SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO - MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

UMJETNA INTELIGENCIJA

PREPORUKA FILMOVA BAZIRANA NA PREFERENCAMA KORISNIKA

SEMINAR

Martina Gaćina

Fran Vojković

Alen Živković

Veljača, 2020.

# Sadržaj

[1. Sadržaj 2](#_Toc33036446)

[2. Uvod 3](#_Toc33036447)

[3. Analiza i obrada podataka 4](#_Toc33036448)

[3.1. Obrada podataka 8](#_Toc33036449)

[4. Modeli 10](#_Toc33036450)

[4.1. Pearsonova R-korelacija 11](#_Toc33036451)

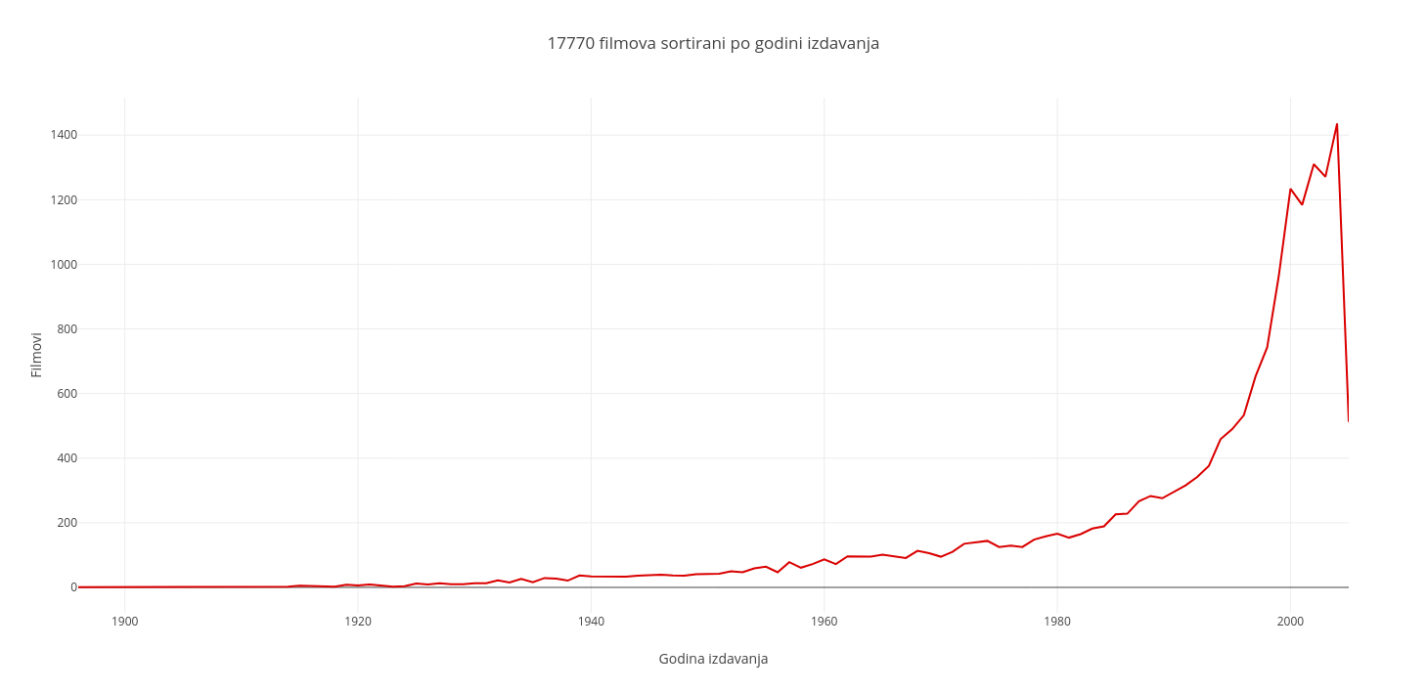
[4.2 Random Forest i XGBoost modeli 12](#_Toc33036452)

