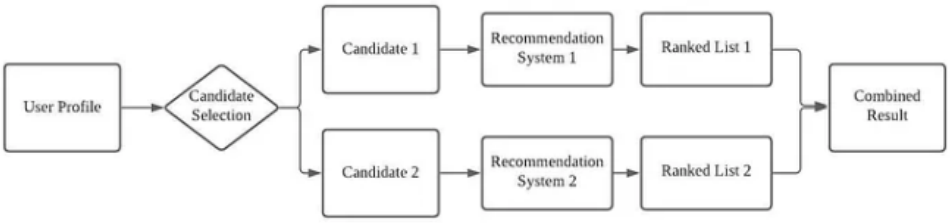


Слика 30. Каскадни хибридни систем [17]

4. Мешовити хибридни систем за давање препорука

Генерална идеја је комбиновање резултата сваке појединачне технике чиме се постиже да корисник добије више различитих препорука за исти производ. Представник

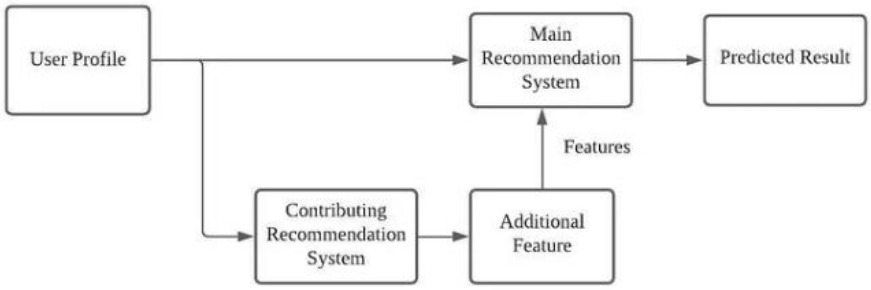
овог типа хибридног система је PTV [3] систем који користи филтрирање по садржају и колаборативно филтрирање како би креирао телевизијски распоред програма. Шема мешовитог хибридног система за давање препорука приказана је на слици 31.



Слика 31. Мешовити хибридни систем [17]

5. Комбинација карактеристика

Комбинација карактеристика је најкласичнији хибридни систем за давање препорука, јер карактеристике добијене једном техником прослеђује као улаз у алгоритам следеће технике. Представник овог система је Pipper [3] који користи оцене из колаборативног филтрирања као улаз за филтрирање по садржају. Шема хибридног модела комбинације карактеристика приказана је на слици 32.



Слика 32. Комбинација карактеристика [17]

6. Проширење карактеристика

Идеја овог типа хибридног система за давање препорука је допуњавање функционалностима резултате који су добијени од стране неког другог система за давање препорука. Главни представник је Libra [3] систем који користи препоруке добијене на основу филтрирања по садржају књига на основу података са сајта Amazon.com, користећи Bayesov класификатор текста. Шема хибридног модела проширења карактеристика приказана је на слици 33.

Слика 33. Проширење карактеристика [17]

7. Мета-ниво хибридни систем за давање препорука

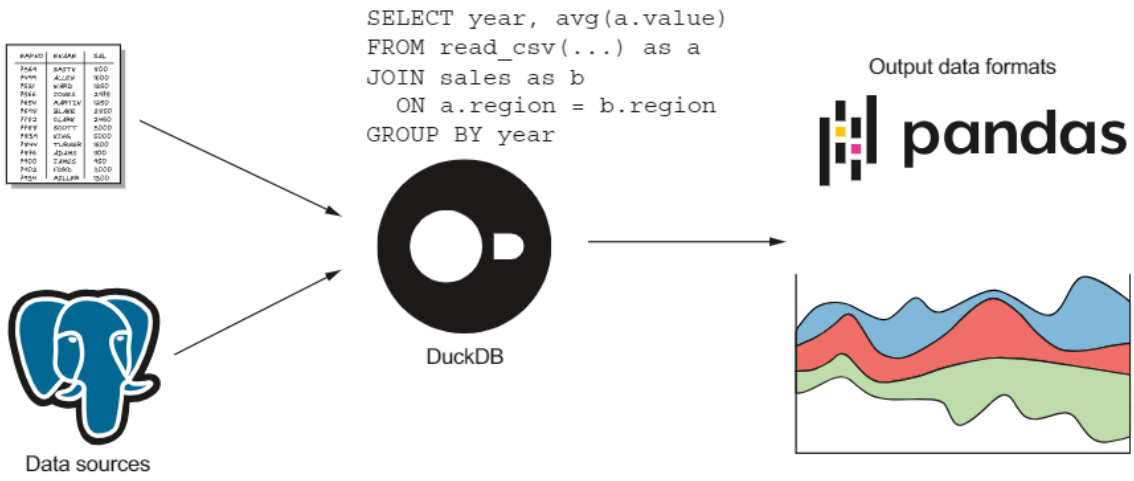
Као и већина хибридних система за давање препорука и у овом типу једна техника генерише податке које користи следећа техника, само је у овој техници у питању модел. Представник је LaboUr [3] систем који користи инстанцијско учење за креирање корисничких профила у филтрирању ро садржају, који се затим користе у колаборативном филтрирању за поређење корисника и давање препорука.

3. DuckDB аналитичка база података

DuckDB је аналитичка, уграђена, релациона база података која је дизајнирана за рад са великим количинама података. DuckDB је релативно нова база података која је брзо стекла популарност због своје једноставности и ефикасности у аналитичким упитима. Први пут је представљена 2019. године, а њен циљ је да реши проблеме с којима се суочавају традиционалне базе података при обради великих количина података у оквиру аналитичких задатака. DuckDB је често упоређиван са SQLite-ом, јер се обе базе фокусирају на минималну инсталацију и лакоћу коришћења, али за разлику од SQLite-а, DuckDB је оптимизован за онлајн аналитичку обраду података (OLAP). [19][27]

Једна од кључних предности DuckDB-а је то што нема екстерне зависности, што значи да не захтева комплексне инсталације или одржавање сервера. Све се одвија унутар хост процеса, што олакшава интеграцију и побољшава брзину преноса података. DuckDB је посебно дизајниран за обраду великих аналитичких упита, као што су агрегација и спајање табела. Његова архитектура омогућава ефикасно извршавање упита захваљујући колонарно-векторизованом механизму за извршавање упита, што га чини бржим у поређењу са традиционалним базама података као што су PostgreSQL или MySQL. Такође DuckDB је бесплатан и отвореног кода што га поред осталих предности чини јако пожељним алатом. [19][27]

DuckDB омогућава обраду података који су складиштени на различите начине. DuckDB има подршку за учитавање CSV, JSON, Parquet и Apache Arrow датотека, али исто тако могуће је повезати DuckDB са неком другом постојећом базом података над чијим ће се подацима извршавати упити, као што су Postgres или SQLite. Поред обраде DuckDB омогућава и складиштење података у већ поменутим врстама датотека као и у базама података (слика 34). [18]



Слика 34. Примена DuckDB-а [18]

3.1. Разлози за коришћење DuckDB базе података

1. Брзина, ефикасност и смањење трошкова

DuckDB је посебно ефикасан када се ради о локалној обради података. DuckDB омогућава локалну обраду података чиме се губи потреба за online cloud сервисима за аналитичку обраду података. Овом карактеристиком DuckDB-а се јако убрзава обрада података и смањују се трошкови јер се не користе cloud сервиси попут AWS Athena или BigQuery-а. Уштеда се највише примећује током обраде велике количине података. [18]

2. Једноставна инсталација

DuckDB се једноставно инсталира у различитим окружењима. Поред једноставности инсталације, DuckDB не захтева велику количину ресурса. Има подршку за велики број програмских језика као што су: C, C++, Java, Node.js, Python… Такође је доступан и за оперативне системе Windows, Linux, macOS. [18]

3. Напредне SQL функционалности

Самим тим што је DuckDB релациона база података, поседује SQL упитни језик, при чему има и неке додатне проширене функције у односу на стандардни SQL који је познат већини програмера. Проширени SQL омогућава да се сложени упити обављају брже и са мање кода. [18]

4. Анализа приватних података

Самим тим што је DuckDB ефикасан за локалну обраду, значи да се њиме могу ефикасно обрађивати локални приватни подаци са већом сигурношћу јер се не обрађују online. Ово је веома битна предност у системима у којима се обрађују осетљиви и тајни подаци. [18]

5. Дистрибуирана обрада података

Иако је DuckDB првенствено намењен локалној аналитици, сваког дана се DuckDB развија и као алат за дистрибуирану обраду података. Пројекат MotherDuck омогућава коришћење DuckDB-а и у cloud окружењима, што даје додатну флексибилност када је потребно анализирати податке са више различитих извора. [18]

3.2. Стандардне примене DuckDB-а

1. Edge computing

DuckDB је ефикасан за анализу података директно на месту где се они прикупљају, као што су паметни бројачи или IoT сензори. Ова локална анализа омогућава брже добијање резултата и смањује потребу за слањем великих количина података у cloud, што помаже у уштеди трошкова преноса и обраде. [19]

2. Анализа великих скупова података

DuckDB је веома ефикасан за анализу великих скупова података, као што су log датотеке, без потребе за њиховим премештањем у cloud. DuckDB омогућава да се подаци анализирају директно тамо где се налазе, што може значајно смањити трошкове и време обраде. [18]

