**Slovenská technická univerzita v Bratislave**

**Fakulta informatiky a informačných technológií**

**Ilkovičova 2 , 842 16 Bratislava**

**Zadanie č.2**

Information retrieval

**Cvičiaci:​** doc. Ing. Michal Kompan, PhD.

Cvičenie: Ut, 8:00  
**Autor:​** Bc. Jozef Varga   
**Študijný program:​** Inteligentné softvérové systémy  
**Predmet:​** Vyhľadávanie informácií  
**Akad. rok:​** 2019/2020

Obsah

[2. Úvod 3](#_Toc27219406)

[3. Načítanie a čistenie dát 4](#_Toc27219407)

[Základné informácie o datasete 4](#_Toc27219408)

[Základné informácie o dátach 5](#_Toc27219409)

[4. Tvorba odporúčaní 7](#_Toc27219410)

[Príprava dát pre odporúčania 7](#_Toc27219411)

[Tvorba odporúčania 8](#_Toc27219412)

[5. Testovanie 9](#_Toc27219413)

[Kaggel 9](#_Toc27219414)

[Mean average precision (mAP) 9](#_Toc27219415)

[Normalized Discounted Cumulative Gain (mDCG) 10](#_Toc27219416)

[Úprava dát na základe testov 11](#_Toc27219417)

[6. Príloha A – Tvorba odporúčaní 13](#_Toc27219418)

# Úvod

Úlohou zadania bolo vytvorenie odporúčania pre zákazníkov ktorý nakupujú v obchode z ktorého sme dostali dataset. Tieto dáta obsahovali videné a kúpené produkty. Toto zadanie bolo riešené v jazyku Python, na ktorý som použil prostredie jupyter notebook. Dataset som spracoval pomocou knižnice Pandas a knižnica Sklearn mi umožnila využívať k-NN. Nižšie je vidno knižnice ktoré som v tejto práci využil:

from scipy import sparse

from scipy.sparse import csr\_matrix

from sklearn.model\_selection import train\_test\_split

from pandas.api.types import CategoricalDtype

from sklearn.neighbors import NearestNeighbors

from datetime import datetime

from tqdm import tqdm

from pandas import DataFrame

import pandas as pd

import numpy as np

Na tvorbu odporúčania som využil Collaborative Filtering Založené na User-base (viď obrázok nižšie)

Obrázok, na ktorom je text, mapa

Automaticky generovaný popis

Obrázok 1 Colaborative Filtering user-base

# Načítanie a čistenie dát

## Základné informácie o datasete

Dataset ktorý sme dostali, bol pomerne veľký. Skladal sa z dvoch súborov a to:

* Purchases\_train.csv
  + počet záznamov: 188 712
  + veľkosť súboru: 36MB
* Events\_train.csv
  + počet záznamov: 14 614 38
  + veľkosť súboru: 2,77GB

Oba tieto súbory obsahovali dáta s rovnakými stĺpcami a to customer\_id, timestamp, event\_type, product\_id, title, category\_name a price. Spolu sa jednalo o dataset s počtom 14 803 097 záznamov. Ako prvé bolo potrebné získať informácie o dátach. Keďže sme z prednášok daného predmetu získali prvotné informácie o stĺpcoch, a v rámci predmetu prebieha taktiež súťaž, ktorá má svoje pravidlá a sleduje predovšetkým product\_id. Rozhodol som sa používať iba stĺpce:

* customer\_id – id predstavujúceho customera
* timestamp – čas kedy vznikol daný záznam
* event\_type – hodnota prezentujúca či bol produkt zakúpený alebo len prehliadaný
  + purchase\_item – zakúpený produkt
  + view\_item – prehliadaný produkt
* product\_id – id predstavujúce produkt.