# Uvod

Proučavat ćemo modele za preporuku filmova korisniku na temelju vlastitih preferencija, odnosno na temelju korisnikovih ocjena prethodno pogledanih filmova određenog streaming servisa. Modeli se temelje na strojnom učenju iz podataka o prethodnom korisnikovom ponašanju na streaming servisu, njegovim ocjenama filmova te iz tih podataka generiramo model za preporuku sličnih filmova koji bi mogli zanimati korisnika. Uzeli smo data set koji se sastoji od naslova filma, godinom izdavanja filma te identifikacijskog broja (ID) kojeg smo pridružili svakom filmu za potrebe samog algoritma. Također, data set za inicijalizaciju modela, odnosno učenje modela ponašanju korisnika, sadrži ocjene korisnika za pojedini film. Na temelju podataka za inicijalizaciju predstavit ćemo modele za predviđanje ocjene korisnika za neocijenjene filmove. Na taj način dobijemo set filmova koji mogu poslužiti kao preporuka korisniku. Korisnici ocjenjuju filmove na skali od 1 do 5. Prvo ćemo prikazati podatke iz data seta grafički, potom proučavamo generalne preporuke filmova, ne uzimajući u obzir korisnikove preference. Time konstruiramo generalnu top listu filmova po ocjenama svih korisnika na streaming servisu. Također, analizirat ćemo model Pearsonove R-korelacije među filmovima. Na kraju predstavljamo preporuke filmova na temelju korisničkih preferencija pomoću XGBoost modela. Data setovi za model dostupni su na sljedećem [linku](https://pmfhr-my.sharepoint.com/:f:/g/personal/vofran_math_pmf_hr/EmiDTdgiIppCljn2Xxc86gcB4xZKIqASgv96JT1JO_34qQ?e=Rviz8H).