3. Data science и машинско учење

DuckDB може олакшавати рад и анализу јер омогућава бржу припрему, филтрирање и агрегацију података. Због своје брзине обраде великих количина података погодан је за системе који се базирају на машинском учењу, као што су системи за препоруку садржаја. [18]

4. Дистрибуирана анализа података

DuckDB се може користити за анализу података који су расподељени између cloud-а, edge мреже и локалних уређаја. Пројекат MotherDuck омогућава коришћење DuckDB-а у оба окружења тачније у cloudu и локално што омогућава ефикасну обраду података на различитим нивоима. [18]

5. Трансформација и чување података

DuckDB може бити коришћен за трансформацију и припрему података пре него што се обави анализа. Омогућава ефикасну промену назива колона, типова података и вредности, као и рад са угњежденим документима, претварајући их у релационе структуре које су лакше за анализу. [18]

3.3. Примена DuckDB-а у анализи велике количине података

Велика примена DuckDB базе података је у системима у којима је потребна обрада и трансформација великих количина података. Његова колумнарна архитектура омогућава ефикасну обраду података у датотекама које садрже табеле са великим бројем колона и редова. У зависности од врсте система подаци које је потребно обрадити могу достизати величину од неколико десетина па и стотина гигабајта. Велики допринос ефикасности DuckDB базе података обезбеђује могућност обраде података локално, без потребе за коришћењем cloud сервиса, као што су Amazon Redshift или Google BigQuery [18]. DuckDB је обогаћен богатом библиотеком SQL функција као и високо перформансним window функцијама чиме се лакоћа и брзина обраде још више повећава [27]. Такође DuckDB је изузетно користан за трансформацију података пре саме анализе. Уз помоћ напредних SQL функција, могуће је извршити различите врсте трансформација као што су промене назива колона, промена типова података, и уређивање структура података. [18]

Предности DuckDB базе података у обради велике количине података могу бити искоришћене у великом броју различитих система. На пример, системи за давање препорука обрађују податке о милионима интеракција корисника са различитим производима, често у реалном времену. DuckDB омогућава да се ти подаци обрађују брзо и ефикасно, било да се ради о груписању корисничких интеракција, филтрирању производа на основу специфичних атрибута, или рачунању сличности које су кључне за креирање препорука.

Сама ефикасност DuckDB базе података се најлакше приказује кроз пример. Пример је базиран на представљању разлике у времену учитавања података традиционалним техникама као што је коришћење библиотеке pandas и коришћењем DuckDB базе података. На слици 35 приказан је код и потребно време за учитавање датотеке коришћењем библиотеке pandas, док је на слици 36 приказан код и потребно време за учитавање исте датотеке коришћењем DuckDB базе података. Важно је напоменути да је величина тест датотеке 1.8GB, разлика у брзини зависи од величине датотеке и што је величина датотеке већа то је и предност DuckDB базе података већа.

```
import pandas as pd
import timeit

start = timeit.default_timer()
print("The start time is :", start)

df = pd.read_parquet("D:\\large_file\\raw.parquet")

print("The difference of time is :",timeit.default_timer() - start)


The start time is : 10514.3752879
The difference of time is : 4.882038199999442
```

Слика 35. Учитавање података коришћењем pandas библиотеке [28]

```
import duckdb

conn = duckdb.connect()

start = timeit.default_timer()
print("The start time is :", start)

conn.execute("""
    DROP VIEW IF EXISTS binance_btc;
""")
conn.execute("""
    CREATE VIEW binance_btc AS
    SELECT * FROM 'D:\\large_file\\raw.parquet'
""")

print("The difference of time is :",timeit.default_timer() - start)
```

```
The start time is : 10801.4392424
The difference of time is : 0.0012158000008639647
```

Слика 36. Учитавање података коришћењем DuckDB базе података [28]

3.4. Примена DuckDB-а у data science-у и машинском учењу

DuckDB има велику примену у data science-у и машинском учењу због своје способности да брзо врши припрему, анализу, филтрирање и агрегацију података. Подаци се могу анализирати и припремити без потребе за посебним ресурсима или процесима, чиме се смањује сложеност подешавања система и омогућава ефикасна припрема података за машинско учење. У процесу тренирања модела, DuckDB се може користити за чишћење и усклађивање података, промену типова колона, вредности или агрегацију табела. [18]

DuckDB такође омогућава рад са угњежденим подацима, као што су JSON или Parquet датотеке, претварајући их у релационе структуре које су погодне за даљу анализу и тренирање модела. Трансформација података је кључна када су у питању системи за машинско учење јер обезбеђује да су подаци припремљени на најприкладнији начин пре тренинга модела, што може значајно побољшати тачност и брзину алгоритама машинског учења. DuckDB-ова способност брзе обраде великих количина података чини га идеалним за ове процесе, не користећи сложену инфраструктуру и додатне ресурсе [18]. На слици 37 приказан је пример који представља припрему података за тренинг модела. Приказани код чисти, филтрира и агрегира податке како би били спремни за даљу анализу.

```
import duckdb
import pyarrow as pa
import pyarrow.dataset as ds
nyc = ds.dataset('nyc-taxi/', partitioning=["year", "month"])
nyc = duckdb.arrow(nyc)
nyc.filter("year > 2014 & passenger_count > 0 & trip_distance > 0.25 & fare_amount > 0")
    .aggregate("SELECT avg(fare_amount), avg(tip_amount), avg(tip_amount / fare_amount) AS tip_pct", "passenger_count").arrow()
```

Слика 37. Припрема података [29]

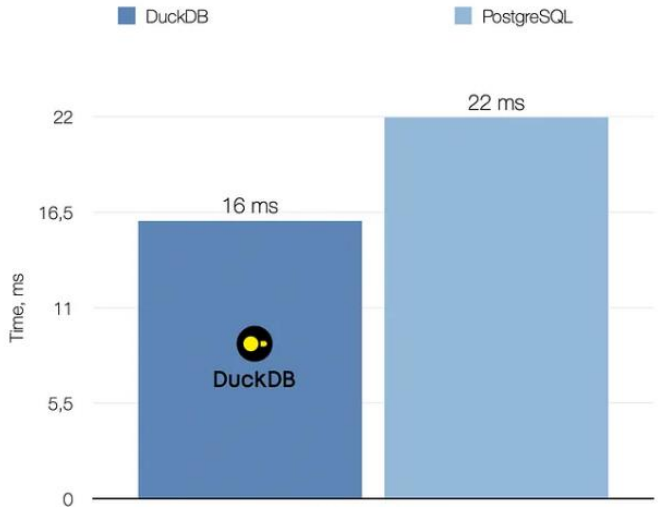
3.5. Разлози за коришћење DuckDB базе за аналитику

DuckDB је оптимизована аналитичка база података (OLAP) и као таква показује добре перформансе приликом читања и обраде велике количине података, нарочито када су у употреби комплексни упити. За разлику од DuckDB-a, PostgreSQL је оптимизована за трансакционе упите (OLTP), међутим, аналитика и обрада велике количине података могу представљати потенцијални проблем. Разлику представља другачија архитектура DuckDb-a јер користи колумнарно складиштење података [25]. Као пример разлика у брзини приликом обраде велике количине података, резултати тестова приказани су на сликама 38, 39 и 40.

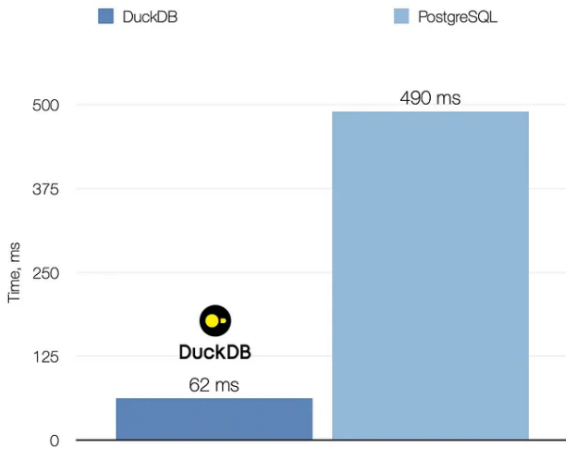
```sql
-- 2,4 mb
SELECT release_year, rating, count(*)
FROM netflix
GROUP BY CUBE (release_year, rating)
```

```sql
-- 415 mb
SELECT search_country, search_position, count(*)
FROM linkedin_job
GROUP BY CUBE (search_country, search_position);
```

Слика 38. Пример упита [24]



Слика 39. Пример резултата над подацима величине 2.4 MB [24]



Слика 40. Пример резултата над подацима величине 450MB [24]

4. Имплементација система за давање препорука коришћењем DuckDB базе података

Ово поглавље објашњава имплементацију концепата наведених у теоријском делу. Биће описана имплементација система за препоруке коришћењем DuckDB-а у e-commerce апликацији. Целокупна идеја апликације је да се омогући кориснику да у online продавници добије препоруке на основу његових укуса, преференција и понашања. Систем ће комбиновати објашњене концепте и улогованом кориснику налазити сличне кориснике и производе, користећи те податке за генерисање одговарајућих препорука. Систем за препоруке биће посебан јер су сви познати алгоритми и концепти прилагођени SQL упитима DuckDB-а. Такође, већина доступне литературе у вези са алгоритмима система за препоруке имплементирана је у програмском језику Python, док је у овом пројекту коришћен програмски језик Node.js. Комплетан код имплементираног система доступан је у јавном репозиторијуму на адреси https://github.com/TheShone/RecommendationSystem.

