

# Personalizovane preporuke animea Primenom kolaborativnog i content-based filtriranja

# Sadržaj

Kolaborativno filtriranje

Content-based filtriranje

SVD (Singular Value Decomposition)

Hibridni model

Obuka modela

Evaluacija modela

Optimizacija

Analiza rezultata

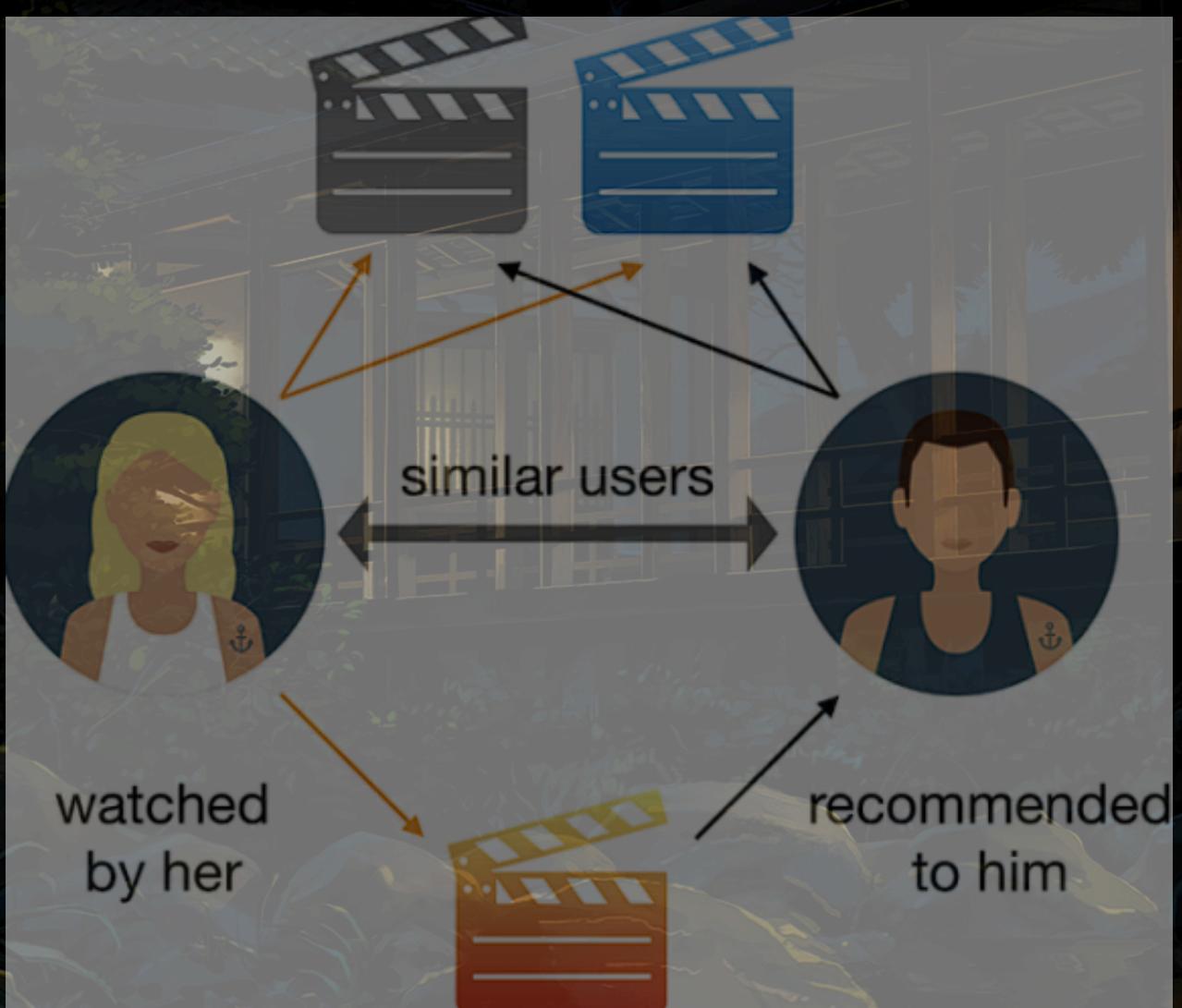
Prvobitni model: Preporuka destinacija u Indiji