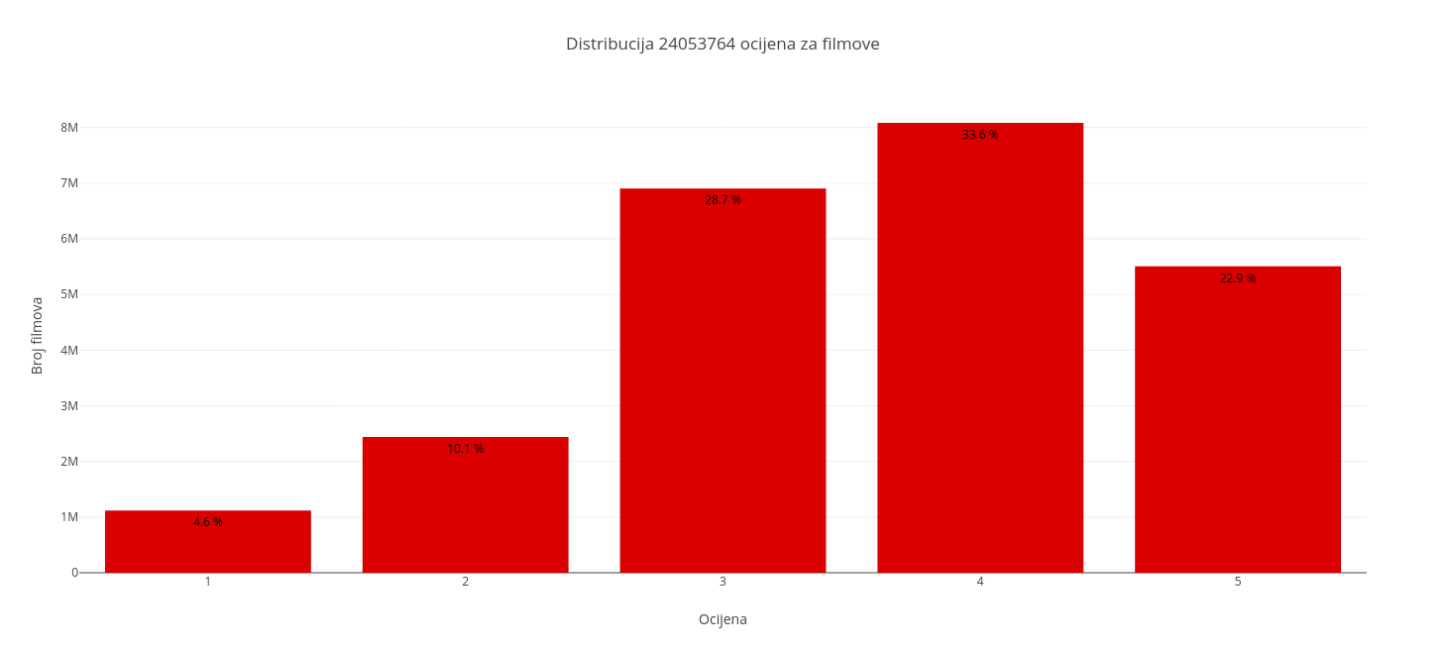
# Analiza i obrada podataka

Naš data set sadrži podatke o 17 770 filmova, njihovom datumu izdavanja te nazivu tih filmova. Također, imamo 24 053 764 ocjena korisnika. Koristimo samo jedan data set kako bi ubrzali samo izvođenje algoritma i smanjili količinu podataka. Sljedeći graf prikazuje odnos novo-izdanih filmova po godinama.

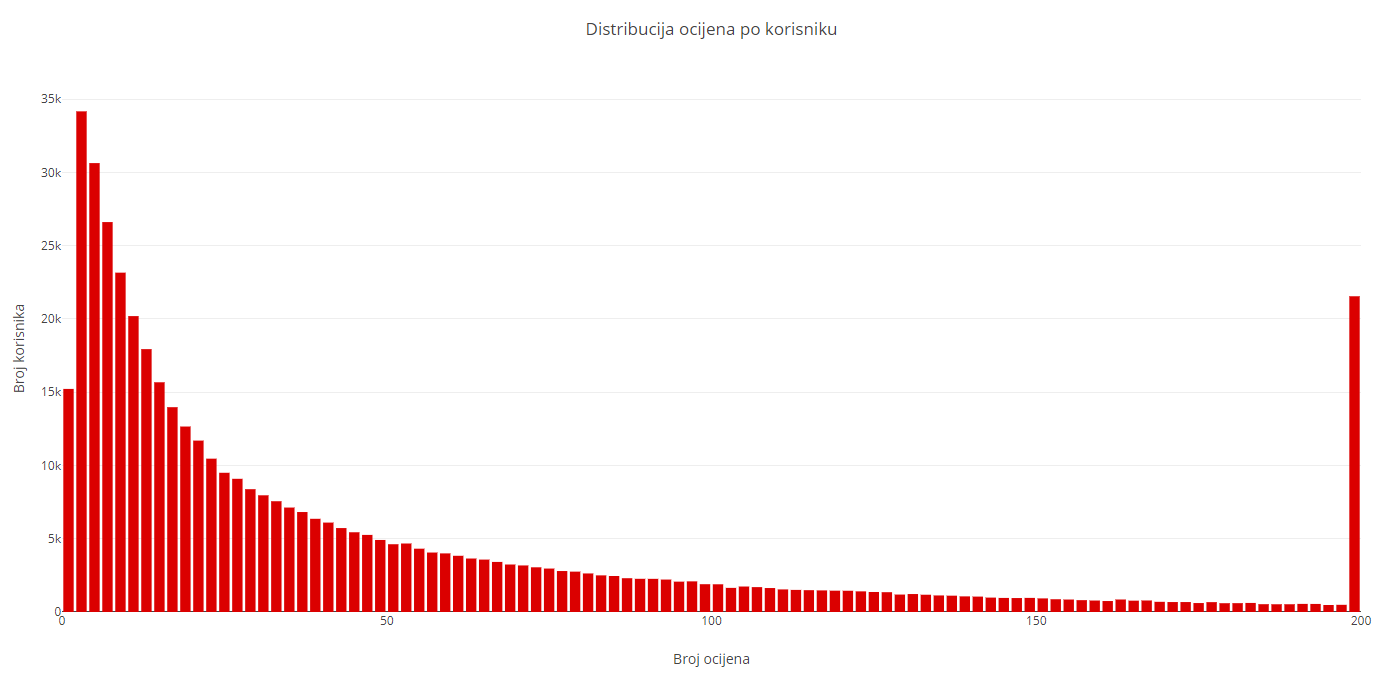


Iz navedenog vidimo da je rast novoizdanih filmova približno eksponencijalan do 2004. godine, kada broj novoizdanih filmova iznosi 1 436. Nakon 2004. godine slijedi pad, no taj pad je zanemariv. Pad dobivamo zato što smo se ograničili na samo jedan data set te zato što su podaci koje imamo nepotpuni za zadnjih par godina nakon 2004. godine.