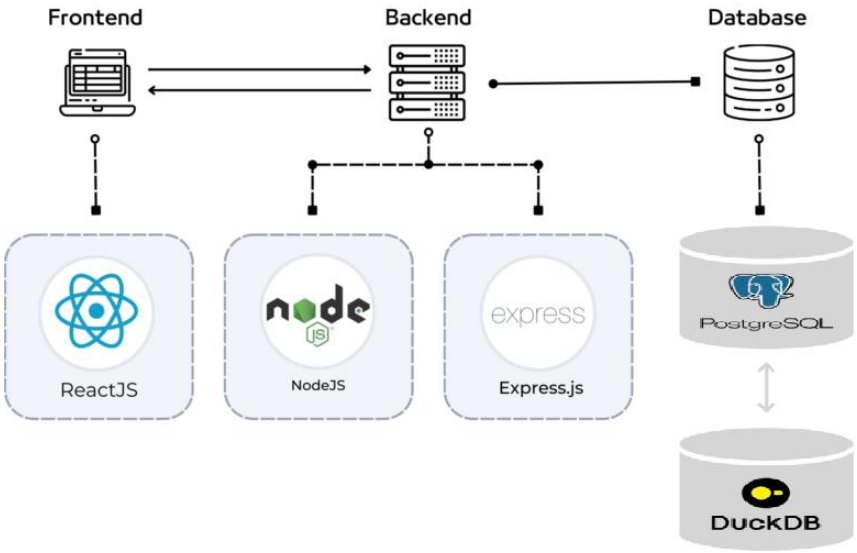
4.1. Архитектура пројекта

Архитектура пројекта базира се на клијент-сервер моделу, где комуникација између клијента и сервера се обавља преко REST API-ја. Слојеви система комуницирају међусобно преко јасно дефинисаних интерфејса, чиме се омогућава скалабилност. Архитектура система приказана је на слици 41 а главне компоненте које се могу идентификовати су:

1. Клијент (Frontend) – Представља део апликације који треба да омогући пријатну, лагодну и лаку употребу апликације корисницима. Омогућава корисницима

пријављивање на систем, систематичан преглед, филтрирање, куповину и оцењивање производа као и приказ препоручених производа пријављеним корисницима. Технологија којом је имплементирана клијентска страна апликације је већ поменути React.js.

2. Сервер (Backend) – Представља део апликације који прима захтеве клијента и позива одговарајуће функције за манипулацију подацима у бази података, другим речима, представља посредника између клијента и базе података. Поред класичне манипулације подацима у бази, веома битна функционалност серверског дела је анализа података из базе и генерисање препорука корисницима. Технологија коришћена за имплементацију сервера је већ поменута комбинација Node.js и Express.js-а.
3. База података – У пројекту користе се две базе података са различитим наменама:
    - PostgreSQL – користи се као примарна база података за чување података као што су подаци о корисницима, производима, брендовима, куповинама, оценама итд.
    - DuckDB – користи се као секундарна база података која се повезује на PostgreSQL и врши упите над подацима. Комплетна математика и израчунавања везана за генерисање препорука обављају се помоћу DuckDB-а.



Слика 41. Архитектура система [22][23]

4.2. Коришћене технологије

Пројекат је рађен у PERN stecku-у, односно у скупу компатибилних технологија: Postgres, Express, React.js и Node.js. Поред класичног PERN приступа коришћена је и DuckDB аналитичка база података.

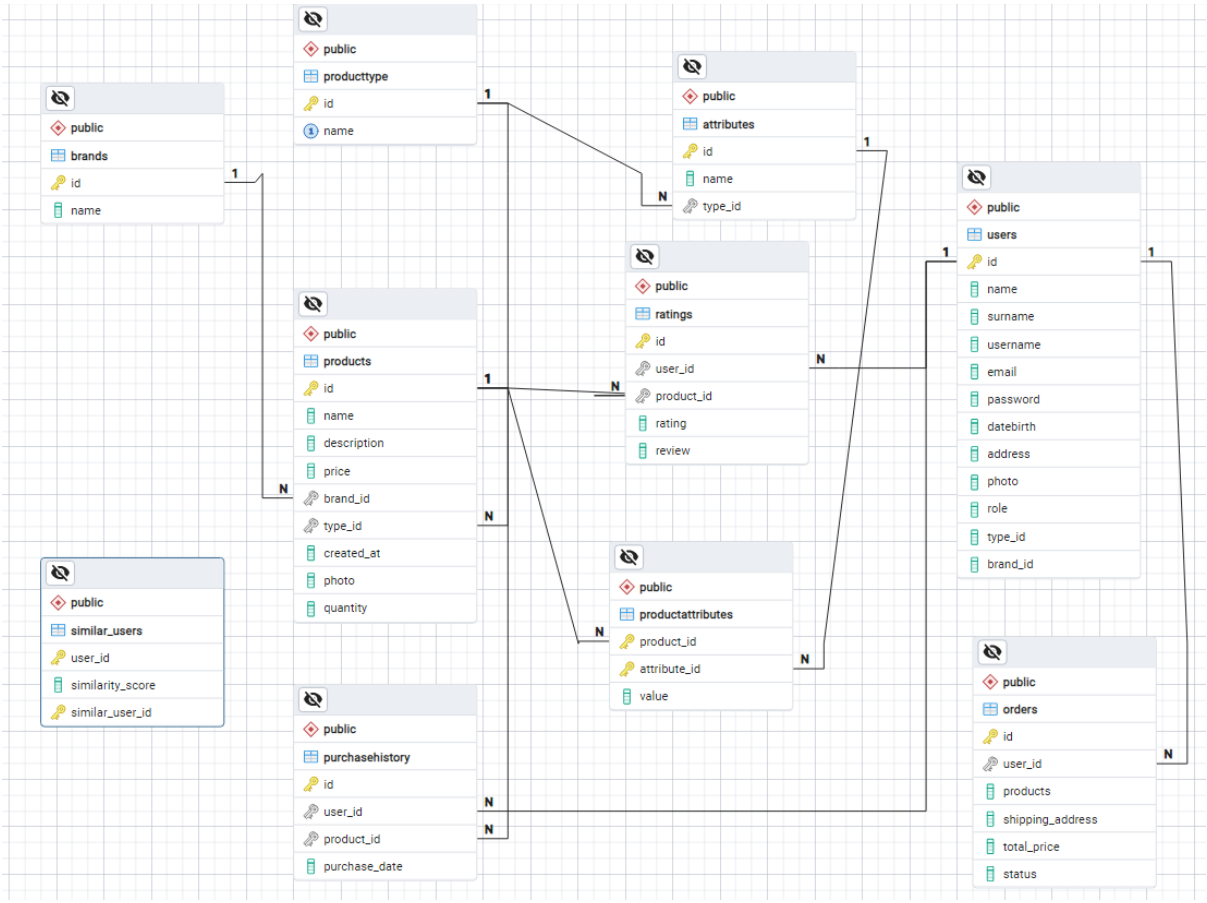
- DuckDB – у овом пројекту има централну улогу. Комплетно израчунавање односно имплементација поменутих алгоритама се обавља коришћењем упита

реализованих DuckDB-јевим проширеним SQL-ом.

- PostgreSQL – коришћен као база за складиштење података. Поред ACID својстава која су неопходна за e-commerce апликацију, главна предност је што DuckDB има екстензију за PostgreSQL па је могуће директно се повезати на податке из базе и вршити израчунавања над њима.
- Node.js – коришћен за имплементацију серверског дела апликације. Један од најактуелнијих програмских језика у свету серверских технологија, врло лако се комбинује са клијентским технологијама које су такође базиране на JavaScript-у. Међу осталим програмским језицима и Node.js има подршку за DuckDB па је самим тим био погодно решење за интеграцију са њим.
- Express.js – коришћен у комбинацији са Node.js-ом за креирање REST API апликације. Поједностављује креирање web апликације тиме што омогућава рутирање и креирање middleware-а.
- React.js – коришћен за имплементацију клијентског дела апликације. Модеран концепт у креирању динамичких web страница, веома заступљен и коришћен. Технологија је заснована на JavaScript-у па се ефикасно комбинује са Node.js-ом и уз помоћ јавно доступне литературе и бројних проширења и библиотека омогућава креирање адекватног корисничког интерфејса.

4.3. Модел

Модел у e-commerce апликацији се користи да би дефинисао и објаснио све ентитете, као и њихове атрибуте и релације. Модел је задужен за представљање типова података који ће бити коришћени у апликацији. Квалитетан модел је неопходан јер представља темељ целокупног пројекта, доприноси томе да се на најлакши начин и са што мање упита манипулише и анализира велика количина података. Приликом креирања модела употребљен је database-first приступ односно у PostgreSQL бази података креирана је база података која одговара моделу. На слици 42 је приказана шема релационе базе података која обухвата све табеле у бази података, као и њихове атрибуте и релације. Табеле су међусобно повезане преко примарних и страних кључева, чиме се омогућава правилно складиштење података. На основу ове шеме креиране су класе модела у коду апликације, где свака табела одговара одређеној класи у коду.