Literatura

- 3
- 4-5
- 6-7
- 8
- 9
- 10-11
- 12
- 13
- 14
- 15

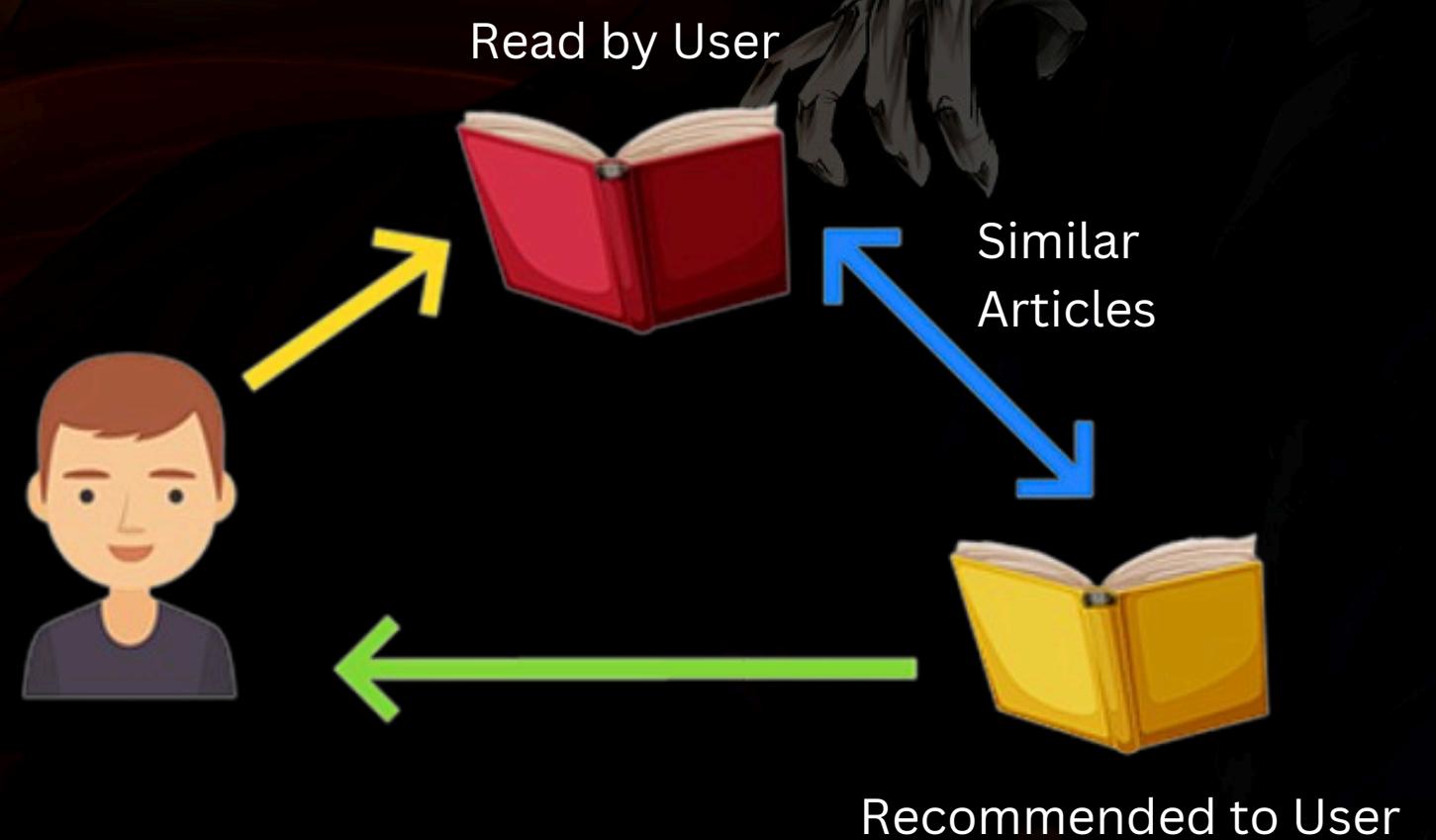
# Kolaborativno filtriranje

- Tehnika preporučivanja koja koristi podatke o korisničkim ocenama i preferencijama da bi predvidela ocene ili rangirala stavke koje korisnik još nije video.
- Temelji se na pretpostavci da će korisnici sličnih interesovanja ocenjivati stavke na sličan način



# Content-based filtriranje

- Tip sistema preporuka koji je u stanju da svakom korisniku pruži veoma personalizovane preporuke za articlē.
- Primer: ako volimo da gledamo Marvelove filmove, onda ćemo verovatno u budućnosti gledati Betmena pre nego "Krive su zvezde". To je osnovna ideja filtriranja zasnovanog na sadržaju.
- Ovaj algoritam daje preporuke za articlē na osnovu onih koje smo voleli u prošlosti.



