PERSONALIZOVANI SISTEM ZA PREPORUKU FILMOVA

Nevena Nikolić, 1021/2018

Luka Kalinić, 1058/2018

UVOD

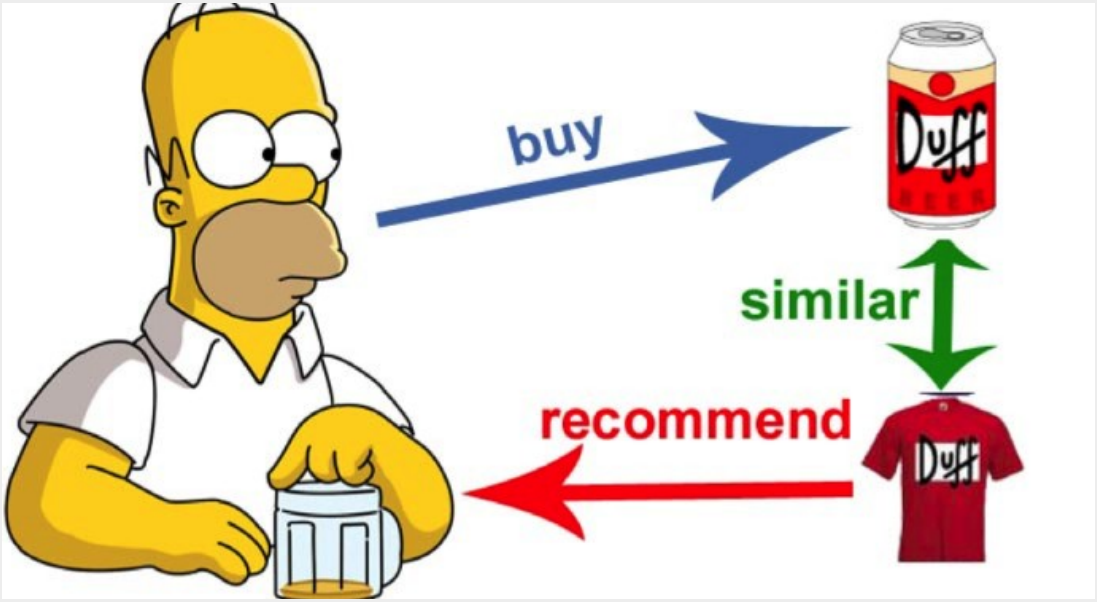
- Sistemi za preporuku su sistemi čiji je glavni zadatak da pruže korisniku informaciju - preporuku o potencijalno zanimljivom sadržaju i predstavljaju jednu od najuspešnijih primena mašinskog učenja
- Popularnost ovih sistema naročito je porasla poslednjih godina, povećanjem količine informacija koje nas okružuju, te digitalizacijom naše svakodnevnice i pojavom internet servisa, kao što su *Facebook, Google, Amazon, eBay, Netflix*...

RAZLIČITI PRISTUPI

- Sistemi za preporuku zasnovani na filtriranju sadržaja (eng. c*ontent-based filtering*)
- Sistemi za preporuku zasnovani na kolaborativnom filtriranju (eng. *collaborative filtering*)
- Hibridni sistemi (eng. hybrid)

FILTRIRANJE SADRŽAJA

- Korisniku se preporučuju objekti slični objektima koji su mu bili zanimljivi u prošlosti, poput objekata koje je ranije pretraživao ili ocenio visokom ocenom