Obrázok, na ktorom je noviny, text

Automaticky generovaný popis

Obrázok 2 Ukážka dát

## Základné informácie o dátach

Následne bolo nutné získať informácie o dátach. Na začiatok som zistil počty unikátnych zákazníkov a produktov:

|  |  |  |
| --- | --- | --- |
| Dataset | Typ | Počet |
| purchases\_train | Počet unikátnych zakazníkov | 71 566 |
| events\_train | Počet unikátnych zakazníkov | 2 210 171 |
| spolu | Počet unikátnych zakazníkov | 2 211 386 |
| purchases\_train | Počet unikátnych produktov | 33 613 |
| events\_train | Počet unikátnych produktov | 123 724 |
| Spolu | Počet unikátnych produktov | 123 837 |

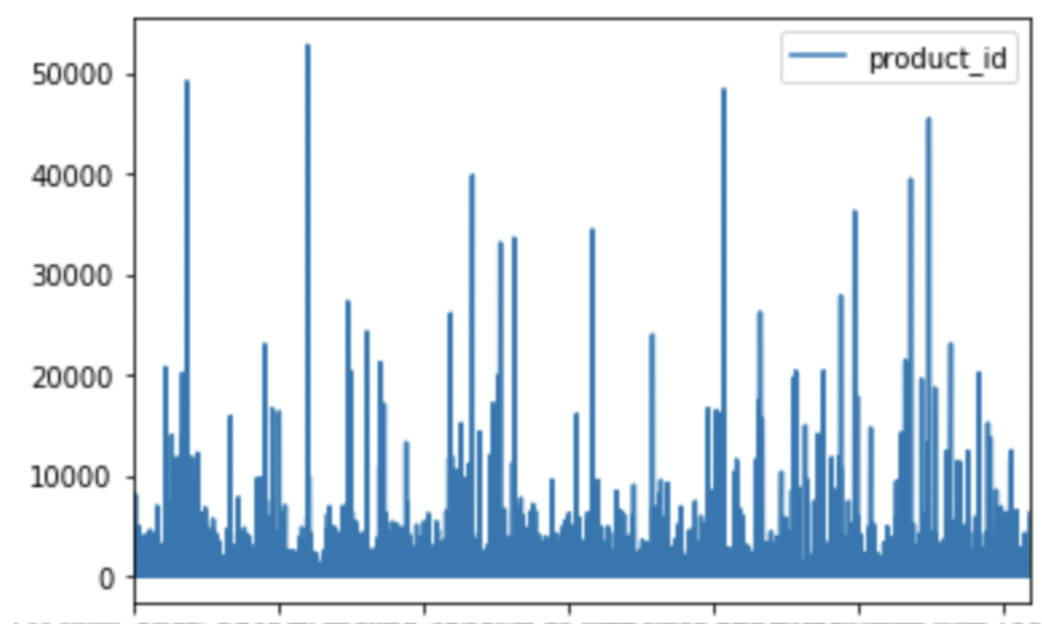
Tabuľka 1 Unikátne hodnoty

Potom som sa pozrel koľko zákazníkov zakúpilo aký počet produktov:

|  |  |  |  |
| --- | --- | --- | --- |
| Počet záznamov | events\_train | purchases\_train | Spolu |
| 1 | 915242 | 30843 | 913643 |
| 2 | 660220 | 35058 | 659174 |
| 3 | 598080 | 24684 | 597525 |
| 4 | 519336 | 22480 | 518884 |
| 5 | 486945 | 13150 | 485995 |
| 6 | 434868 | 13236 | 434964 |
| 7 | 399560 | 7266 | 399378 |
| 8 | 362288 | 7912 | 362408 |
| 9 | 341181 | 5058 | 342324 |
| 10 | 316090 | 4800 | 315480 |
| 11 a viac | 9580575 | 24225 | 9773322 |