# Kako to izgleda u našem kodu?

```
# Funkcija za pronalaženje sličnih animea na osnovu žanra
def find_similar_anime(anime_id, anime_ids_watched, n=5):
    if anime_id not in anime_df.index:
        return []

    genres = anime_df.loc[anime_id, 'genre']
    if pd.notnull(genres) and isinstance(genres, str):
        genres_list = genres.split(', ')
        similar_anime = anime_df[anime_df['genre']
            .apply(lambda x: isinstance(x, str) and any(genre in x.split(', ')
                for genre in genres_list))]

    else:
        similar_anime = pd.DataFrame()

# Isključivanje već gledanih animea
similar_anime = similar_anime[~similar_anime.index.isin(anime_ids_watched)].head(n)
return similar_anime.index.tolist()
```

- Ukoliko se korisnik nalazi u rating fajlu, znači da je već pogledao i ocenio neke anime, pa njih korstimo kao kriterijum za predlaganje novog.

- Ključan atribut za pronalaženje sličnih animea jeste žanr. Kriterijum za predlaganje narednog animea jeste poklapajući žanr.

```
# Funkcija za dobijanje sadržajno zasnovanih preporuka
def get_content_based_recommendations(user_id, k=5):
    anime_ids_watched = rating_df[rating_df['user_id'] == user_id]['anime_id'].tolist()
    content_based_recommendations = []
    for anime_id in anime_ids_watched:
        similar_anime_ids = find_similar_anime(anime_id, anime_ids_watched)
        content_based_recommendations.extend(similar_anime_ids)

    anime_names = [anime_df.loc[idx, 'name'] for idx in content_based_recommendations if idx in anime_df.index]
    return anime_names[:k]
```

# SVD (Singular Value Decomposition)

- Matematička tehnika koja se koristi za dekompoziciju matrice u tri manje matrice.
- U kontekstu sistema za preporuku, koristi se za dekompoziciju korisničko-objektne matrice (matrice ocena) kako bi se identifikovale latentne (skrivene) faktore koji objašnjavaju obrasce u ocenama. To omogućava predviđanje ocena koje korisnici nisu dali.
- Dekompozicija: Matrica ocena se dekomponuje u tri matrice:  $U$  (korisničke karakteristike),  $\Sigma$  (dijagonalna matrica sa singularnim vrednostima) i  $V^*$  (karakteristike objekata).
- Redukcija dimenzija: Singularne vrednosti u  $\Sigma$  se koriste za redukciju dimenzija, zadržavajući samo najvažnije faktore.
- Rekonstrukcija i predikcija: Korišćenjem ovih faktora, predikcije se vrše za nedostajuće ocene u originalnoj matrici.

$$M = \begin{matrix} U & \Sigma & V^* \\ m \times n & m \times m & m \times n & n \times n \end{matrix}$$

# Kako to funkcioniše u našem kodu?

```
# Funkcija za dobijanje preporuka za korisnika koristeći SVD model
def get_svd_recommendations(user_id, n=5):
    anime_ids = anime_df['anime_id'].unique()
    user_rated_anime = rating_df[rating_df['user_id'] == user_id]['anime_id'].tolist()
    anime_to_predict = [anime_id for anime_id in anime_ids if anime_id not in user_rated_anime]

    predictions = [svd.predict(user_id, anime_id) for anime_id in anime_to_predict]
    predictions.sort(key=lambda x: x.est, reverse=True)

    recommended_anime = [anime_df.loc[anime_df['anime_id'] == pred.iid, 'name'].values[0] for pred in predictions[:n]]
    return recommended_anime
```

- Koraci:
- 1. Prikupljamo sve jedinstvene vrednosti za id-eve animea.
- 2. Pronalazimo sve ocenjene animee od strane prosleđenog user-a.
- 3. Prikupljamo sve neocenjene animee od strane prosleđenog user-a
- 4.Za svaki anime koji korisnik nije ocenio, funkcija koristi SVD model (svd) da predvidi ocenu koju bi korisnik dao tom animeu.
- 5.Sortiranje predikcija po oceni u opadajućem redosledu.
- 6.Prikupljamo i vraćamo prvih n najbolje prediktovanih animea.