FILTRIRANJE SADRŽAJA

PREDNOSTI

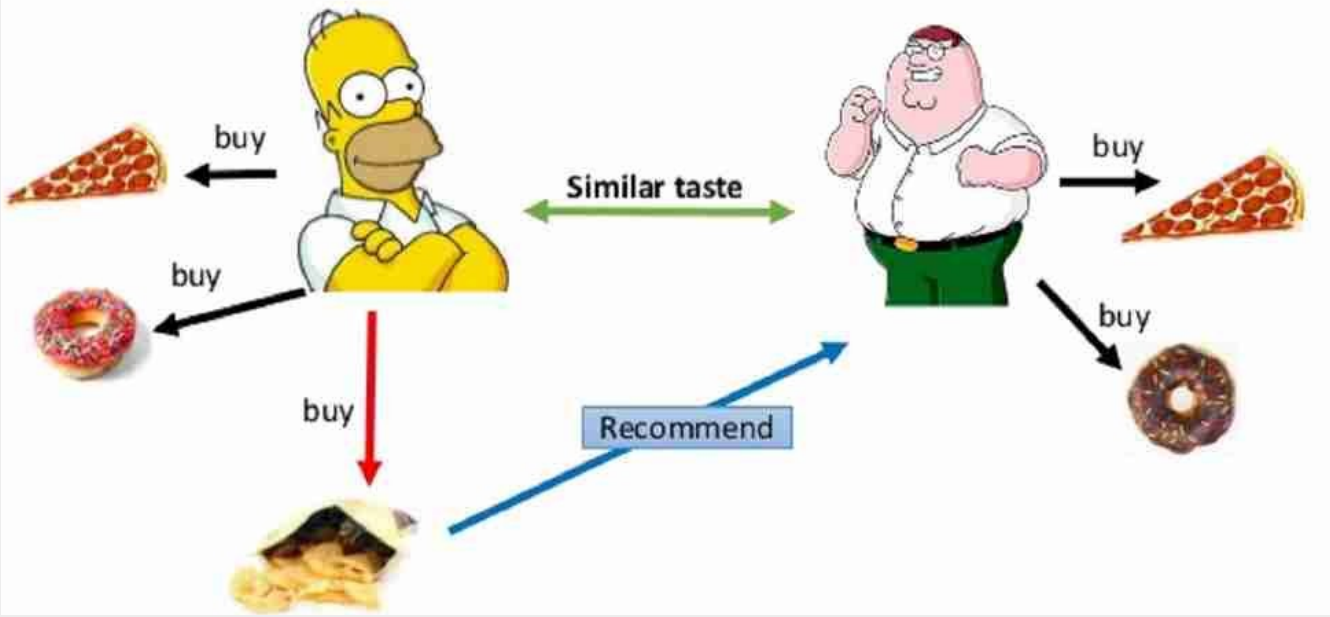
- nezavisnost od ostalih korisnika, nema formiranja zajednice
- mogućnost poređenja objekata
- mogućnost preporučivanja novih objekata koji prethodno nisu bili ocenjeni
- mogućnost preporučivanja korisnicima sa jedinstvenim ukusom
- transparentnost – moguće je jasno odrediti zašto je objekat preporučen

MANE

- generisanje atributa može biti komplikovano
- tendencija ka preteranoj specijalizaciji (eng. *overspecialization)* - preporučuju se jedino objekti koji su slični postojećim zanimljivim objektima
- problem novog korisnika - u slučaju novog korisnika, sistem ne može dati preciznu preporuku

KOLABORATIVNO FILTRIRANJE

- Korisniku se preporučuju objekti koji su bili zanimljivi korisnicima sa sličnim interesima (sa sličnim ukusom)



KOLABORATIVNO FILTRIRANJE

PREDNOSTI

- radi za proizvoljne objekte – generisanje atributa nije potrebno
- ignoriše kontektst
- jednostavnost implementacije
- bolje performanse u odnosu na content-based pristup (u smislu greške predviđanja i vremena izršavanja)

MANE

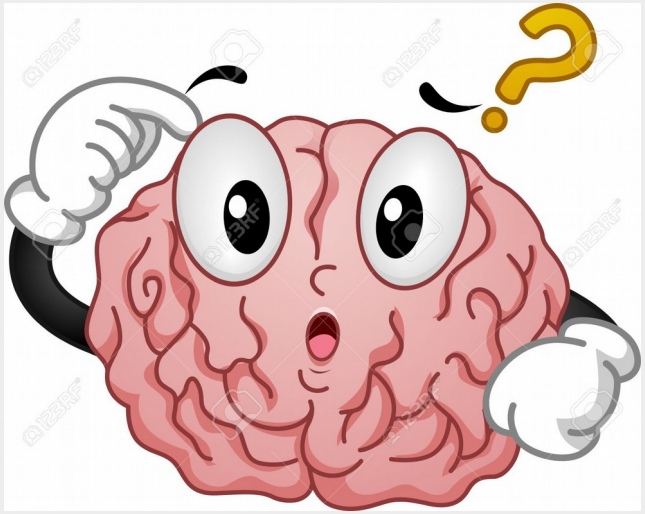
- problem novog objekta – ako nije dovoljno puta ocenjen, objekat neće biti preporučivan
- problem novog korisnika - potrebno je imati dovoljan broj korisnika u sistemu za nalaženje sličnog
- user/ratings matrica je retka, teško je pronaći korisnike koji su ocenili iste objekte
- tendencija ka preporučivanju popularnih objekta