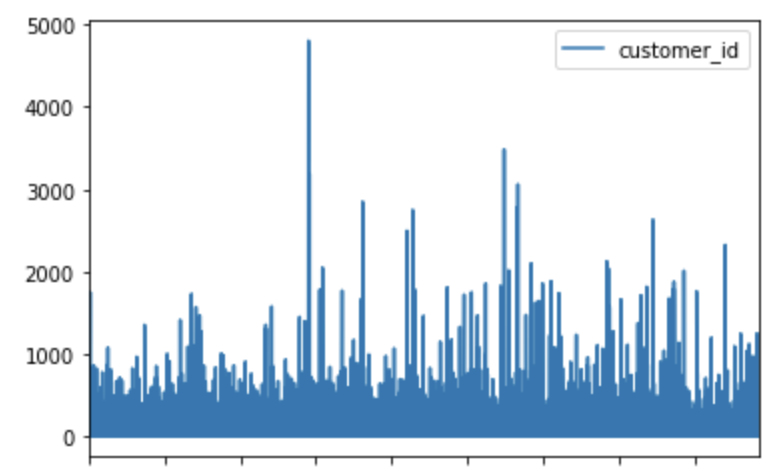
Tabuľka 2 Počet záznamov na jedného používateľa

Čo sa týka počtu produktov, obr. č.2 hovorí o tom, že je pár produktov, ktoré boli veľmi žiadané.



Obrázok 3 Počet, koľko krát sa produkt nachádza v datasete

Taktiež je z obr. č.3 vidieť že existuje veľa zákazníkov, ktorý majú v datasete viac ako 1000 záznamov.



Obrázok 4 Počet, koľko krát sa zákazník nachádza v datasete

Na základe získaných informácií som sa rozhodol odstrániť zákazníkov, ktorý sa nachádzali v datasete menej ako 5 krát. Ďalšie úpravy sú spomenuté v testovacej časti.

counts = df3['customer\_id'].value\_counts()

df3 = df3[~df3['customer\_id'].isin(counts[counts < 5].index)]

# Tvorba odporúčaní

## Príprava dát pre odporúčania

Najväčšou prekážkou bolo vytvorenie matice, ktorá by vyzerala nasledovne:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | Produkt 1 | Produkt 2 | ............ | Produkt x |
| Customer 1 | 1 | 25 | ............ | Hodnota produktu x pre zákazníka 1 |
| Customer 2 | 26 | 0 | ............ | Hodnota produktu x pre zákazníka 2 |
| ............ | ............ | ............ | ............ | ............ |
| Customer y | Hodnota produktu 1 pre zákazníka y | Hodnota produktu 2 pre zákazníka y | ............ | Hodnota produktu x pre zákazníka y |

Obrázok 5 Zobrazenie predstavy ako by boli uložené dáta

Tvorba tejto matice bola problematická kvôli nedostatku miesta na operačnej pamäti. Riešením nakoniec bolo, vytvoriť sparse\_matrix ktorá neukladá hodnoty s obsahom 0. Keďže nie každý zákazník pristúpil ku každému produktu. Nie je potrebné tieto informácie ukladať. Teda je pre nás podstatné iba to, že ktoré produkty boli pre neho zaujímavé.

train\_row = train.customer\_id.astype(train\_customer).cat.codes

train\_col = train.product\_id.astype(train\_product).cat.codes

train\_sparse\_matrix = csr\_matrix((train["event\_type"], (train\_row, train\_col)), shape=(train\_customer.categories.size, train\_product.categories.size))

Čo sa týka odporúčania. Tu som využil algoritmus K-NN. Na začiatku som si vytvoril sparse maticu, ktorá obsahovala všetkých užívateľov a hodnoty (vzdialenosti), nakoľko sú si používatelia podobný. Teda vyzeralo to nejako takto (avšak v sparse matrix by neboli tie 0 pre ukažku ich nechám):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | Customer 1 | Customer 2 | ............ | Customer x |
| Customer 1 | 0 | 3 | ............ | Vzdialenosť zákazníka 1 pre zákazníka y |
| Customer 2 | 3 | 0 | ............ | Vzdialenosť zákazníka 2 pre zákazníka y |
| ............ | ............ | ............ | ............ | ............ |
| Customer y | Vzdialenosť zákazníka 1 pre zákazníka y | Vzdialenosť zákazníka 2 pre zákazníka y | ............ | Vzdialenosť zákazníka x pre zákazníka y |

