**[Event Recommendation Engine](https://www.kaggle.com/c/event-recommendation-engine-challenge)**

Boris Fidler i Marko Buljandrić

2. mart 2018

1. Uvod

U ovom radu bavili smo se rekomender sistemima. Dobijenim datasetom korisnika i događaja, zadatak nam je bio da predvidimo za koji događaj će korisnik biti zainteresovan. Dobili smo 38000 korisnika, 3 miliona događaja i pregršt podataka o njima kao što su: prijatelji, prisustvovanje i zainteresovanost za događaj, itd.

2. Predprocesiranje

U proces predprocesiranje smo upakovali prve tri faze CRISP-DM metodologije: Business Understanding, Data Understanding i Data Preparation. Pored toga u našu fazu predprocesiranja spada i klaserovanje, sređivanje lokacijskih podataka i sređivanje vremenskih podataka.

Business Understanding

Samo razumevanje rekomender sistema je navedeno u uvodu. Poenta je da određenom korisniku, na osnovu njegovih dosadašnjih izjašnjavanja, preporučimo događaje za koje bi oni bili zainteresovanicx.

Data Understanding

Celokupan dataset ima 6 fajlova: train.csv, test.csv, users.csv, user\_friends.csv, events.csv, and event\_attendees.csv. Od svih, koristićemo sledeće: train.csv, users.csv, events.csv i event\_attendees.

**Na osnovu toga da li su pozvani ili ne, zainteresovani ili ne, možemo kreirati 6 kategorija korisnika.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kategorija | Invited | Interested | Not Interested | Example |
| A | 1 | 1 | 0 | User was invited, visited the page and clicked interested. |
| B | 1 | 0 | 1 | User was invited, visited the page and clicked not interested. |
| C | 1 | 0 | 0 | User was invited, but either didn't visit the event page, or visited the page, and clicked neither. |
| D | 0 | 1 | 0 | User was not invited, but visited the event page and clicked interested. |
| E | 0 | 0 | 1 | User was not invited, but visited the event page clicked not interested. |
| F | 0 | 0 | 0 | User was not invited, but visited the event page, and clicked neither |

Slika 1

Smatramo da kategorije C i F nisu značajne za dalju analizu, stoga će biti isključene.

Data preparation

Inicijalnu fazu spremanja podataka smo opisali u sledećem delu teksta. Spremanje podataka se ne može izvršiti u jednoj fazi, daljim radom na projektu smo imali konstantnu potrebu za sređivanjem podataka.

**event**

Preimenovane kolone: event>event\_id; user>user\_id;

**event\_attendees**

Kreirane 4 nove kolone: yes\_count, no\_count, maybe\_count i invited\_count

event\_attendees['yes\_count']=event\_attendees.yes.str.count(' ')+1

event\_attendees['no\_count']=event\_attendees.no.str.count(' ')+1

event\_attendees['maybe\_count']=event\_attendees.maybe.str.count(' ')+1

event\_attendees['invited\_count']=event\_attendees.invited.str.count(' ')+1

Nakon toga, dropovane su kolone: yes, maybe, no, invited.

Dakle, za svaki event: koliko korisnika je reklo da je zainteresovano, nije zainteresovano, možda zainteresovano i koliko korisnika je bilo pozvano.

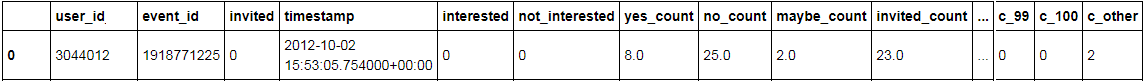
Tabeli train su pripojene tabele: event\_attendees, users i events.

train=train.merge(event\_attendees1, on='event\_id', how='left')

train1=train.merge(users, on='user\_id', how='inner')  
train2=train1.merge(events, on='event\_id', how='inner')

U novoj Train tabeli, popunjene sve null vrednosti sa 0.