PREPORUKA FILMOVA

- Naš cilj je izgradnja personalizovanog sistema za ocenjivanje i preporuku filmova
- Različiti ljudi imaju različit ukus za filmove; naš sistem pomaže korisnicima da odmah pronađu filmove po svom ukusu, bez obzira na to koliko njihovi ukusi mogu biti raznoliki

METODE

- Eksperimentišemo sa oba pristupa
- Filtriranje sadržaja:
  - TF-IDF
- Kolaborativno filtriranje:
  - K-najbližih suseda, dekompozicija matrice



PODACI

- Koristimo **The Movies** skup podataka
  - sadrži metapodatke za preko 45,000 filmova koji se nalaze u *Full MovieLens* skupu podataka
  - 26 miliona ocena od preko 270,000 korisnika
  - podaci kao što su uloge, filmska ekipa, ključne reči, budžet, prihodi, posteri, jezici, izdavačke kuće, broj TMDB glasova, prosečne ocene..
  - dostupna i datoteka koja sadrži 100,000 ocena od strane 700 korisnika za mali podskup od 9,000 filmova, koju ćemo mi koristiti zbog računarskih ograničenja

PODACI

- movies_metadata.csv
  - glavna datoteka koja sadrži metapodatke za preko 45,000 filmova
  - atributi kao što su naziv filma, žanrovi, posteri, budžet, prihod, datum izlaska, opis filma...
- keywords.csv
  - sadrži ključne reči radnje filma
- credits.csv
  - sadrži informacije o ulogama, režiserima, producentima..
- links_small.csv:
  - sadrži TMDB i IMDB identifikatore malog podskupa od oko 9,000 filmova
- ratings_small.csv
  - sadrži podskup od 100,000 ocena od strane 700 korisnika za oko 9,000 filmova

ANALIZA PODATAKA

- Top 10 filmova prema težinskim ocenama (eng. *weighted score*), izračunatih koristeći *IMDB weighting*

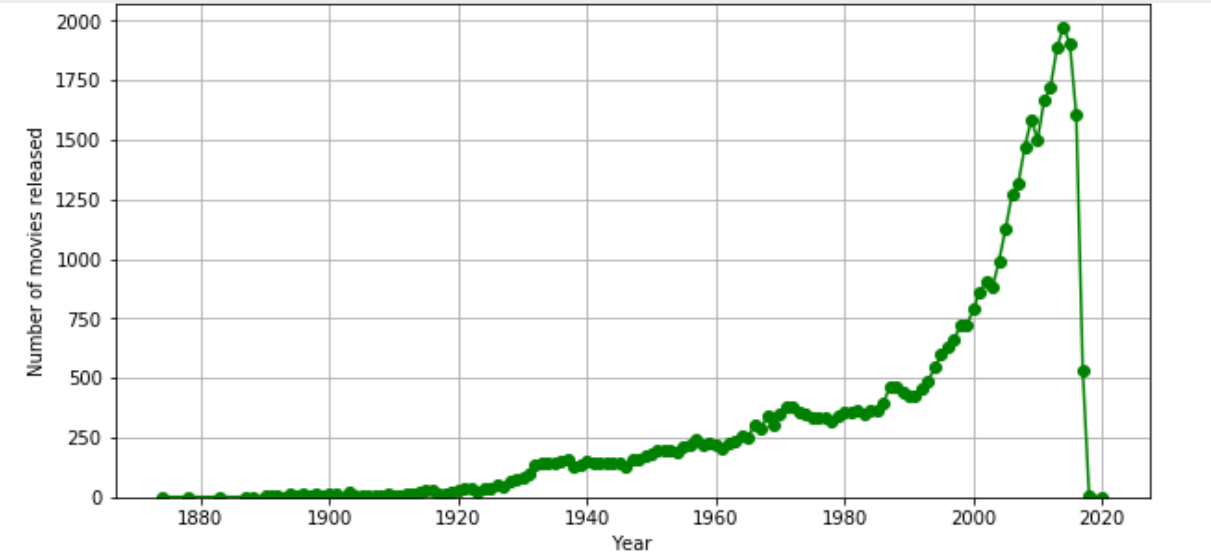
```
m = movies.vote_count.quantile(0.9)
C = movies.vote_average.mean()
qualified = movies[(movies['vote_count'] >= m) & (movies['vote_count'].notnull())
                   & (movies['vote_average'].notnull())]

v = qualified.vote_count
R = qualified.vote_average
weighted_score = (v/(v+m)*R) + (m/(v+m)*C)
weighted_score = round(weighted_score, 6)

movie_title = qualified.title

top_rated = np.vstack([movie_title,R,v,weighted_score]).T
top_rated = pd.DataFrame(top_rated,columns = ['Movie Title','Avg Votes','Num Votes','Weighted Score'])
top_rated = top_rated.sort_values('Weighted Score',ascending=False)
top_rated.head(10)
```