Слика 42. Шема релационе базе података

Класа корисник (слика 43) поред основних информација садржи и референце ка почетним преференцијама које se наводе током регистрације. Приликом регистрације корисник уноси фаворизовани бренд и тип производа како би се решио проблем хладног старта за нове кориснике.

```
class User {
  constructor(id,name,surname,username,email,password,dateBirth,address,photo,role,type_id,brand_id
  ) {
    this.id = id;
    this.name = name;
    this.surname = surname;
    this.username = username;
    this.email = email;
    this.password = password;
    this.dateBirth = dateBirth;
    this.address = address;
    this.photo = photo;
    this.role = role;
    this.type_id = type_id;
    this.brand_id = brand_id;
  }
}
```

Слика 43. Класа корисник

Класа производ (слика 44) поред основних информација садржи и референце бренда и типа производа којим припада.

```
class Product {
  constructor(id,name,description,price,brand_id,type_id,created_at,photo,quantity)
  {
    this.id = id;
    this.name = name;
    this.description = description;
    this.price = price;
    this.brand_id = brand_id;
    this.type_id = type_id;
    this.created_at = created_at;
    this.photo = photo;
    this.quantity = quantity;
  }
}
```

Слика 44. Класа производ

Класа производ_атрибут (слика 45) садржи референце производа, атрибута и вредност атрибута за дати производ.

```
class ProductAttribute {
  constructor(product_id, attribute_id, value) {
    this.product_id = product_id;
    this.attribute_id = attribute_id;
    this.value = value;
  }
}
```

Слика 45. Класа производ_атрибут

Класа оцена (слика 46) садржи референце корисника и проивода, оцену и коментар.

```
class Rating {
  constructor(id, user_id, product_id, rating, review) {
    this.id = id;
    this.user_id = user_id;
    this.product_id = product_id;
    this.rating = rating;
    this.review = review;
  }
}
```

Слика 46. Класа оцена

Класа историја_куповине (слика 47) садржи референце корисника, производа и датум креирања куповине.

```
class PurchaseHistory {
  constructor(id, user_id, product_id, purchase_date) {
    this.id = id;
    this.user_id = user_id;
    this.product_id = product_id;
    this.purchase_date = purchase_date;
  }
}
```

Слика 47. Класа историја_куповине

Детаљи ових класа су неопходни за разумевање алгоритама за генерисање персонализованих препорука корисницима. Поменути алгоритми се заснивају на вредностима атрибута објашњених класа.

4.4. Регистрација и решавање проблема хладног старта за новог корисника

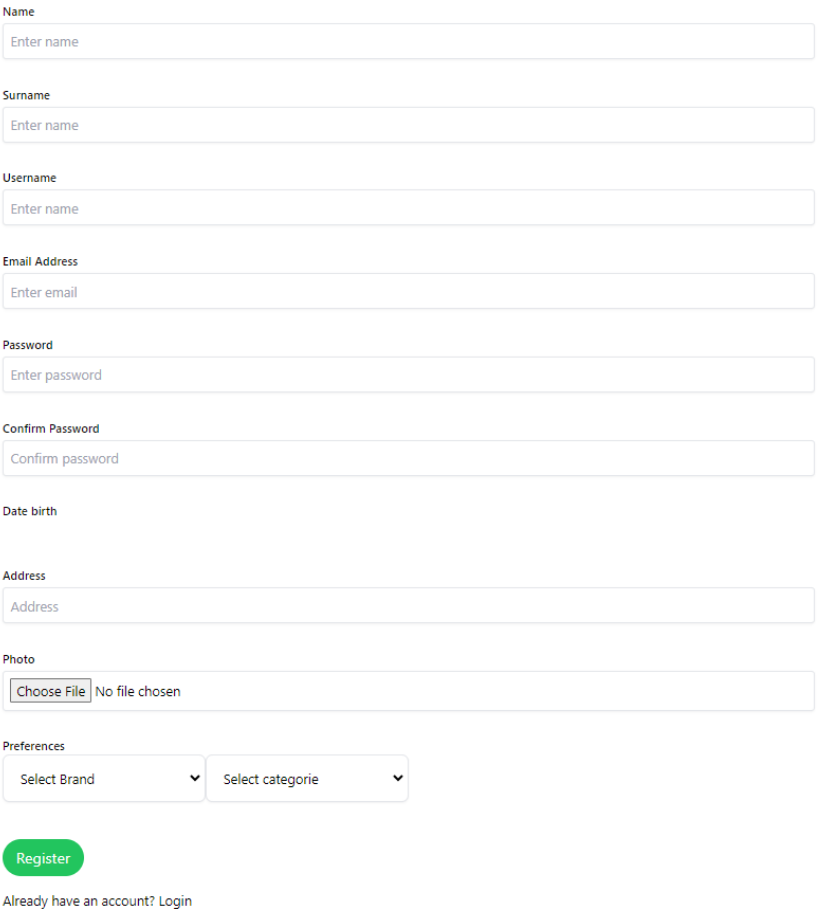
Регистрација је процес у коме корисник уноси своје податке у систем и креира кориснички налог чиме се омогућава да обавља све потребне операције, оцењивања, наручивања, прегледа и филтрирања производа. Као што је већ објашњено велики проблем системима за давање препорука представља проблем хладног старта, прецизније на хладни старт код нових корисника, где систем није у могућности да даје правилне препоруке због изостанка података на основу којих је могуће генерисање препорука. Због тога је у систему поред имплементирања класичног регистровања на систем омогућено корисницима да унесу и почетне преференције у виду омиљеног бренда или типа производа.

Серверска функција регистрације у којој се уносе кориснички лични подаци као и иницијалне преференције приказана је на слици 48, у којој се види да на основу прослеђених података најпре врши хешовање прослеђене шифре коришћењем библиотеке bcrypt како би се заштитио кориснички налог од неовлашћеног приступа бази и могућег читања шифре корисника. Коришћењем креиране конекције са базом и прослеђених података се позива insert наредба у sql упиту којом се у PostgreSQL базу уписују подаци о кориснику.

```
async function createUser(name,surname,username,email,password,dateBirth,address,photo,role,type_id,brand_id
) {
  return new Promise(async (resolve, reject) => {
    const hashedPassword = await bcrypt.hash(password, 10);
    pool.query(
      `INSERT INTO users (name, surname, username, email, password, datebirth, address, photo, role, type_id,brand_id)
      VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10,$11) RETURNING *`,
      [
        name,
        surname,
        username,
        email,
        hashedPassword,
        dateBirth,
        address,
        photo,
        role,
        type_id,
        brand_id,
      ],
      (err, resault) => {
        if (err) return reject(err);
        return resolve(resault.rows[0]);
      }
    );
  });
}
```

Слика 48. Серверска регистрација корисника

На клијентском делу пројекта регистрација је омогућена посебном компонентом за регистрацију која садржи форму у којој корисник уноси своје податке и уноси иницијалне преференције. Форма је приказана на слици 49.



Слика 49. Форма за регистрацију

4.5. Систем за давање препорука

Централни део пројекта представља систем за давање препорука. Као што је већ речено идеја система за давање препорука је да се пријављеном кориснику генеришу препоруке на основу његових преференција. С тим у виду целокупни систем се заснива око главне функције којој се као параметар прослеђује јединствени идентификатор корисника и на основу тог идентификатора систем генерише препоруке. Функција је приказана на слици 50.

```
async function getRecommendations(userId) {
    // Creating connection
    const hasUserPurchaseHistory = await checkPurchaseHistoryByUser(userId, conn);

    if (hasUserPurchaseHistory) {
      const recommendedProducts = await recommendBasedOnHistory(userId, conn);
      if (recommendedProducts && recommendedProducts.length > 0) {
        return recommendedProducts;
      } else return await recommendForNewUser(userId, conn);
    } else {
      return await recommendForNewUser(userId, conn);
    }
  }
```

Слика 50. Функција која генерише препоруке

Из већ објашњених разлога користи се DuckDB базу података као аналитичка база којом се генеришу препоруке, а PostgreSQL база као база која се користи за складиштење података. С тим у вези, пре било каквог израчунавања, прво се у функцији креира инстанца DuckDB базе података и успоставља се конекција. Након тога се креира екстензија DuckDB базе тако да може да користи податке PostgreSQL базе података путем install и load наредби. На основу конекционог стринга, који одређује постојећу PostgreSQL базу, и наредбе attach, креира се конекција DuckDB и PostgreSQL базе.

Након што се успостави конекција, проверава се да ли тренутни корисник има историју куповина или представља новог корисника. Уколико корисник није направио ниједну куповину, позива се функција recommendForNewUser, која генерише препоруке за новог корисника. Ако је корисник већ имао историју куповине, позива се функција recommendBasedOnHistory, која генерише препоруке за кориснике са историјом. Уколико у систему не постоји ниједан сличан корисник или ниједан сличан производ онима које је корисник купио, тада се и за корисника са историјом позива функција за генерисање препорука новим корисницима.