# Hibridni model

## Kombinacija prethodna dva modela

```
# Funkcija za kombinovane preporuke
def get_combined_recommendations(user_id, n=5):
    svd_recommendations = get_svd_recommendations(user_id, n=n)
    content_based_recommendations = get_content_based_recommendations(user_id, k=n)

    combined_recommendations = list(set(svd_recommendations + content_based_recommendations))[:n]
    return combined_recommendations
```

- Ovde koristimo kombinaciju kolaborativnog i content-based filtriranja da bismo videli kakve predloge daje takav hibridni model.

# Obuka modela

```
# Učitavanje podataka
anime_df = pd.read_csv('anime.csv')
rating_df = pd.read_csv('rating.csv')

# Pretprocesiranje podataka
rating_df.drop_duplicates(inplace=True)
rating_df = rating_df[rating_df['rating'] > 0] # Uklonili smo nevalidne ocene

# Učitavanje podataka u Surprise dataset
reader = Reader(rating_scale=(1, 10))
data = Dataset.load_from_df(rating_df[['user_id', 'anime_id', 'rating']], reader)

# Podela podataka na trening i test skupove
trainset, testset = train_test_split(data, test_size=0.2)

# Treniranje SVD modela
svd = SVD()
svd.fit(trainset)
```

3. korak

1. korak nakon učitavanja neophodnih podataka je uklanjanje duplikata i nevalidnih ocena.

2. korak predstavlja prebacivanje podataka iz pandas DataFrame-a u Surprise dataset

4. korak

```
# Evaluacija modela
predictions = svd.test(testset)
print(f'RMSE: {rmse(predictions):.4f}')
print(f'MAE: {mae(predictions):.4f}')
```

# Evaluacija modela

## RMSE (Root Mean Squared Error)

Metrika koja meri prosečnu kvadratnu razliku između stvarnih i predviđenih vrednosti.

RMSE se računa kao kvadratni koren proseka kvadrata razlika između stvarnih ocena  $r_{ui}$  i predviđenih ocena  $\hat{r}_{ui}$ :

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{u,i} (r_{ui} - \hat{r}_{ui})^2}$$

- gde su N broj uzoraka, u indeks korisnika, i indeks stavke (npr. film ili proizvod).
- Manja vrednost RMSE ukazuje na bolje performanse modela, jer manja razlika između stvarnih i predviđenih ocena sugerira da model bolje generalizuje i precizno predviđa.

## MAE (Mean Absolute Error)

Metrika koja meri prosečnu apsolutnu razliku između stvarnih i predviđenih vrednosti.

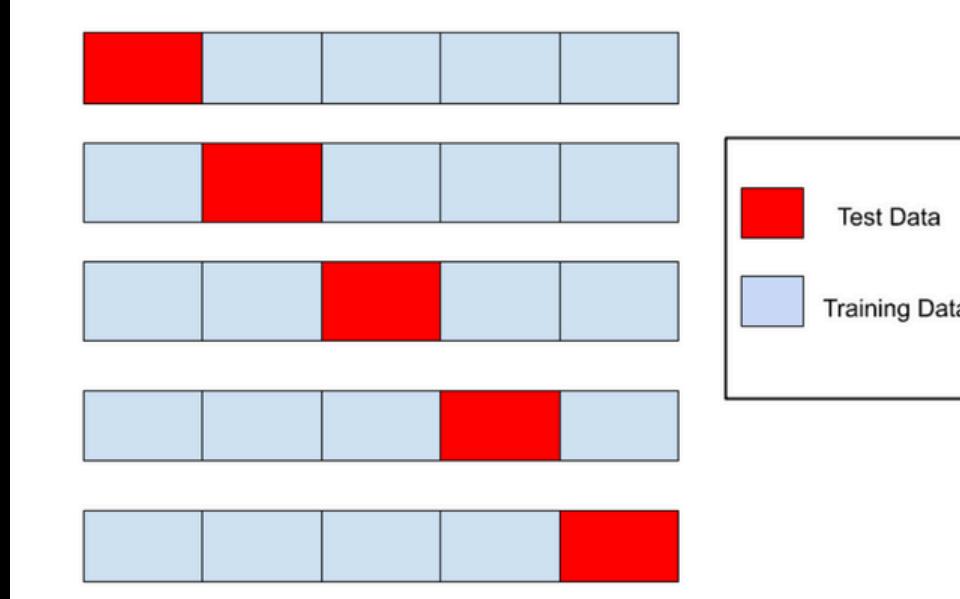
MAE se računa kao prosečna apsolutna razlika između između stvarnih ocena  $r_{ui}$  i predviđenih ocena  $\hat{r}_{ui}$ :

$$\text{MAE} = \frac{1}{N} \sum_{u,i} |r_{ui} - \hat{r}_{ui}|$$

- gde su N, u, i definisani kao u slučaju RMSE.
- Slično kao i RMSE, manja vrednost MAE ukazuje na bolje performanse modela. MAE je intuitivnija jer direktno meri prosečnu grešku u istim jedinicama kao i originalne ocene.