Movie Title	Avg Votes	Num Votes	Weighted Score
The Shawshank Redemption	8.5	8358	8.44587
The Godfather	8.5	6024	8.42544
Dilwale Dulhania Le Jayenge	9.1	661	8.42145
The Dark Knight	8.3	12269	8.26548
Fight Club	8.3	9678	8.25638
Pulp Fiction	8.3	8670	8.25141
Schindler's List	8.3	4436	8.20664
Whiplash	8.3	4376	8.2054
Spirited Away	8.3	3968	8.19605
Life Is Beautiful	8.3	3643	8.18717

ANALIZA PODATAKA

- Broj filmova po godini

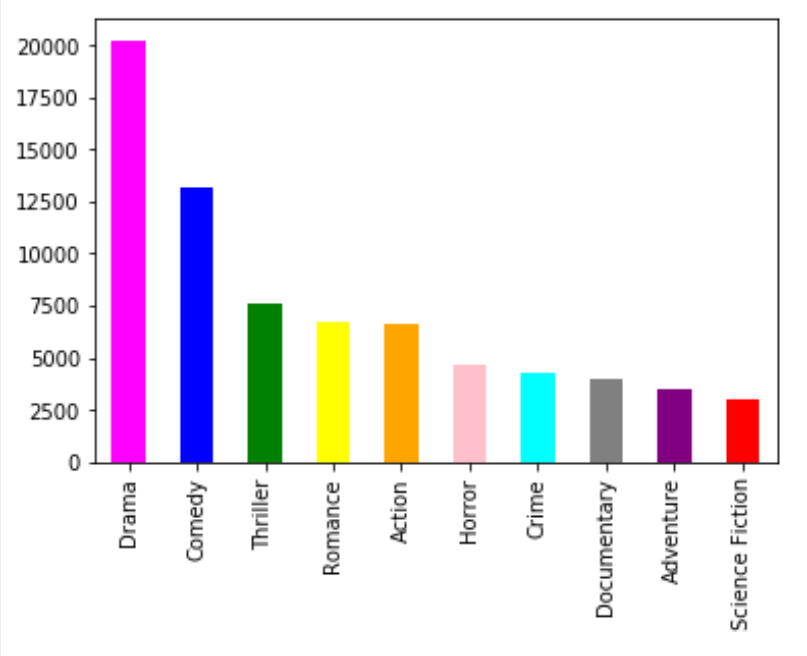
```
movies['year'] = movies.release_date.str.extract("(\d{4})", expand = True)
movies.year = pd.to_datetime(movies.year, format='%Y')
movies.year = movies.year.dt.year
dftmp = movies[['id', 'year']].groupby('year')
fig, ax1 = plt.subplots(figsize=(10,5))
ax1.plot(dftmp.year.first(), dftmp.id.nunique(), "g-o")
ax1.grid(None)
ax1.set_ylim(0,)
ax1.set_xlabel('Year')
ax1.set_ylabel('Number of movies released')
```



ANALIZA PODATAKA

- Broj filmova po žanru

```
def counting_values(df, column):
    value_count = {}
    for row in df[column].dropna():
        if len(row) > 0:
            for key in row:
                if key in value_count:
                    value_count[key] += 1
                else:
                    value_count[key] = 1
        else:
            pass
    return value_count

genres_count = pd.Series(counting_values(movies, 'genres'))
colors = ['magenta','blue','green','yellow','orange','pink','aqua','gray','purple','red']
genres_count.sort_values(ascending = False).head(10).plot(kind = 'bar',color = colors)
plt.show()
```



SISTEM ZA PREPORUKU FILMOVA ZASNOVAN NA FILTRIRANJU SADRŽAJA

TF-IDF

- Metrika *TF-IDF* koristi se za dodeljivanje težine nekom terminu
- Važnost termina (reči) ***t*** raste sa brojem njegovog pojavljivanja u dokumentu ***d***, ali se otežava njegovim ukupnim brojem pojavljivanja u celokupnom skupu dokumenata (koji ima više dokumenata)
- Za reč ***t*** u dokumentu ***d*** računamo *TF-IDF* meru kao proizvod sledeće dve veličine:
  - *TF(t, d)* - Broj pojavljivanja reči ***t*** u dokumentu ***d***
  - *IDF(t, d, D)* - Logaritam količnika ukupnog broja dokumenata ***d*** sa brojem dokumenata (iz skupa *D*) u kojima se pojavljuje reč ***t***