Функција која проверава историју куповине корисника приказана је на слици 51. Функција на основу прослеђеног идентификатора корисника и конекције позива упит који враћа број производа које је корисник купио.

```
async function checkPurchaseHistoryByUser(userId, conn) {
  return new Promise((resolve, reject) => {
    conn.all(
      `
    SELECT COUNT(*) as count
    FROM db.public.purchaseHistory
    WHERE user_id = ${userId};
`,
      (err, result) => {
        if (err) {
          return reject("");
        }
        resolve(Number(result[0].count));
      }
    );
  });
}
```

Слика 51. Провера историје куповине корисника

Како би се решио проблем хладног старта за нове кориснике, раздвојене су функције које генеришу препоруке за кориснике који су имали историју куповине и за кориснике који нису имали историју куповине. Већ је објашњено да нови корисници приликом регистрације уносе почетне преференције у виду фаворизованог бренда и типа производа. Функција за генерисање препорука новим корисницима генерише информације баш на основу тих унешених преференција приликом регистрације. На самом почетку функције покреће се упит који прибавља те преференције на основу прослеђеног идентификатора корисника. Затим се креира упит који проналази у бази

производе које задовољавају преференције корисника које су издвојене предходним упитом. Поред општих информација о проиводима, прибављају се и подаци о средњој вредности оцена које има тај производ коришћењем наредбе avg и коришћењем наредбе coalesce се обезбеђује уколико резултат буде null да се врати 0. Издвојени производи се уређују по опадајућој средњој вредности оцена и по растућој цени и узимају се најбољих десет. Функција за креирање препорука новим корисницима је приказана на слици 52.

```
async function recommendForNewUser(userId, conn) {
  return new Promise((resolve, reject) => {
    conn.all(
      `
        SELECT type_id, brand_id
        FROM db.public.users
        WHERE id = $1;
      `,
      [userId],
      (err, userPreferences) => {
        if (err) {···
        }
        if (!userPreferences || userPreferences.length === 0) {···
        }
        const { type_id, brand_id } = userPreferences[0];
        conn.all(
          `
            SELECT p.id, p.name, p.description, p.price, p.photo,
                   COALESCE(AVG(r.rating), 0) AS average_rating
            FROM db.public.products as p
            LEFT JOIN db.public.ratings as r
            ON p.id = r.product_id
            WHERE p.type_id = ${type_id} OR p.brand_id = ${brand_id}
            GROUP BY p.id, p.name, p.description, p.price, p.photo
            ORDER BY average_rating DESC, p.price ASC
            LIMIT 10;
          `,
          (err, recommendedProducts) => {
            if (err) {···
            }
            resolve(recommendedProducts);
          }
        );
      }
    );
  });
}
```

Слика 52. Функција за генерисање препорука новим корисницима

Пре него што се објасни функција за давање препорука корисницима који имају историју куповина, неопходно је прво објаснити функције којима се проналазе слични производи као и слични корисници. Функција за проналажење сличних производа одговара објашњеном методу филтрирања по садржају односно проналази сличне проиводе на основу сличности њихових садржаја. У овој конкретној функцији сличност је одређена на мало другачији и специфичнији начин. Сличност се базира на специфичности креираног модела с обзиром да сваки производ садржи листу атрибута као и специфичну вредност сваког атрибута за дати производ. Идеја функције је да су два производа слична уколико садрже два иста атрибута са двема истим вредностима, граница је постављена произвољно рачунајући да два производа која имају две исте вредости два атрибута деле доста сличних особина. Улазни атрибути функције су идентификатор производа и конекција за приступ бази података. На основу прослеђеног идентификатора производа и конекције креира се DuckDB упит који проналази слчне производе прослеђеном производу тако што се упоређују подаци из табела

производ_атрибут прослеђеног производа и анализираног производа и уколико постоје два иста атрибута са истим вредностима тај производ се узима као сличан. Функција која израчунава сличне кориснике приказана је на слици 53.

```
async function findSimilarProducts(productId, conn) {
  return new Promise((resolve, reject) => {
    conn.all(
      `
      SELECT pa2.product_id
      FROM db.public.productattributes AS pa1
      JOIN db.public.productattributes AS pa2
        ON pa1.attribute_id = pa2.attribute_id AND pa1.value = pa2.value
      WHERE pa1.product_id = $1 AND pa1.product_id != pa2.product_id
      GROUP BY pa2.product_id
      HAVING COUNT(DISTINCT pa1.attribute_id) >= 2
      `,
      [productId],
      (err, result) => {
        if (err) return reject(err);
        resolve(result.map((row) => row.product_id));
      }
    );
  });
}
```

Слика 53. Функција за налажење сличних производа

Функција за проналажење сличних корисника одговара већ објашњеном концепту колаборативног филтрирања по кориснику са малим унапређењем. Функција за проналажење сличних корисника поред тражења корисника на основу куповине истих производа такође тражи и кориснике који су купили сличне производе. Па самим тим функција се састоји из две подфункције: слични корисници на основу куповине истих производа и слични корисници на основу куповине сличних произода.

Функција за проналажење сличних корисника на основу куповине истих производа се фокусира на сличност коринсика на основу њихових оцена заједничких производа. Сличност се рачуна већ објашњеном метриком косинусне сличности између оцена које су дали истим производима. Повезивањем табела оцена и историја_куповина омогућава се анализа оцена које су корисници дали заједничким купљеним производима. Издвојени корисници се сортирају по сличности у опадајућем редоследу и издваја се најбољих десет корисника. Функција за проналажење сличних корисника на основу куповине истих производа приказана је на слици 54.

```
const similarUsersBySameProductsPromise = new Promise((resolve, reject) => {
  conn.all(
    `
    SELECT ph2.user_id,
   COALESCE(SUM(pr1.rating * pr2.rating) /
    (NULLIF(SQRT(SUM(pr1.rating * pr1.rating)), 0) * NULLIF(SQRT(SUM(pr2.rating * pr2.rating)), 0)), 0) AS similarity_score
  FROM db.public.purchasehistory ph1
  JOIN db.public.ratings pr1
    ON ph1.product_id = pr1.product_id AND pr1.user_id = ph1.user_id
  JOIN db.public.purchasehistory ph2
    ON ph1.product_id = ph2.product_id
  JOIN db.public.ratings pr2
    ON ph2.product_id = pr2.product_id AND pr2.user_id = ph2.user_id
  WHERE ph1.user_id = $1
    AND ph1.user_id != ph2.user_id
  GROUP BY ph2.user_id
  ORDER BY similarity_score DESC
  LIMIT 10;
    `,
    [userId],
    (err, result) => {
      if (err) return reject(err);
      resolve(result.map((row) => row.user_id));
      reject(err);
    }
  );
});
```

Слика 54. Слични корисници на основу куповине истих производа

Функција за проналажење сличних корисника на основу куповине сличних производа се фокусира на сличност коринсика на основу њихових оцена. Логика којом се рачунају слични производи је иста као и у функцији за налажење сличних производа, односно два производа су слична ако имају два иста атрибута и исте вредности одговарајућих атрибута. Исто као и у предходној функцији њихова сличност се рачуна на основу косинусне сличности оцена које су давали сличним производима. Функција за проналажење сличних корисника на основу куповине сличних производа приказана је на слици 55.

```
const similarUsersBySimilarProductsPromise = new Promise(
  (resolve, reject) => {
    conn.all(
      `
    SELECT ph2.user_id,
    COALESCE(SUM(pr1.rating * pr2.rating) /
   (NULLIF(SQRT(SUM(pr1.rating * pr1.rating)), 0) * NULLIF(SQRT(SUM(pr2.rating * pr2.rating)), 0)), 0) AS similarity_score
    FROM db.public.purchasehistory AS ph1
    JOIN db.public.productattributes AS pa1
      ON ph1.product_id = pa1.product_id
    JOIN db.public.productattributes AS pa2
      ON pa1.attribute_id = pa2.attribute_id AND pa1.value = pa2.value
    JOIN db.public.purchasehistory AS ph2
      ON pa2.product_id = ph2.product_id AND ph1.user_id != ph2.user_id
    JOIN db.public.ratings pr1
      ON pr1.product_id = ph1.product_id
    JOIN db.public.ratings pr2
      ON pr2.product_id = pa2.product_id AND pr2.user_id = ph2.user_id
    WHERE ph1.user_id = ${userId}
    GROUP BY ph2.user_id
    HAVING COUNT(DISTINCT pa2.attribute_id) >= 2
    ORDER BY similarity_score DESC;
    `,
      (err, result) => {
        if (err) return reject(err);
        resolve(result.map((row) => row.user_id));
      }
    );
  }
);
```

Слика 55. Слични корисници на основу куповине сличних производа

Главна функција за генерисање сличних корисника комбинује резултате из обе подфункције. Креира се сет од резултата због могућности да се исти корисник нашао у оба резултата, па се спречава дуплирање података. Функција за генерисање сличности корисника је приказана на слици 56.

```
async function findSimilarUsers(userId, conn) {
  try {
    const similarUsersBySameProductsPromise = new Promise((resolve, reject) => {···
    });
    const similarUsersBySimilarProductsPromise = new Promise(
      (resolve, reject) => {···
      }
    );
    const [similarUserIdsByPurchase, similarUserIdsByAttributes] =
      await Promise.all([
        similarUsersBySameProductsPromise,
        similarUsersBySimilarProductsPromise,
      ]);
    const allSimilarUserIds = new Set([
      ...similarUserIdsByPurchase,
      ...similarUserIdsByAttributes,
    ]);

    return Array.from(allSimilarUserIds);
  } catch (error) {
    console.error("Error in finding similar users: ", error);
    throw error;
  }
}
```

Слика 56. Слични корисници

Функција приказана на слици 56 за налажење сличних корисника ради тако што се приликом сваког позива функција која треба да генерише препоруке траже слични корисници. Самим тим што се она увек позива подаци су тачнији и ажурнији јер функција у реалном времену израчунава који корисници имају сличне преференције са пријављеним корисником.