## K-folds validacija

Tehnika koja pomaže u proceni performansi modela tako što podatke deli na K podskupova (foldova). Svaki podskup se koristi kao test skup dok se ostali podskupovi koriste za obuku modela. Postupak se ponavlja K puta, pri čemu se svaki podskup koristi kao test skup tačno jednom.



## Precision

Meri tačnost pozitivnih predviđanja modela. Preciznost odgovara na pitanje: "Od svih predviđenih pozitivnih instanci, koliko je tačno?" Visoka preciznost znači da je veliki deo pozitivnih predviđanja zaista tačan..

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

TP (True Positives): Broj tačno pozitivnih predviđanja (kada je predviđanje modela pozitivno i tačno je).

FP (False Positives): Broj lažno pozitivnih predviđanja (kada je predviđanje modela pozitivno, ali je tačno negativno).

## Recall

Meri koliko stvarno pozitivnih instanci model može da identifikuje. Recall odgovara na pitanje: "Od svih stvarnih pozitivnih instanci, koliko je model uspeo da identifikuje?"

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

FN (False Negatives): Broj lažno negativnih predviđanja (kada je predviđanje modela negativno, ali je tačno pozitivno).

# Optimizacija modela

## Podešavanje hiperparametara

Podešavanje parametara je proces odabira optimalnih vrednosti parametara algoritma kako bi se postigla bolja tačnost ili performanse modela.

Kako se vrši podešavanje parametara u projektu?

- Koristili smo GridSearchCV iz surprise.model\_selection modula.
- Definisali smo mrežu parametara (param\_grid) koja obuhvata kombinacije vrednosti za n\_factors, n\_epochs, lr\_all, i reg\_all.

n\_facotrs predstavlja broj latentnih faktora.

n\_epochs predstavlja broj epoha.

lr\_all predstavlja stopu učenja (learning rate)

reg\_all predstavlja regularizacioni parametar koji pomaže u kontroli prenaučenosti

```
# Podešavanje hiperparametara
param_grid = {
    'n_factors': [50, 100, 150],
    'n_epochs': [20, 30, 40],
    'lr_all': [0.002, 0.005, 0.01],
    'reg_all': [0.02, 0.05, 0.1]
}

gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=3)
gs.fit(data)

# Najbolji parametri
print(f'\nNajbolji parametri: {gs.best_params["rmse"]}')
```

# Analiza rezultata

RMSE pre optimizacije: 1.2230

MAE pre optimizacije: 0.9271

Precision (SVD, pre optimizacije): 0.9418

Recall (SVD, pre optimizacije): 0.5636

Najbolji parametri: {'n\_factors': 100, 'n\_epochs': 30, 'lr\_all': 0.01, 'reg\_all': 0.15}

RMSE posle optimizacije: 1.1965

MAE posle optimizacije: 0.9094

Precision (SVD, posle optimizacije): 0.9438

Recall (SVD, posle optimizacije): 0.5658

Preporuke za korisnika 815 koristeći SVD pre optimizacije:

1. Hunter x Hunter (2011)
2. Steins;Gate
3. Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare
4. Mahou Shoujo Madoka★Magica Movie 3: Hangyaku no Monogatari
5. Rainbow: Nisha Rokubou no Shichinin

Preporuke za korisnika 815 koristeći filtriranje zasnovano na sadržaju pre optimizacije:

1. Kimi no Na wa.
2. Fullmetal Alchemist: Brotherhood
3. Haikyuu!!: Karasuno Koukou VS Shiratorizawa Gakuen Koukou
4. Ginga Eiyuu Densetsu
5. Clannad: After Story

Kombinovane preporuke za korisnika 815 pre optimizacije:

1. Kimi no Na wa.
2. Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare
3. Clannad: After Story
4. Mahou Shoujo Madoka★Magica Movie 3: Hangyaku no Monogatari
5. Rainbow: Nisha Rokubou no Shichinin