SLIČNOST FILMOVA

- Dakle, pomoću *TF-IDF* rečima u rečenici dajemo neku vrstu ocene važnosti
- Sledeći korak je računanje sličnosti
  - koristimo kosinusno rastojanje koje se definiše na sledeći način:

$$cosine(x, y) = \frac{x * y^T}{||x|| * ||y||}$$

- Računamo dve matrice sličnosti
  - prvu korišćenjem opisa filma i etiketa
  - drugu korišćenjem imena glumaca, imena režisera i ključnih reči

ZAŠTO DVE MATRICE SLIČNOSTI?

- Eliminišemo razmake između imena I prezimena da bismo izbegli konfuziju kod različitih ljudi sa istim imenom
- Opisi filmova i etikete su obično dugački, dok su imena glumaca I režisera samo jedna reč
- Njihovo kombinovanje u jedan vektor reči rezultiralo bi davanjem mnogo veće težine opisima
- Zato računamo dve matrice sličnosti i kombinujemo rezultate koristeći težinsku sumu koja minimizuje grešku na skupu za obučavanje

PREDVIĐANJE OCENA

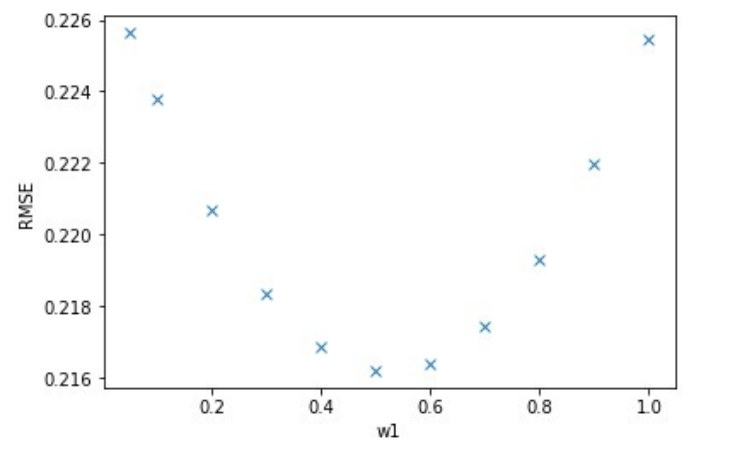
- Generisane matrice sličnosti potom koristimo da predvidimo korisnikovu ocenu za neki film
- Predviđanje ocene korisnika ***u*** za film ***i*** vrši se na osnovu formule:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_j \text{sim}(i,j)(r_j - \mu_u)}{\sum_j \text{sim}(i,j)}$$

  - ***sim(i,j)*** je sličnost filmova ***i*** i ***j***
  - r_j je ocena koju je korisnik dao filmu ***j***
  - μ_u je prosečna ocena korisnika ***u***

TEŽINSKA SUMA

- Finalno predviđanje korisnikove ocene za neki film:
  - $w_1 * prediction_1 + w_2 * prediction_2$
  - $prediction_1$ predviđanje dobijeno na osnovu prve matrice sličnosti
  - $prediction_2$ predviđanje dobijeno na osnovu druge matrice sličnosti
- w_1, w_2 koji minimizuju RMSE (*root mean squared error*) na skupu za obučavanje



SISTEM ZA PREPORUKU FILMOVA ZASNOVAN NA KOLABORATIVNOM FILTRIRANJU

KNN PRISTUP

- Koristimo algoritam K najbližih suseda (*K-Nearest Neighbors, KNN*)
- Ocene se predviđaju na osnovu:
  - sličnosti među korisnicima (*user-based collaborative filtering*)
  - sličnosti među filmovima (*item-based collaborative filtering*).
- Preporučuju se oni filmovi za koje je predviđena ocena najviša
- Kao meru sličnosti koristimo kosinusno rastojanje
- Podatke predstavljamo kao matricu u kojoj vrste odgovaraju korisnicima a kolone filmova
- Delimo podatke na skupove za obučavanje i testiranje u razmeri 4:1

KNN PRISTUP

- Podatke predstavljamo kao matricu
  - vrste odgovaraju korisnicima
  - kolone odgovaraju filmovima
  - element ***(i,j)*** matrice predstavlja ocenu 1-5 koju je korisnik ***i*** dao filmu ***j***, ili nula u slučaju da korisnik nije ocenio dati film
- Ovakva matrica je dosta retka
  - svaki korisnik je ocenio relativno mali udeo svih filmova
- Delimo podatke na skupove za obučavanje i testiranje u razmeri 4:1