И поред предности да су подаци увек свежи и ажурни постоји једна мана ове функције, а то је да може бити временски захтевна односно да се споро извршава када је база података велика. Из тог разлога креирана је batch скрипта која се аутоматски покреће на сваких 24h и рачуна сличне кориснике и уписује податке у базу података. Логика функције (слика 57) је да за сваког корисника одреди који су му корисници слични на основу куповине истих и куповине сличних производа, комбинује резултате и уписује их у базу података.

```
async function calculateAndStoreSimilarUsers() {
    //Creating connection
    const allUser = await getAllUserIds(conn);
    const allUserIds = allUser.map((user) => user.id);
    for (const userId of allUserIds) {
      const similarUsersByPurchase = await findSimilarUsersByPurchase(
        userId,
        conn
      );
      const similarUsersByAttributes = await findSimilarUsersByAttributes(
        userId,
        conn
      );
      const combinedSimilarUsers = new Set([
        ...similarUsersByPurchase,
        ...similarUsersByAttributes,
      ]);
      await storeSimilarUsersInDatabase(userId, combinedSimilarUsers, conn);
    }

    console.log("Similar users calculation completed.");
    conn.close();
}
```

Слика 57. Функција за рачунање и уписивање сличних корисника

Покретање скрипте омогућено је помоћу node-cron библиотеке (слика 58). Batch скрипта се покреће сваки дан у поноћ израчунава сличне кориснике и уписује их у базу.

```
const cron = require("node-cron");
const { calculateAndStoreSimilarUsers } = require("./batchScriptService");

cron.schedule("0 0 * * *", async () => {
  try {
    console.log("Script starts...");
    await calculateAndStoreSimilarUsers();
    console.log("Script ends.");
  } catch (error) {
    console.error("Greška prilikom izvršavanja skripte:", error);
  }
});
```

Слика 58. Покретање скрипте

Други начин за генерисање сличних корисника је функција приказана на слици 59. И поред мање ажурности података, ова метода је сасвим довољна за генерисање квалитетних препорука. Квалитет препорука за нијансу опада али се брзина добијања препорука повећава чиме се побољшава корисничко искуство.

```
async function findSimilarUsers2(userId, conn) {
  return new Promise((resolve, reject) => {
    conn.all(
      `SELECT similar_user_id
       FROM db.public.similar_users
       WHERE user_id = $1
       ORDER BY similarity_score DESC
       LIMIT 10`,
      [userId],
      (err, result) => {
        if (err) return reject(err);
        resolve(result.map((row) => row.similar_user_id));
      }
    );
  });
}
```

Слика 59. Други начин за генерисање сличних кориниска

Комбинација претходних функција користи се у функцији која генерише препоруке корисницима који поседују историју куповине. Функција је дизајнирана тако да одговара хибридном приступу у систему за давање препорука, на тај начин да се комбинују колаборативно филтрирање и филтрирање по садржају. Улазни параметри функције су идентификатор корисника и конекција ка бази. На самом почетку функције проналазе се слични корисници пријављеном кориснику позивањем функције findSimilarUsers или findSimilarUsers2 у зависности од потребе и количине података. Након прибављања сличних корисника, прибављају се производи које је корисник купио, и то покретањем упита који из табеле историја-куповине прибавља производе на основу корисничког идентификатора. Објашњени део функције приказан је на слици 60.

```
const similarUsers = await findSimilarUsers(userId, conn);
const productMap = new Map();
const userPurchasedProducts = await new Promise((resolve, reject) => {
  conn.all(
    `
    SELECT product_id
    FROM db.public.purchasehistory
    WHERE user_id = $1;
  `,
    [userId],
    (err, result) => {
      if (err) return reject(err);
      resolve(result.map((row) => row.product_id));
    }
  );
});
```

Слика 60. Почетни део функције за генерисање препорука корисницима са историјом

Након одређених сличних корисника и купљених производа функција на основу сличних корисника генерише препоруке. На самом почетку овог дела функције проверава се да ли уопште постоје слични корисници тренутном кориснику и ако постоје функција користи њихову историју да нађе производе које су они купили, а које тренутни коринсик није. Уколико постоје такви производи они се додају у мапу производа productMap. Мапа ће садржати све препоруке корисника и оне генерисане на основу сличних корисника и сличних производа. Објашњени део функције је приказан на слици 61.

```
if (similarUsers.length > 0) {
  const users = similarUsers.join(",");
  const recommendedProducts = await new Promise((resolve, reject) => {
    conn.all(
      `
      SELECT DISTINCT p.id, p.name, p.description, p.price, p.photo,
                      COALESCE(AVG(r.rating), 0) AS average_rating
      FROM db.public.purchasehistory ph
      JOIN db.public.products p ON ph.product_id = p.id
      LEFT JOIN db.public.ratings r ON p.id = r.product_id
      WHERE ph.user_id IN (${users})
        AND p.id NOT IN (${purchases})
      GROUP BY p.id, p.name, p.description, p.price, p.photo
      ORDER BY average_rating DESC, p.price ASC
      LIMIT 10;
    `,
      (err, result) => {
        if (err) return reject(err);
        resolve(result);
      }
    );
  });

  recommendedProducts.forEach((product) => {
    productMap.set(product.id, product);
  });
}
```

Слика 61. Генерисање препорука на основу сличности корисника

Уколико количина генерисаних препорука није већа од десет у том случају се позива и подфункција за генерисање препорука на основу сличности купљених производа и осталих производа. На почетку функције се за сваки купљени производ налазе њему слични производи. Уколико постоје такви производи креира се упит који враћа производе који имају исти идентификатор као пронађени слични производи. Пронађени производи се уређују у опадајући поредак по просечној оцени и у растући по цени. Подфункција приказана је на слици 62.

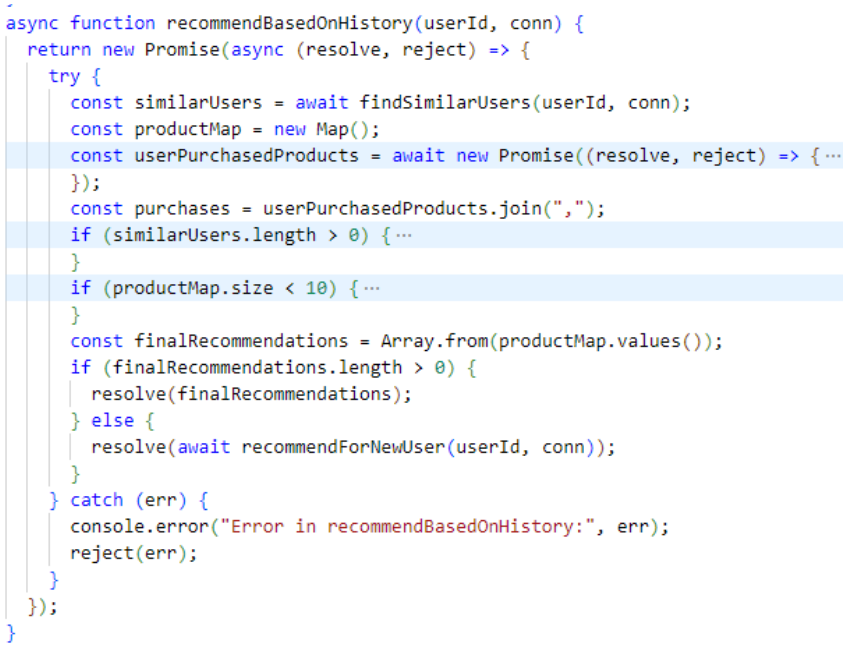
```
if (productMap.size < 10) {
  let similarProducts = [];
  for (let productId of userPurchasedProducts) {
    const simProducts = await findSimilarProducts(productId, conn);
    similarProducts.push(...simProducts);
  }

  if (similarProducts.length > 0) {
    const additionalRecommendedProducts = await new Promise(
      (resolve, reject) => {
        conn.all(
          `
          SELECT DISTINCT p.id, p.name, p.description, p.price, p.photo,
                          COALESCE(AVG(r.rating), 0) AS average_rating
          FROM db.public.products p
          LEFT JOIN db.public.ratings r ON p.id = r.product_id
          WHERE p.id IN (${similarProducts.join(",")})
            AND p.id NOT IN (${userPurchasedProducts.join(",")})
          GROUP BY p.id, p.name, p.description, p.price, p.photo
          ORDER BY average_rating DESC, p.price ASC
          LIMIT ${10 - productMap.size};
        `,
          (err, result) => {
            if (err) return reject(err);
            resolve(result);
          }
        );
      }
    );

    additionalRecommendedProducts.forEach((product) => {
      if (!productMap.has(product.id)) {
        productMap.set(product.id, product);
      }
    });
  }
}
```

Слика 62. Генерисање препорука на основу сличности производа.