Finalna verzija Train tabele nakon predprocesiranja izgleda ovako:



Clustering

Pre nego što smo počeli sa klasterovanjem, postojala je potreba da se ponovo spreme podaci. Kreirali smo dataset „train\_cluster“. Ideja je bila da se klasterovanje odradi na ključnim rečima. Ključne reči su organizovane po kolonama c1...c101 (c101=c\_others). U delu razumevanja podataka zaključili smo da su te kolone ključne reči za određeni event. Da bi se lakše objasnila poenta, sledi improvizacija:

eventA = "The quick brown fox jumps over the lazy dog"

eventB = "The quick brown fox jumps over the lazy cat"

c1 = "the"

c2 = "quick"

c3="brown"

c4="dog"

c5="cat"

c\_other = other

eventA = 2, 1, 1, 1, 0, 4

eventB= 2, 1, 1, 0, 1, 4

Dropovali smo sve kolone osim: event\_id i c1-100. Smatramo da kolona c\_other(c\_101) nema svrhe u klasterovanju, jer se u njoj nalaze sve „other“ reči, koje nemaju tolikog značaja.

Izbacili smo duplirane slogove i popunili null vrednosti:

train\_cluster.drop\_duplicates(keep='first', inplace=**True**)

train\_cluster.fillna(0, inplace=True)

Klasterovanje rađeno putem Kmeans-a.

KMeans algoritam skuplja podatke pokušavajući da odvoji uzorke u N grupama jednake varijanse, minimizirajući kriterijum poznat kao inercija ili „within-cluster sum-of-squares“. Ovaj algoritam zahteva određivanje broja klastera. Dobro se meri na velikom broju uzoraka i koristi se u velikom broju područja primene u mnogim različitim poljima. Algoritam k-means deli niz N uzoraka X u K „diskove“ koji su opisani putem „mean“ uzoraka u klasteru. Sredstva se obično nazivaju klaster centroidi. Algoritam K-means ima za cilj da izabere centroide koji umanjuju inerciju ili sumu kvadratnog kriterijuma unutar klastera.



**import** **sklearn.cluster**

**from** **sklearn.cluster** **import** KMeans

**from** **sklearn** **import** cluster

**from** **sklearn.metrics** **import** silhouette\_score, silhouette\_samples

x=[]

y=[]

range\_n\_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

**for** n\_clusters **in** range\_n\_clusters:

x.append(n\_clusters)

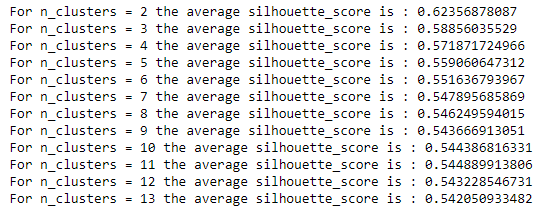
k\_means\_normalized = cluster.KMeans(n\_clusters=n\_clusters, random\_state=100)

cluster\_labels = k\_means\_normalized.fit\_predict(train\_cluster)

silhouette\_avg = silhouette\_score(train\_cluster, cluster\_labels)

y.append(silhouette\_avg)

print("For n\_clusters =", n\_clusters, "the average silhouette\_score is :", silhouette\_avg)



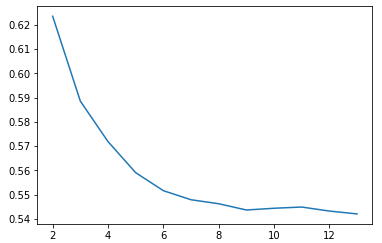
Prikazali smo klastere na plotu, kako bi na grafiku videli „lakat“ tj. tačku preloma:

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

plt.plot(x,y)

plt.show()



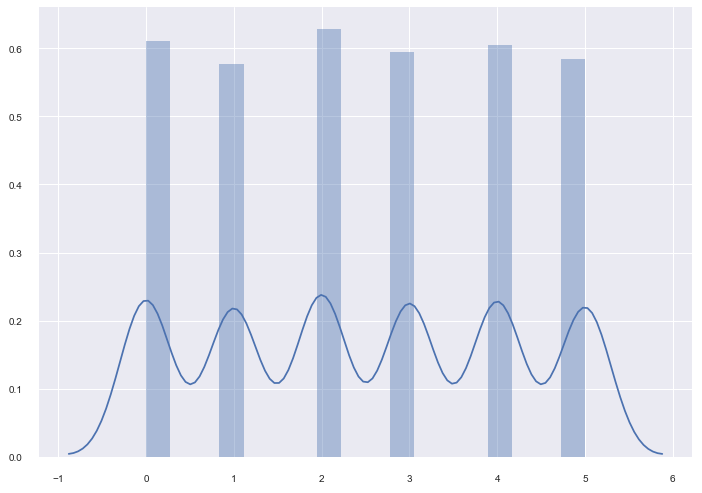
Na ovom grafiku vidi se da je „lakat“ na ~ 6. klasteru, stoga se može zaključiti da je optimalan broj klastera 6. Nakon što smo odredili optimalan broj klastera, prikazujemo ih na grafiku:  
km = KMeans(n\_clusters=6, random\_state=1) km.fit(train\_cluster) predict=km.predict(train\_cluster)

predict  
array([2, 2, 4, ..., 3, 3, 4], dtype=int32)