KNN PRISTUP

```
Non zero count - train:  79748
Density - train:  1.311 %
Non zero count - test:  20256
Density - test:  0.333 %
```

```python
def k_nearest_neighbors(i, similarity_matrix, k=10):

    neighbors = []
    similarities = list(zip(similarity_matrix[i][:],range(similarity_matrix.shape[0])))
    similarities.sort(key = lambda x: x[0], reverse=True)
    for i in range(1,k+1):
        neighbors.append(similarities[i][1])

    return neighbors
```

USER-BASED KNN

- Računamo matricu sličnosti između korisnika
- Predviđanje ocene korisnika ***u*** za film ***i*** vrši se na osnovu formule:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v)(r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

  - $N_i^k(u)$ je skup ***k*** najsličnijih korisnika korisniku ***u***
  - ***sim(u,v)*** je sličnost korisnika ***u*** i ***v***
  - r_{vi} je ocena koju je korisnik ***v*** dao filmu ***i***
  - μ_u je prosečna ocena korisnika ***u***

ITEM-BASED KNN

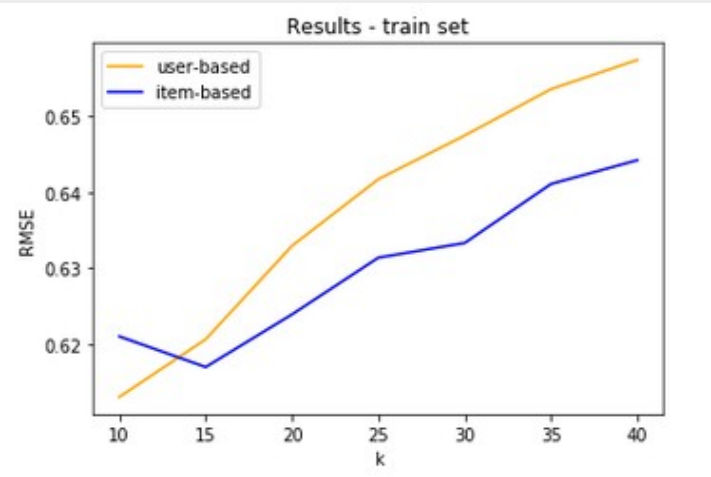
- Računamo matricu sličnosti između filmova
- Predviđanje ocene korisnika ***u*** za film ***i*** vrši se na osnovu formule:

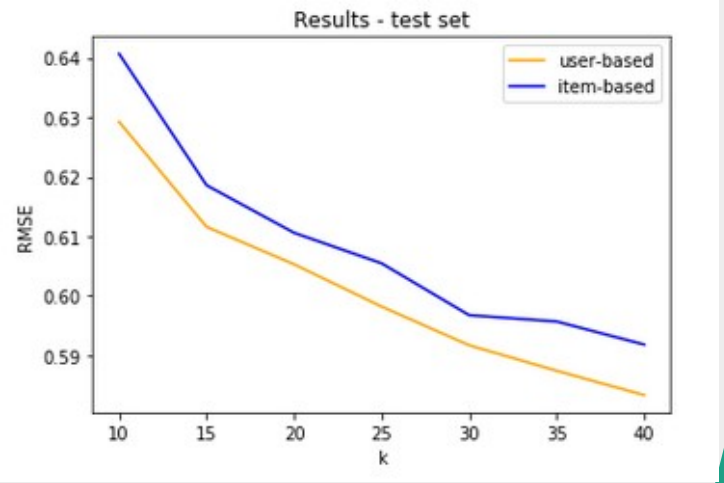
$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} \mathrm{sim}(i,j)(r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} \mathrm{sim}(i,j)}$$

  - $N_u^k(i)$ je skup *k* najsličnijih filmova filmu ***i***
  - ***sim(i,j)*** je sličnost filmova ***i*** i ***j***
  - r_{uj} je ocena koju je korisnik ***u*** dao filmu ***j***
  - μ_i je prosečna ocena za film ***i***

REZULTATI

- Tražimo broj suseda ***k*** koji minimizuje RMSE na skupu za obučavanje
- Primećujemo da se optimalne vrednosti parametra *k* razlikuju na skupu za obučavanje i testiranje