Целокупна функција која спаја резултате обе подфункције у финални сет препорука приказана је на слици 63.



```
async function recommendBasedOnHistory(userId, conn) {
  return new Promise(async (resolve, reject) => {
    try {
      const similarUsers = await findSimilarUsers(userId, conn);
      const productMap = new Map();
      const userPurchasedProducts = await new Promise((resolve, reject) => {···
      });
      const purchases = userPurchasedProducts.join(",");
      if (similarUsers.length > 0) {···
      }
      if (productMap.size < 10) {···
      }
      const finalRecommendations = Array.from(productMap.values());
      if (finalRecommendations.length > 0) {
        resolve(finalRecommendations);
      } else {
        resolve(await recommendForNewUser(userId, conn));
      }
    } catch (err) {
      console.error("Error in recommendBasedOnHistory:", err);
      reject(err);
    }
  });
}
```

Слика 63. Генерисање препорука корисницима који имају историју куповине

5. Тестирање функционалности имплементираног система

Ово поглавље приказаће резултате имплементираног система, позивом функције за давање препорука корисницима. Обзиром да се функције разликују у зависности да ли је у питању нови корисник или корисник са историјом куповине, тестирање ће се обавити за две врсте корисника: новог и са историјом. Да би се приказале разлике тестирање се обавља на истом кориснику који нема историју куповине и приказују се препоруке, затим корисник изврши куповину и опет се прикажу препоруке.

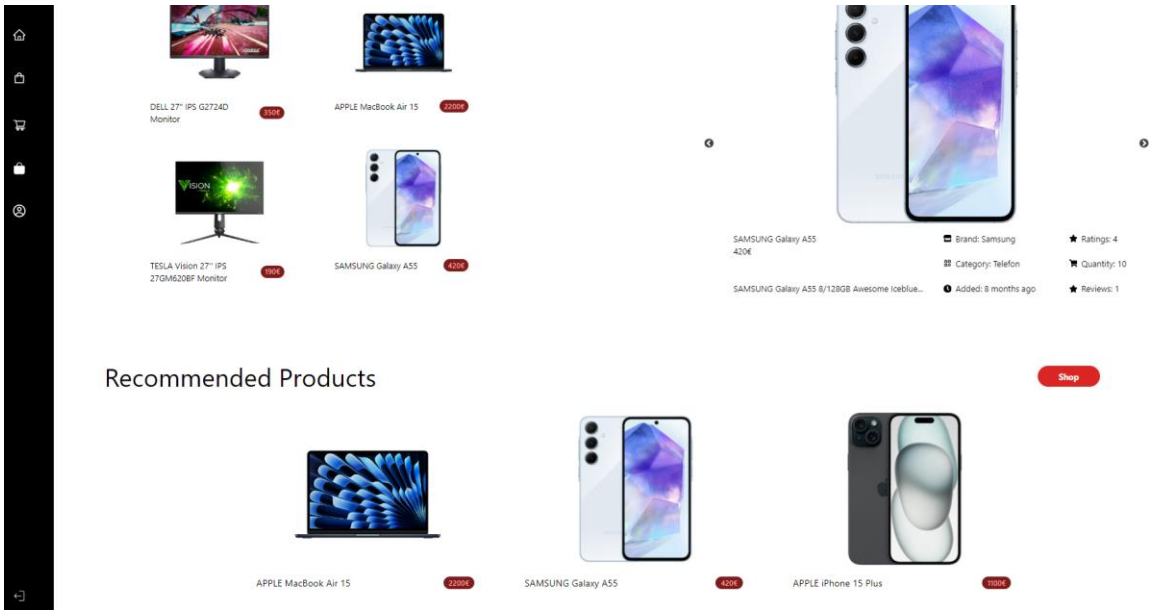
5.1. Нови корисник

Обзиром да је неопходно да корисник приликом регистровања унесе почетне преференције, препоруке ће се базирати на тим преференцијама. Примера ради, корисник као почетне преференције уноси тип производа мобилни телефон и бренд Apple, што је приказано на слици 64. Тест подаци у бази података приказани су на слици 65. Мобилни телефон је тип производа са идентификатором 15, док бренд Apple има идентификатор 6. На основу тест података систем ће препоручити производе са идентификатором 3, 4 и 24 јер одговарају корисничким преференцијама, што је приказано на слици 66.

Слика 64. Почетне преференције

	id [PK] integer	name character varying (255)	description character varying (255)	price integer	brand_id integer	type_id integer
1	1	TOSHIBA DYNABOOK Satellite Pro C50-G-...	TOSHIBA DYNABOOK Satellite Pro C50-G-110 (Tamno plava) Intel i3-10110U, 8GB, 256GB SSD, Win 11 Home (A1PZS23E11...	250	18	3
2	2	TOSHIBA DYNABOOK Satellite Pro C40-G-...	Toshiba Dynabook Satellite Pro C40-G-109 14" HD Intel Celeron 5205U 1.9GHz, 8GB Rama, 128 GB SSD, Windows 10 Edu	260	18	3
3	3	APPLE MacBook Air 15	APPLE MacBook Air 15.3 M3 8/512GB Space Grey MRYN3ZE/A	2200	6	3
4	4	SAMSUNG Galaxy A55	SAMSUNG Galaxy A55 8/128GB Awesome Iceblue	420	2	15
5	5	LG UltraFine Ergo 27"	LG UltraFine Ergo 27" IPS 27UN880P-B Monitor	304	1	5
6	10	AMD Ryzen 9 5900X 3.7GHz	Ryzen procesor serije 9, generacije 5	420	19	6
7	11	INTEL Core i3-10105F 3.7 GHz (4.40 GHz)	Intel procesor i3, 10te generacije. Odlican za kancelarjsku upotrebu.	70	4	6
8	12	INTEL Core i7-12700 2.10 GHz	Odličan procesor	350	4	6
9	13	AMD Ryzen 3 4100 3.8GHz (4.0GHz)	Procesor za kućnu upotrebu	70	19	6
10	14	INTEL Core i5-14600KF 3.50GHz (5.30GHz)	INTEL Core i5-14600KF 3.50GHz (5.30GHz) Procesor	370	4	6
11	22	DELL 27" IPS G2724D Monitor	Odličan monitor za kućnu upotrebu, sa odzivom ekrana od 165 hz pogodan i za gejming	350	3	5
12	23	TESLA Vision 27" IPS 27GM620BF Monitor	Monitor namenjen kućnoj upotrebi, sa velikim ekranom i odličnim karakteristikama	190	17	5
13	24	APPLE iPhone 15 Plus	APPLE iPhone 15 Plus 6/128GB Black MU0Y3SX/A	1100	6	15

Слика 65. База података



Слика 66. Приказ резултата 1

5.2. Корисник са историјом

Да би корисник имао историју неопходно је да обави куповину неког производа. Корисник изврши куповину производа са идентификатором 23. На слици 67 приказана је историја куповина на којој се види да је и корисник са идентификатором 19 такође купио исти производ, самим тим он треба да се сматра сличним корисником. На слици 68 приказан је исечак табеле производ_атрибут која приказује који производ садржи који атрибут, на слици може се уочити да производ са идентификатором 22 има два атрибута исте вредности као и производ 23, па самим тим то значи да су то слични производи, а пошто се на слици виде да је корисник са идентификатором 56 пазарио тај производ значи да је и он сличан кориснику. На основу историје куповине види се да су слични корисници пазарили пет производа које пријављени корисник није, знајући да уколико је број таквих производа мањи од десет траже се и слични производи производима које је корисник пазарио. На основу слике 68 види се да је призвод са идентификатором 5 сличан производу које је корисник пазарио. Самим тим производи који требају бити препоручени кориснику су призводи са идентификаторима: 1, 5, 10, 12, 14, 22. Приказ резултата је на слици 69.

id [PK] integer	user_id integer	product_id integer	purchase_date timestamp without time zone
2	20	1	2024-08-18 17:21:48.768225
3	19	1	2024-08-19 00:13:38.897377
4	21	2	2024-08-19 00:38:27.858162
5	1	1	2024-04-01 00:00:00
6	2	2	2024-04-02 00:00:00
7	3	3	2024-04-03 00:00:00
8	20	13	2024-09-04 00:16:32.357
9	20	3	2024-09-04 00:19:08.935
10	20	13	2024-09-04 00:19:08.935
13	20	14	2024-09-04 00:22:27.289
14	20	10	2024-09-04 00:22:27.29
15	19	14	2024-09-04 00:23:56.625
16	19	10	2024-09-04 00:23:56.626
17	56	12	2024-09-04 19:00:48.108
18	56	10	2024-09-04 19:00:48.109
19	56	1	2024-09-04 22:51:29.564
20	56	10	2024-09-04 22:51:29.564
26	19	23	2024-09-08 08:55:47.2
27	56	22	2024-09-08 08:55:56.082
29	59	23	2024-09-20 22:22:28.317

Слика 67. Историја куповине

product_id [PK] integer	attribute_id [PK] integer	value character varying (255)
23	31	27"
23	14	5ms
5	33	3840x2160
5	14	5ms
5	13	60hz
5	6	IPS
5	31	27"
22	13	165hz
22	14	1ms
23	6	IPS
23	13	165hz

Слика 68. Производи и њихови атрибути

Слика 69. Приказ резултата 2

6. Закључак

У данашњем времену, са све већом експанзијом интернета, тешко је замислити било коју врсту успешног пословања које нема присуство на мрежи. Самим тим свака озбиљнија продавница мора своје услуге пружати и online у виду неке e-commerce апликације. И потраживања корисника су се променила, па и корисници све више очекују да свака e-commerce апликација нуди персонализоване препоруке које би им олакшале одлуке при куповини. Самим тим системи за давање препорука су добили на великом значају и масовно се користе и тек ће се користити. Системи се користе како би се повећала конкурентност пословања на тржишту.