Preporuke za korisnika 815 koristeći SVD sa najboljim parametrima:

1. Gintama°
2. Gintama&#039;; Enchousen
3. Ginga Eiyuu Densetsu
4. Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare
5. Kuroko no Basket 2nd Season

```
print(f"Srednja F1-mera: {collab_metrics[2]}")
```

```
print("\nFiltriranje zasnovano na sadržaju:")
print(f"Srednja preciznost: {content_metrics[0]}")
print(f"Srednji odziv: {content_metrics[1]}")
print(f"Srednja F1-mera: {content_metrics[2]})")
```

25m 44.1s

君  
の  
名  
は  
|

Preporuke za korisnika 815 koristeći filtriranje zasnovano na sadržaju sa najboljim parametrima:

1. Kimi no Na wa.
2. Fullmetal Alchemist: Brotherhood
3. Haikyuu!!: Karasuno Koukou VS Shiratorizawa Gakuen Koukou
4. Ginga Eiyuu Densetsu
5. Clannad: After Story

Kombinovane preporuke za korisnika 815 sa najboljim parametrima:

1. Kimi no Na wa.
2. Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare
3. Clannad: After Story
4. Mahou Shoujo Madoka★Magica Movie 3: Hangyaku no Monogatari
5. Rainbow: Nisha Rokubou no Shichinin

- $0 \leq RMSE < \infty$
- $0 \leq MAE \leq \infty$
- $0 \leq Precision \leq 1$
- $0 \leq Recall \leq 1$

- Za RMSE i MSE vrednosti bliže nuli su bolje, dok kod Precision i Recall vrednosti bliže 1 smatrane su boljima.

# Prvobitan model

## Preporuka destinacija u Indiji

User-Destination Matrix:

DestinationID	1	2	3	4	5	6	7	8	9	10	\
---------------	---	---	---	---	---	---	---	---	---	----	---

UserID

1	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

DestinationID	...	991	992	993	994	995	996	997	998	999	1000
---------------	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------

UserID

1	...	0.0	0	0.0	0.0	0	0.0	0.0	0.0	0	0.0
2	...	0.0	0	0.0	0.0	0	0.0	0.0	0.0	0	0.0
3	...	0.0	0	0.0	0.0	0	0.0	0.0	0.0	0	0.0
5	...	0.0	0	0.0	0.0	0	0.0	0.0	0.0	0	0.0
7	...	0.0	0	0.0	0.0	0	0.0	0.0	0.0	0	0.0

[5 rows x 1000 columns]

Korisnik Rohan nema dovoljno istorije poseta ili sličnih korisnika za generisanje preporuka.

Korisnik sa ID-em 202 nije pronađen u matrici sličnosti korisnika.

Korisnik sa ID-em 203 nije pronađen u matrici sličnosti korisnika.

Korisnik Ishaan nema dovoljno istorije poseta ili sličnih korisnika za generisanje preporuka.

Korisnik sa ID-em 205 nije pronađen u matrici sličnosti korisnika.

Korisnik Riya nema dovoljno istorije poseta ili sličnih korisnika za generisanje preporuka.

...

Korisnik Aditya nema istoriju poseta ili nema preporuka.

Korisnik Hitesh nema dovoljno istorije poseta ili sličnih korisnika za generisanje preporuka.

Korisnik Hitesh nema istoriju poseta ili nema preporuka.

Mean NDCG: 0.0

Mean NDCG score: 0.4905

Top 5 recommendations for user 1:

1. Kerala Backwaters

2. Leh Ladakh

3. Goa Beaches

4. Taj Mahal

5. Goa Beaches

# Literatura

- <https://www.be-terna.com/sr/blog/sistemi-preporuka-u-elektronskoj-trgovini-sta-se-zaista-desava-iza-kulis>
- <https://www.youtube.com/watch?v=i4a0Of22QRg>
- <https://www.youtube.com/watch?v=mBcLRGuAFUk&t=293s>
- <https://www.youtube.com/watch?v=HAJey9-Q8js&t=2s>
- <https://www.youtube.com/watch?v=VZKMyTaLIOO&t=44s>
- <https://velog.io/@jiselectric/CV>
- <https://towardsdatascience.com/simple-svd-algorithms-13291ad2eef2>
- <https://www.geeksforgeeks.org/singular-value-decomposition-svd/>



終わり

(The end)

ニコリーナ・ドティグ