Sljedeći graf prikazuje distribuciju ocjena po filmovima. Iz same distribucije vidimo da su ocjene pretežito veće ili jednake 3 (85.2 %), što možemo opravdati time da nezadovoljni korisnici napuštaju streaming servis. Iz tog razloga je broj „negativnih“ ocjena (ocjena manjih od 3) zanemariv, samo 14.7 %.

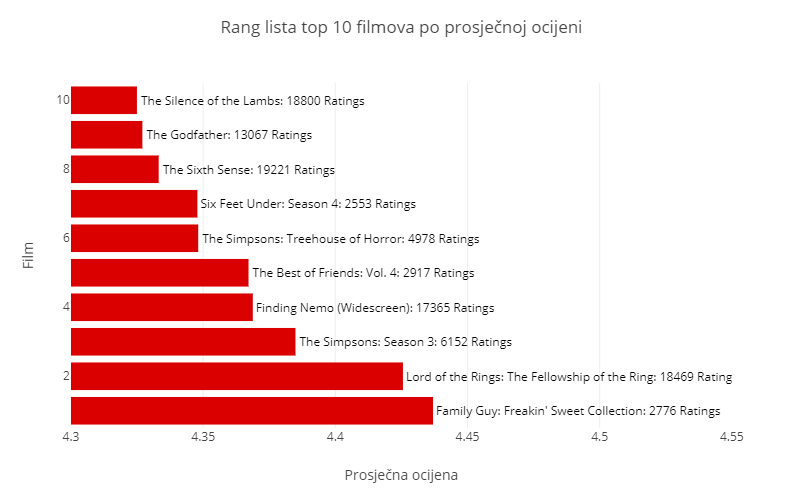


Sljedeći graf prikazuje broj ocjena po korisniku, gdje vidimo da je pad ocjena po korisniku približno eksponencijalan. Iz grafa vidimo da većina korisnika streaming servisa ocijeni do 50 filmova. U našim modelima ćemo zbog toga očistiti podatke filmova koji imaju malo ocjena te podatke korisnika koji su ocijenili manji broj filmova. Tako će naš model biti pouzdaniji.