Obrázok 6 Zobrazenie matice zákazníkov a ich vzdialeností

## Tvorba odporúčania

Na začiatku si rozdelím dataset na trénovaciu a testovaciu množinu v pomere 80:20. Keďže sú dáta veľmi veľké a k-NN trvá veľmi dlho, rozhodol som sa rozdeliť trenovaciu množinu ešte na 85 a 15. V tom prípade bolo v trénovacej množine1353 unikátnych zákazníkov. Na rozdelenie som využíval train\_test\_split. Pri odporúčaní prechádzam každého unikátneho používateľa, ktorý je v testovacej množine. Skúšam zistiť či existuje zákazník s takým ID v trenovacej množine. Ak takýto zákazník neexistuje. Pridám mu zoznam top 10 produktov. Ak existuje, nájdem 10 najpodobnejších susedov, ku ním nájdem ich produkty a všetky vložím do jedného zoznamu. Tu nastáva veľký problém, keďže ak daného zákazníka nemám v trenovacej množine a pridávam mu dáta len z top 10 produktov, toto odporúčanie sa stáva veľmi náhodné a títo zákazníci nám znižujú presnosť odporúčania.

V zozname uchovávam taktiež hodnoty produktov. Následne urobím groupby nad product\_id, čím spočítam jednotlivé hodnoty produktov, ak sú v zozname nejaké rovnaké. Následne odstránim z tohoto zoznamu všetky produkty ktorých hodnota u zákazníka, pre ktorého to odporúčam je väčšia ako 24. To je z toho dôvodu, že ak je táto hodnota 25 je veľký predpoklad že používateľ tento produkt už kúpil, avšak ak je jeho hodnota len 20. Tento produkt si určite nekúpil iba veľa krát pozeral a je dosť pravdepodobné že si ho ešte možno kúpi. Avšak do výsledného odporúčania sa dostane iba v prípade, že ostatný používatelia ho hodnotili viac ako ostatné produkty. Hodnotenie je určené pre kúpene produkty 25 a pre prezeraná 1

Po pridaní tejto funkcionality sa skóre pri testovaní zvýšilo z približne 0,02 na 0,06. Po zostavení zoznamu produktov, ktoré chceme odporúčať, sa tento zoznam oreže na 10 produktov podľa hodnoty akú mu udelili zákazníci. Ak v zozname nie je dosť produktov, doplní sa z top 10 ktoré sme si už opísali vyššie. Tieto produkty sú už výsledné odporúčania pre zákazníka.

Vstupom do funkcie na odporúčanie je list zákazníkov. V testovacej vzorke sú len unikátny zákazníci.

# Testovanie

### Kaggel

Počas tvorby odporúčaní bola nápomocná aj súťaž na stránke koggel. Rozdiel medzi jednotlivými odovzdaniami bol najme z pridaním rôznych funkcionalít a nastavovaní parametrov.

|  |  |  |
| --- | --- | --- |
| Dátum a čas | Skóre | Vylepšenia |
| 2019-11-26 01:44:49 | 0.01082 | Základný odporúčač iba 10% dát |
| 2019-11-26 18:21:28 | 0.01774 | Zväčšenie počtu trenovacích dát |
| 2019-11-28 09:15:18 | 0.01961 | Odstránenie trénovacých dát ktoré majú menej ako 2 záznamy. |
| 2019-11-29 08:16:00 | 0.02087 | Zmena limitov hodnotenia videných a kúpených |
| 2019-11-30 20:07:06 | 0.03068 | Zväčšenie počtu trénovacých dát |
| 2019-12-07 22:07:45 | 0.05381 | Pridanie možnosti kúpiť už videný produkt |

Obrázok 7 Odovzdania na súťaž Kaggel

Následne sa skúšali rôzne zmeny parametrov až výsledné skóre dňa 14.12 na Kaggel je d