**import** **matplotlib.pyplot** **as** **plt** %matplotlib inline **import** **seaborn** **as** **sns** sns.set(rc={'figure.figsize':(11.7,8.27)})

sns.distplot(predict)

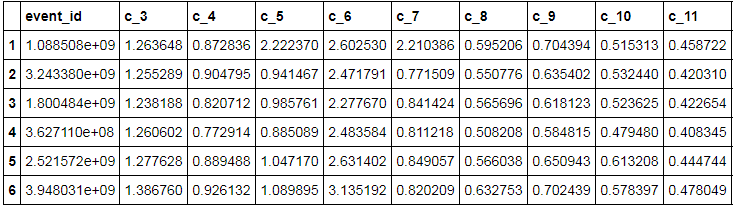
sns.despine(offset=10, trim=**True**)



Nakon prikazanih klastera, kreirali smo predstavnike centroida za svaki klaster:

cluster\_representatives=pd.DataFrame(km.cluster\_centers\_, columns= train\_cluster.columns, index = range(1,7))

cluster\_representatives



Sređivanje lokacijskih podataka

Nakon dalje analize dataseta i našeg zadatka, recommender sistema, uočili smo da nam je lokacija (korisnika i događaja) izuzetno važan podatak, a te kolone nisu sređene. Naime, neki od redova poseduju samo adresni podatak (City, State i Country) dok neki od redova poseduju samo podatke o koordinatama (latitude i longitude).

Od 3137972 reda, 1311299 ne poseduje ništa od lokacijskih podataka (City, State, Country, Latitude i Longitude) stoga su dropovani. Nema poente preporučivati korisnicima događaje za koje ne postoji podatak gde se nalaze. Ostalo je 1826673 reda.

Za sređivanje geo podataka, koristili smo Mapbox.

Mapbox je komercijalna lokacijska data platforma koja se koristi za određivanje lokacije na osnovu ulaznih parametera. U zavisnosti od ulaznih parametara, razlikuju se dve tehnike geokodiranja, forward i reverse. Forward tehnika se koristi kako bi na osnovu adresnih podataka (City, State i Country) dobili koordinate. Dok se reverse tehnika koristi za dobijanje adresnih podataka na osnovu koordinata.   
!pip install mapbox

**import** **mapbox**

geocoder = mapbox.Geocoder(access\_token='pk.eyJ1IjoibWlsb3NtODMiLCJhIjoiY2pkcGM1aXJhMWs4MzMzcWhpMHpobW9paCJ9.smkRAuDMn-2txP\_ybEiqqA')

user\_lat=response.geojson()['features'][1]['geometry']['coordinates'][1] user\_lng=response.geojson()['features'][1]['geometry']['coordinates'][0]

unique\_user\_locations['location'].str.strip() unique\_user\_locations['location'].replace('', np.nan, inplace=**True**) unique\_user\_locations.dropna(inplace=**True**)

unique\_user\_locations['user\_lat']=''

unique\_user\_locations['user\_lng']=''

**for** i, row **in** unique\_user\_locations.iterrows(): response=geocoder.forward(row.location) **if** len(response.geojson()['features'])>0: unique\_user\_locations.set\_value(i,'user\_lat',response.geojson()['features'][0]['geometry']['coordinates'][1]) unique\_user\_locations.set\_value(i,'user\_lng',response.geojson()['features'][0]['geometry']['coordinates'][0])

Pored toga, napravili smo kolonu u kojoj će biti podaci o distanci. Ta kolona predstavlja udaljenost između korisnika i događaja.  
  
S obzirom da distanca ne može biti prava linija, zbog toga što Zemlja nije ploča, koristili smo „vincenity“. Vincenity meri udaljenost, na osnovu geografske dužine, širine i zakrivljenosti globusa.