SVD PRISTUP

- Pristup se zasniva na singularnoj dekompoziciji matrice (*Singular Value Decomposition*, SVD)
- Matrica ocena kao $R=M\Sigma U^T$
  - U je matrica korisnika
  - M je matrica filmova
  - Σ je dijagonalna matrica
- Jednostavnosti radi, možemo posmatrati jednakost bez dijagonalne matrice, odnosno samo $R=MU^T$
- Jednakost na osnovu koje se vrši predviđanje ocena:
  - $r_{ui}= p_u * q_i$
  - p_u je red matrice M koji se odnosi na korisnika u
  - qi je kolona matrice U^T koja se odnosi na film i

SVD PRISTUP

- Koristimo funkciju **svds** iz paketa **scipy.sparse.linalg**

```
M, sigma, Ut = svds(train, k = 4)
```

- Predviđanje ocene:

```
predictions_svd_matrix = np.dot(np.dot(M,sigma),Ut)
```

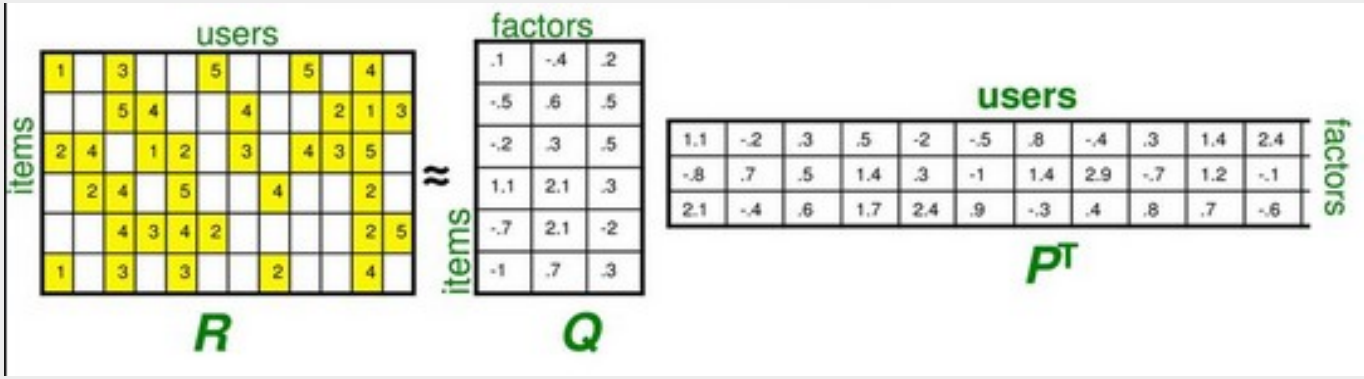
- Rezultati:

```
Singular Value Decomposition train RMSE:  2.956196813823273
```

```
Singular Value Decomposition test RMSE:  3.049447802265524
```

METOD LATENTNIH FAKTORA

- *R* je matrica ocena
- Dekomponujemo je (faktorišemo) na dve matrice manjih dimenzija *P* i *Q*



- U matrici *P* svaki red se objašnjava kao korisnikovo interesovanje za skrivene faktore *F1*, *F2* and *F3*, odnosno meri koliko je korisnik zainteresovan za film određenih karakteristika
- U matrici *Q* svaka kolona predstavlja film opisan pomoću skrivenih faktora, odnosno meri koliko određeni film poseduje određene karakteristike

METOD LATENTNIH FAKTORA

- Ovakva dekompozicija podseća na ono što se dešava tokom SVD ili analize glavnih pravaca (PCA)
- Cilj je pronaći latentne faktore i smanjiti dimenzionalnost
- Zamislimo da imamo ogromnu matricu ocena R I da smo redukovali dimenzionalnost na određen broj komponenti (skrivenih faktora)
- Za to možemo koristiti PCA ili SVD
- Problem je što je matrica ocena dosta retka: ne znamo sve ocene za sve korisnike i sve filmove

OPTIMIZACIONI PROBLEM

$r_{1,1} = p_1 \cdot q_1$

$r_{2,1} = p_2 \cdot q_1$

.....

$r_{u,i} = p_u \cdot q_i$

.....