### Mean average precision (mAP)

Čo sa týka testovanie, využívam mAP (Mean average precision). Vypočíta sa ako podiel súčtu priemerných presností (average precision) a počtu hľadaní. Funkcia mAP sa vyhodnocuje pre kolekciu hľadaní. Priemerná presnosť pre konkrétne hľadanie sa získa podielom súčtov presností v jednotlivých bodoch vyhľadávania a celkovým počtom relevantných dokumentov. Funkcia nadobúda hodnoty z intervalu 0 - 1 vyššia hodnota značí relevantnejšie výsledky.

def map\_score(k, Xrecommend\_data, Xtest\_data):

global test\_customer\_uniq

mapk = 0

for u in range(len(test\_customer\_uniq)):

y\_true = Xtest\_data[u]

u\_pred = Xrecommend\_data[u]

y\_pred = np.sort(u\_pred)[::-1][:k]

n\_hit = 0

precision = 0

actual = set(y\_true)

for i, p in enumerate(y\_pred, 1):

if p in actual:

n\_hit += 1

precision += n\_hit / i

mapk += precision / min(len(actual), k)

mapk /= len(test\_customer\_uniq)

return mapk

### Normalized Discounted Cumulative Gain (mDCG)

Ako ďalšiu metriku som zvolil NDCG aby som zvážil aj to ako kvalitné je odporúčanie z pohľadu postupnosti. V mojej implementácii pozerám na to či poradie v odporúčaní a v tom čo sa nachádza v testovacej zložke rovná. Ak je to tak, tak sa pridá hodnota 3 ak nie tak sa pridá iba hodnota 1.

def dcg\_at\_k(y\_true, y\_score, k = 10):

y\_dcg = np.zeros(len(y\_true))

for i in range(len(y\_true)):

for j in range(len(y\_score)):

if y\_true[i] == y\_score[j]:

if i == j:

y\_dcg[i] = 3

y\_dcg[i] = 1

gains = 2 \*\* sum(y\_dcg) - 1

discounts = np.log2(np.arange(2, gains.size + 2))

dcg = np.sum(gains / discounts)

return dcg

def ndcg\_score(Xrecommend\_data, Xtest\_data, k):

global test\_customer\_uniq

ndcg = 0.0

for u in range(len(test\_customer\_uniq)):

actual = dcg\_at\_k(sorted(Xtest\_data[u], reverse=True), Xrecommend\_data[u], k)

best = dcg\_at\_k(sorted(Xtest\_data[u], reverse=True), sorted(Xtest\_data[u], reverse=True), k)

ndcg += actual / best

avg\_ndcg = ndcg / len(test\_customer\_uniq)

return avg\_ndcg

Tabuľka nižšie zobrazuje jedno z testovaní ktoré bolo vykonané. V ňom išlo o zmenu počtu zákazníkov podľa ktorých sa tvoril zoznam odporúčaní. Teda koľko najpodobnejších zákazníkov berieme do úvahy. Veľkosť trénovacej množiny bola 1353 zákazníkov.

|  |  |  |
| --- | --- | --- |
| Počet zakazníkov | mAP | nDCG |
| 5 | 0.0558662028633961 | 0.03905552277718037 |
| 10 | 0.06356642920643127 | 0.044219790127584806 |
| 20 | 0.0691789407449493 | 0.05007445386501801 |
| 30 | 0.07053185655198824 | 0.052984721011700514 |
| 40 | 0.07370843735923104 | 0.05451764164022772 |
| 50 | 0.07472750056470547 | 0.055825160715393216 |

Pritom je jasno vidieť, že zväčšovaním počtu podobných používateľov sa zlepšuje výsledné skóre.