!pip install vincenty  
  
def distance\_calc (row):  
  
   start = (row['lat'], row['lng'])  
  
   stop = (row['user\_lat'], row['user\_lng'])  
  
  
  
   return vincenty(start, stop)  
  
  
train2['distance'] = train2.apply (lambda row: distance\_calc (row),axis=1)  
  
  
Sređivanje vremenskih podataka  
  
Preimenovali smo kolone: timestamp > time\_saw\_event; joinedAT > join\_date; start\_time > event\_start\_time.

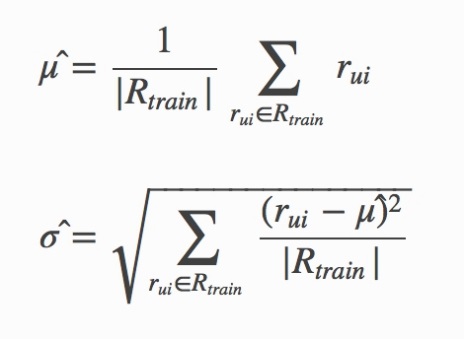
Kreirali smo kolonu day of the week. Dan u nedelji može biti dobarpodatak za analizu.  
  
train2["day\_of\_week\_of\_event"]=train2["event\_date"].dt.weekday

3. IZBOR ALGORITAMA I KREIRANJE MODELA

Algoritmi koje smo koristili su:

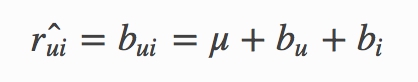
NormalPredictor

Algoritam koji predviđa nasumične ocene zasnovane na distribuciji naseg trening seta, koji predviđa normalnu distribuciju.



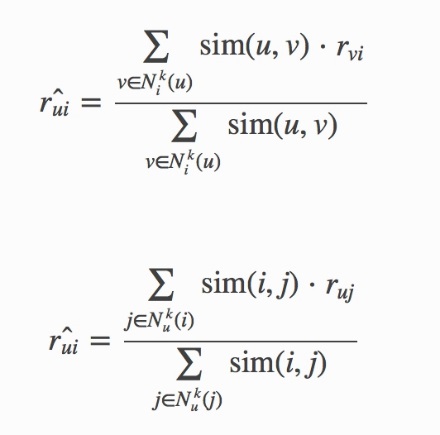
BaselineOnly

Algoritam predviđa baseline za svakog usera i svaki event. Ako user nije poznat pretpostavlja se da je 0, isto važi i za item.



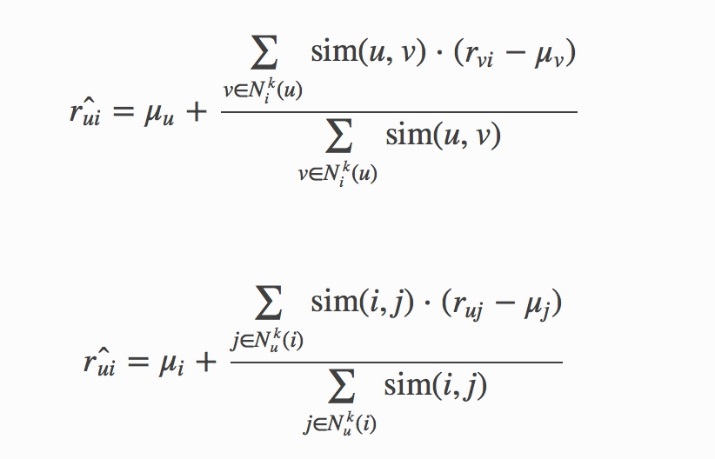
KNN Basic

Osnovni kolaborativni filtering algoritam.



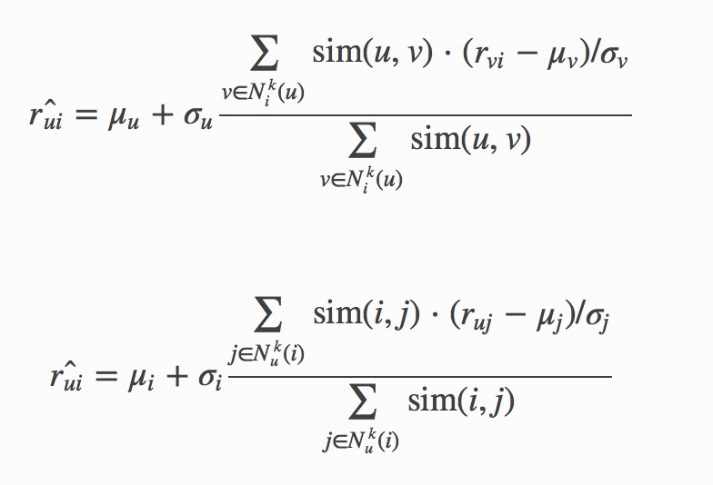
KNN with means

Osnovni kolaborativni filtering algoritam koji uzima u obzir i srednje vrednosti rejtinga svakog usera.