Управо је то разлог због чега се овај рад бави имплементирањем система за давање препорука. У раду су објашњене основе система за давање препорука. Рад се дотакао најпознатијих алгоритама и принципа који се користе. Објашњени су и изазови с којима се сусрећу методе у виду проблема хладног старта и недовољне количине података и начини за њихово превазилажење. Приказана је имплементација хибридног система за давање препорука због његових добрих страна и због решавања изазова које би имале појединачне методе. Приказана је оптимизација алгоритама за колаборативно филтрирање као и филтрирање по садржају тако да их је могуће одрадити помоћу sql упита. Имплеметирани систем је омогућио давање прецизних препорука корисницима на основу њихове историје чиме се побољшава корисничко искуство и повећава задовољство купаца.

Поред система за давање препорука акценат је стављен и на употребу DuckDB аналитичке базе података. База се показала као јако ефикасан и врло моћан алат за обраду велике количине података у реалном времену. Рад са DuckDB-ом током израде овог пројекта показао се као веома једноставан и интуитиван. И поред тога што је DuckDB релативно нова база података добро је покривена документацијом. Званична документација DuckDB базе података као и књиге и чланци написани о њој, пружају сасвим довољно информација за лаку употребу DuckDB базе података. Својом посебном архитектуром омогућила је генерисање препорука у веома кратком временском периоду. И поред брзине базе, систем је још оптимизованији коришћењем batch скрипте која је омогућила да се унапред рачуна и чува сличност између корисника.

Унапређење имплементираног система могло би се остварити применом машинског учења чиме би се унапредио процес давања препорука. Применом машинског учења тренирао би се модел на подацима о понашању корисника чиме би систем могао на бољи начин да предвиди које производе да препоручи корисницима. Поред машинског учења још једно унапређење могло би да буде употреба дистрибуиране обраде. Дистрибуирана обрада би могла бити уведена кроз интеграцију са MotherDuck-ом, чиме би се постигла ефикаснија обрада великих скупова података који су складиштени на више локација.

Рад је показао да се коришћењем DuckDB-а може имплементирати јако добар и ефикасан систем за давање препорука. Имплементација приказала је комбиновање колаборативног филтрирања и филтрирања по саржају чиме се повећава прецизност препорука. Систем не само да пружа висок ниво персонализације, већ је и оптимизован за брзе аналитичке упите.

7. Литература

[1] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*. Scholars Portal, 2019.

[2] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: An Introduction Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich*. New York: Cambridge University Press, 2012.

[3] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, Nov. 2015. doi:10.1016/j.eij.2015.06.005

[4] H. Ko, S. Lee, Y. Park, and A. Choi, "A survey of recommendation systems: Recommendation models, techniques, and Application Fields," *Electronics*, vol. 11, no. 1, p. 141, Jan. 2022. doi:10.3390/electronics11010141

[5] "What is a recommendation system?," NVIDIA Data Science Glossary, https://www.nvidia.com/en-eu/glossary/recommendation-system/ (accessed Sep. 9, 2024).

[6] J. B. Schafer, J. Konstan, and J. Riedl, "Recommender systems in e-commerce," *Proceedings of the 1st ACM conference on Electronic commerce*, Nov. 1999. doi:10.1145/336992.337035

[7] Y. Verma, "A guide to building hybrid recommendation systems for beginners," AIM, https://analyticsindiamag.com/developers-corner/a-guide-to-building-hybrid-recommendation-systems-for-beginners/ (accessed Sep. 10, 2024).

[8] D. Roy and M. Dutta, "A systematic review and research perspective on Recommender Systems," *Journal of Big Data*, vol. 9, no. 1, May 2022. doi:10.1186/s40537-022-00592-5

[9] D. Pandit, "Recommender system : User Collaborative Filtering," Medium, https://medium.com/@deepapandithu/recommender-system-user-collaborative-filtering-37613f0c6a9 (accessed Sep. 10, 2024).

[10] J. M. Ph.D. and E. Kavlakoglu, "What is collaborative filtering?," IBM, https://www.ibm.com/topics/collaborative-filtering (accessed Sep. 11, 2024).

[11] J. chiang, "Overview of item-item collaborative filtering recommendation system," Medium, https://medium.com/geekculture/overview-of-item-item-collaborative-filtering-recommendation-system-64ee15b24bb8 (accessed Sep. 11, 2024).

[12] F. Casalegno, "Recommender Systems - A Complete Guide to Machine Learning Models," Medium, https://towardsdatascience.com/recommender-systems-a-complete-guide-to-machine-learning-models-96d3f94ea748 (accessed Sep. 11, 2024).

[13] Turing, “A guide to content-based filtering in Recommender Systems,” A Guide to Content-based Filtering in Recommender Systems, https://www.turing.com/kb/content-based-filtering-in-recommender-systems (accessed Sep. 11, 2024).

[14] U. Iftikhar, “Part 3: Exploring content-based filtering in recommendation systems,” Medium, https://umair-iftikhar.medium.com/part-3-exploring-content-based-filtering-in-recommendation-systems-836e5e2fe152 (accessed Sep. 9, 2024).

[15] U. Desai, “Recommendation systems explained: Understanding the basic to advance,” Medium, https://utsavdesai26.medium.com/recommendation-systems-explained-understanding-the-basic-to-advance-43a5fce77c47 (accessed Sep. 10, 2024).

[16] R. Caballar and C. Stryker, “What is a recommendation engine?,” IBM, https://www.ibm.com/think/topics/recommendation-engine (accessed Sep. 8, 2024).

[17] J. chiang, “7 types of hybrid recommendation system,” Medium, https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8 (accessed Sep. 9, 2024).

[18] M. Needham, M. Hunger, and M. Simons, *DuckDB in Action*. Shelter Island, NY: Manning Publications, 2024.

[19] P. Sharma, “What is duck db?,” RudderStack, https://www.rudderstack.com/blog/what-is-duck-db/ (accessed Sep. 10, 2024).

[20] C. Dennis, “What is DuckDB and why it’s the new tool for a data analyst.,” Hightouch, https://hightouch.com/blog/duckdb (accessed Sep. 11, 2024).

[21] “DuckDB,” Database of Databases, https://dbdb.io/db/duckdb (accessed Sep. 11, 2024).

[22] Artyom, “Tutorial Fullstack: REACT TYPESCRIPT + nodejs express + postgresql,” tutofox.com, https://www.tutofox.com/react/tutorial-fullstack-react-typescript-nodejs-express-postgresql/ (accessed Sep. 10, 2024).

[23] M. Raasveldt, “DuckDB on linkedin: New blog post by Mark Raasveldt: Multi-database support in...: 17 comments,” DuckDB on LinkedIn: New blog post by Mark Raasveldt: Multi-Database Support in... | 17 comments, https://www.linkedin.com/posts/duckdb_new-blog-post-by-mark-raasveldt-multi-database-activity-7156732440961507329-e_YT/ (accessed Sep. 10, 2024).

[24] Vitaliy, “The difference in productivity between DuckDB and PostgreSQL at various data volumes,” Medium, https://medium.com/@marvin_data/the-difference-in-productivity-between-duckdb-and-postgresql-at-various-data-volumes-536bd345a155 (accessed Sep. 21, 2024).

[25] “Compare duckdb vs PostgreSQL,” InfluxData, https://www.influxdata.com/comparison/duckdb-vs-postgres/ (accessed Sep. 21, 2024).

[26] M. Jadeja, “Jaccard similarity made simple: A beginner’s guide to data comparison,” Medium, https://medium.com/@mayurdhvajsinhjadeja/jaccard-similarity-34e2c15fb524 (accessed Sep. 23, 2024).

[27] G. User, “Why duckdb,” DuckDB, https://duckdb.org/why_duckdb.html (accessed Sep. 11, 2024).

[28] D. Raichaudhuri, “Ducktales-part 2: Pandas vs DuckDB-large dataset processing comparison,” Medium, https://diptimanrc.medium.com/ducktales-part-2-pandas-vs-duckdb-large-dataset-processing-comparison-70a605097b1e (accessed Oct. 1, 2024).

[29] P. H. and J. Keane, “DuckDB quacks arrow: A Zero-copy data integration between Apache arrow and duckdb,” DuckDB, https://duckdb.org/2021/12/03/duck-arrow.html (accessed Oct. 1, 2024).