### Úprava dát na základe testov

Pri testovaní bolo zistené že rozhodnutie odstrániť zákazníkov ktorý majú iba 5 záznamov nebolo najvhodnejšie, dôvodom bolo zníženie výsledného skóre, avšak rozdiel medzi odstránením tých čo majú iba 2 záznamy a 5 nie je až tak veľký, no rapídne nám to ušetrý čas na výpočet keďže sa máš dataset zmenší. Keďže bol dataset pomerne veľký, rozhodol som sa odstrániť dáta staršie ako 8 dní. To nám taktiež umožnilo rýchlejšie spracovať dáta a rozdiel pri testovaní bol malí

df3['timestamp'] = pd.to\_datetime(df3['timestamp'])

indexNames = df3[df3['timestamp'].max() + pd.Timedelta(days=-8) > df3['timestamp']].index

df3.drop(indexNames , inplace=True)

Následne som odpozoroval že ak som pridal ďalšie dni, skóre na testoch sa zlepšilo. Veľmi dobré skóre som získaval pri 14 a 18 dňoch. Keďže som chcel pri odporúčaní, brať do úvahy aj to, či používateľ produkt pozrel alebo kúpil. Rozhodol som sa tieto hodnoty zameniť za čísla.

df3 = df3.replace('purchase\_item', 25)

df3 = df3.replace('view\_item', 1)

df3['event\_type'] = df3['event\_type'].astype(int)

Zaujímavé bolo sledovať ako sa mení skóre aj v závislosti od toho, ako vysoko ohodnotím kúpenie produktu oproti prezretiu. Nakoniec som stanovil prezeranie produktu na hodnotu 1 a kúpenie produktu na hodnotu 25.

Po rozdelení dát na testovaciu a trénovaciu množinu:

train, test = train\_test\_split(df3, test\_size=0.2)

som ešte raz aplikoval odstránenie používateľov ktorý majú menej prezeraní / kúpení produktu ako 5.

Pri odporúčaní treba predpokladať že prídu aj používatelia, ktorý nemajú žiaden nákup ani prezeranie produktov. Tým pádom týmto používateľom odporúčam top 10 produktov

train\_top\_10\_product = train.groupby(['product\_id']).sum().sort\_values(by=['event\_type'],ascending=False).reset\_index().head(10)

Týchto top 10 som získal ako súčet hodnôt, ktoré reprezentujú, či daný používateľ kúpil, alebo prezeral produkt. Následne som ich zoradil pomocou získaného súčtu a vybral som top 10 z nich.

# Príloha A – Tvorba odporúčaní

def find\_recommend(users\_data):

result\_recommend = pd.DataFrame(columns=['customer\_id','product\_id'])

for x in tqdm(range(len(users\_data))):

try:

index\_value = train\_customer\_uniq.index(users\_data[x])

except ValueError:

index\_value = -1

top10\_for\_user = train\_top\_10\_product

top10\_for\_user['customer\_id'] = users\_data[x]

if index\_value == -1:

result\_recommend = result\_recommend.append(top10\_for\_user)

else:

spam, user\_neighbours = user\_knn\_distance\_matrix[index\_value].nonzero()

user\_for\_recommend = train.loc[train['customer\_id'] == train\_customer\_uniq[index\_value]].iloc[:,[3,2]]

user\_for\_recommend['customer\_id'] = users\_data[x]

user\_for\_recommend = user\_for\_recommend[user\_for\_recommend.event\_type > 24] #pozor

topNuser = 10

if len(user\_neighbours) < topNuser :

topNuser = len(user\_neighbours)

for i in range(topNuser):

if i == 0:

recommend = train.loc[train['customer\_id'] == train\_customer\_uniq[user\_neighbours[i]]].iloc[:,[2,3]]

else:

recommend = recommend.append(train.loc[train['customer\_id'] == train\_customer\_uniq[user\_neighbours[i]]].iloc[:,[2,3]])

recommend = recommend.groupby(['product\_id'], as\_index=False).sum()

recommend['customer\_id'] = users\_data[x]

if 'event\_type' in recommend.columns:

recommend = recommend.sort\_values(by=['event\_type'],ascending=False)

recommend.reset\_index(drop=True, inplace=True)

recommend = recommend.append( user\_for\_recommend ).drop\_duplicates(subset='product\_id', keep=False).head(10)

r, c = recommend.shape

if r < 11:

recommend = recommend.append(top10\_for\_user).head(10)

result\_recommend = result\_recommend.append(recommend)

return result\_recommend