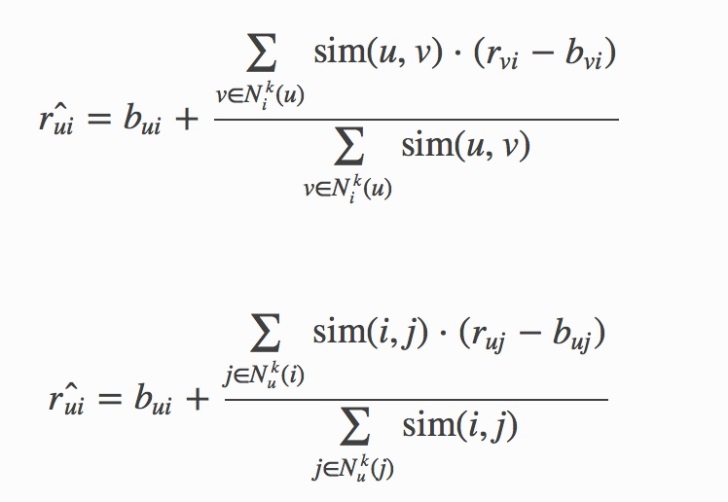
KNN with Z score

Osnovni kolaborativni filtering algoritam koju uzima Z skor normalizacije svakog usera.



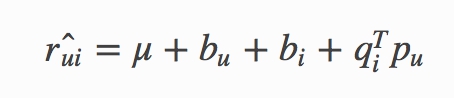
KNN Baseline

Osnovni kolaborativni filtering algoritam koji uzima u obzir baseline rejting.

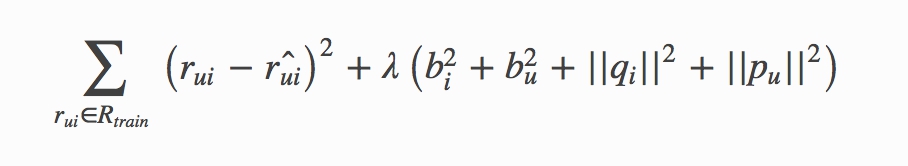


SVD

Čuveni SVD algoritam popularizovan od strane Simon Funk-a tokom Netflix takmičenja. Kada se Baseline ne koristi to je ekvivalentno probabilističkoj matrici faktorizacije prikazanoj ispod:

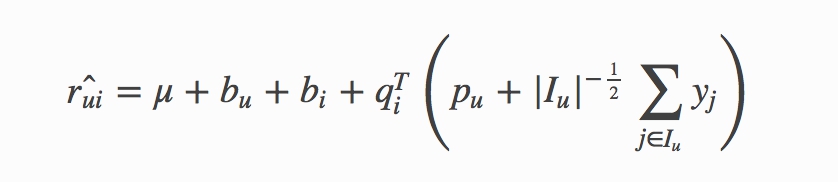


Isto kao sa Baseline algoritmom ako korisnik nije poznat onda su byas i faktori. Da bi predvideli nepoznato koristimo minimalizaciju sledećih regularizacionih kvadratnih grešaka.



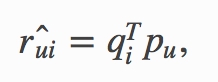
SVD++

Je algoritam koji je nastavak SVD uzimajući u obzir implicitne rejtinge.



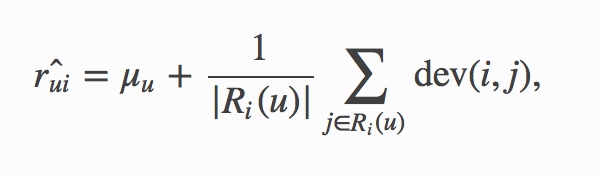
NMF

Kolaborativni filtering algoritam zasnovan na ne-negativnoj matrici faktorizaciji.



SlopeOne

Jednostavan ali ipak veoma tačan algoritam za kolaborativno filtriranje.



Co-Clustering

Kolaborativni filtering algoritam zasnovan na co-klasterovanju u kome se tehnički, korisnici i stavke dodeljuju nekom klasteru i nekom co-klasteru.