$r_{m,n} = p_n \cdot q_n$

- Treba rešiti sistem jednačina prikazan na slici
- Rešavamo samo za one ocene koje postoje u skupu za obučavanje
- Nalazimo skup faktor-vektora P_u za svakog korisnika u i Q_i za svaki film i
- Dobijamo optimizacioni problem

$$min_{q^*,p^*,b^*} \sum_{(u,i)\in s} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

- Da bismo izbegli preprilagođavanje (preveliki broj skrivenih faktora) dodajemo regularizacioni izraz

STOHASTIČKI GRADIJENTNI SPUST

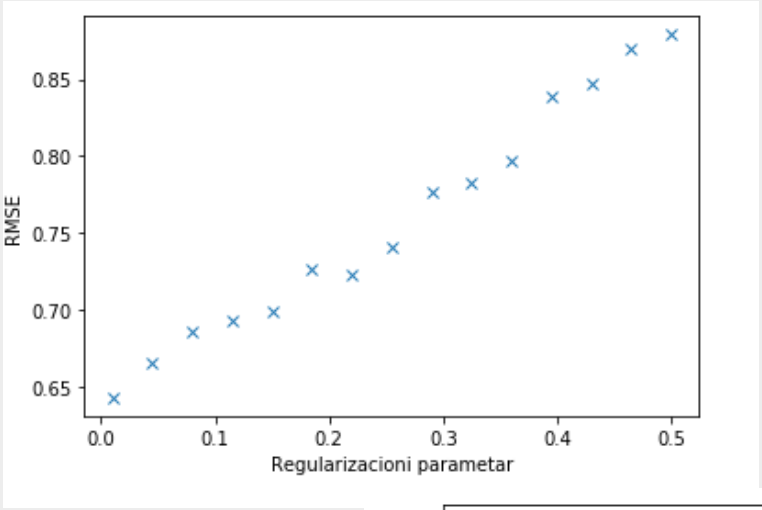
- Da bismo rešili prethodni optimizacioni problem koristimo stohastički gradijentni spust (*Stohastic Gradient Descent*, SGD)
- Najpre inicijalizujemo tražene matrice P i Q nasumičnim vrednostima iz unifomne raspodele
- Zatim za svaki par (korisnik-film, ocena) iz skupa za obučavanje ažuriramo parametre prema grešci između stvarne i predviđene ocene
- Na primer, faktor p_u se ažurira na sledeći način:

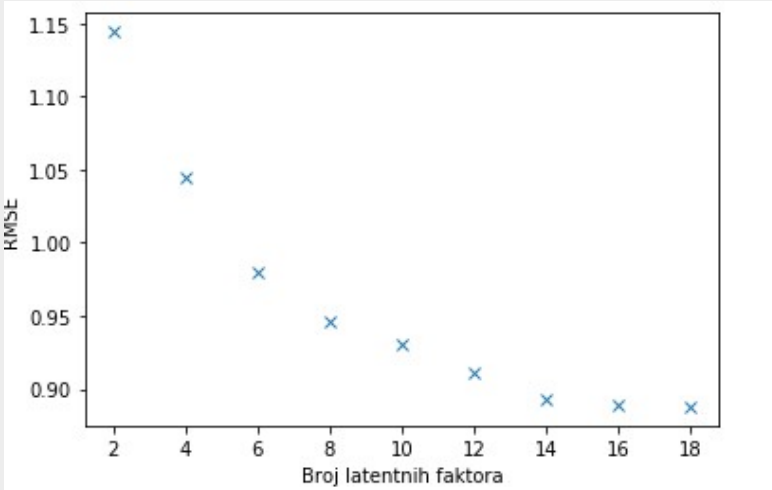
$$p_u = p_u + \alpha * ((r_{ui} - r'_{ui})q_i - \lambda p_u)$$

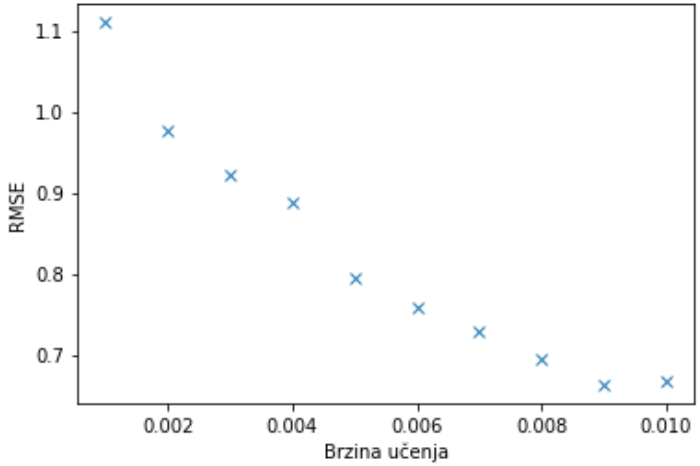
  - α je brzina učenja (*learning rate*)
  - λ je regularizacioni parametar

STOHASTIČKI GRADIJENTNI SPUST

- Treba odabrati pogodne parametre tako da se minimizuje RMSE na skupu za obučavanje







HVALA NA PAŽNJI! :)