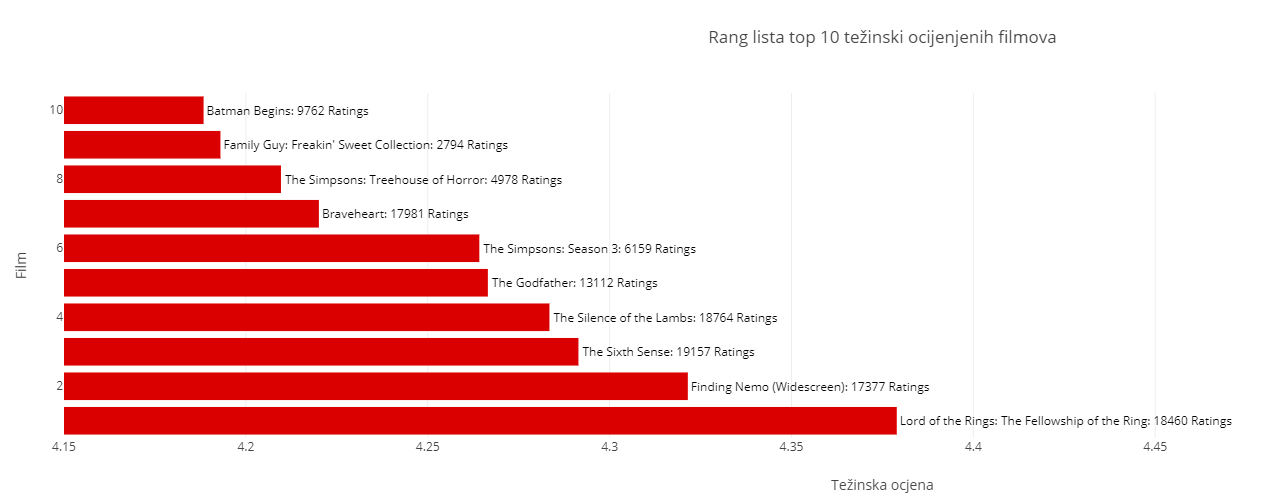


Koristit ćemo matrični prikaz filmova i korisnika. Reci u matrici su nam korisnici, a stupci su filmovi. Na -tom mjestu u matrici se nalazi ocjena korisnika za film (za one filmove koji su ocijenjeni). Za film koji korisnik nije ocijenio mjesto označimo kao prazno. Iz prikaza podataka zaključujemo da će navedena matrica biti rijetka u početku, a kasnije ćemo pokušati predvidjeti našim modelom potencijalne ocjene korisnika kako bi mogli sastaviti listu filmova koje ćemo predložiti krajnjem korisniku. Dakle, naš cilj je procijeniti prazne vrijednosti navedene matrice.

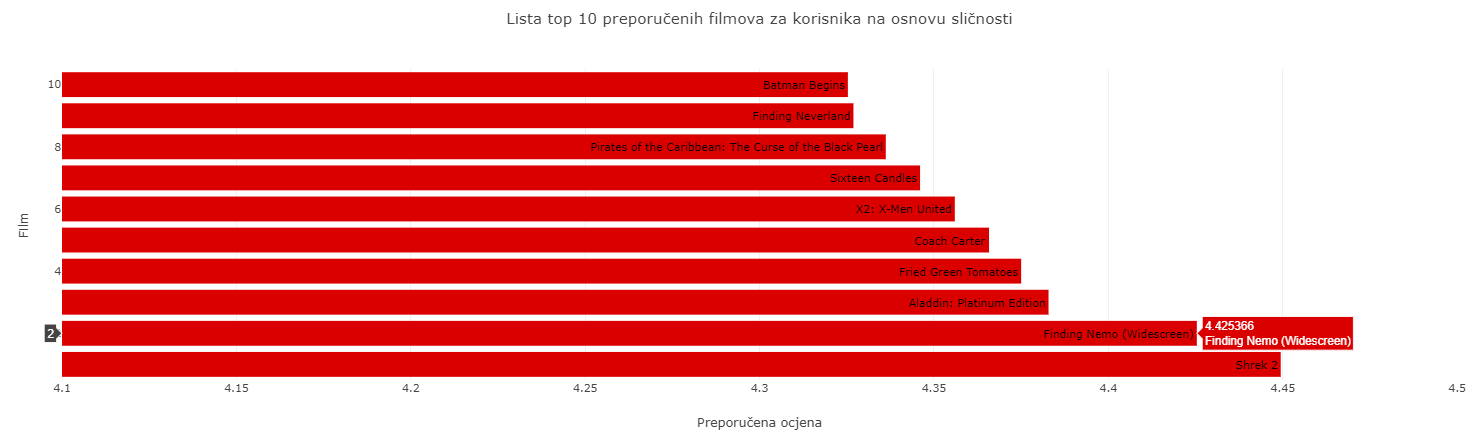
Za nove korisnike, odnosno korisnike koji imaju mali broj ocijenjenih filmova na streaming servisu, preporučiti ćemo filmove na temelju generalnog „top score-a“ filmova dobivenog iz ocjena svih korisnika te platforme. Iz tog razloga analiziramo ocjene svih filmova, poredamo ih po prosječnoj ocjeni korisnika te dobijemo sljedeću top listu.



Ovakvim ocjenjivanjem dolazimo do problema „krivih“ ocjena. Naime, ako bi netko napravio nekoliko „lažnih“ profila te ocijenio neki film ocjenom 1, ta ocjena ne bi bila valjana. Kako bi to minimizirali, uvodimo težinske ocjene kojima provjerenim izvorima (filmski kritičari i sl.) dajemo više na važnosti od korisnika s manjim brojem ocijenjenih filmova.



Ako interpretiramo svaki redak spomenute matrice kao vektor, možemo izračunati sličnost među svim korisnicima- vektorima. Ovo nam dopušta da pronađemo sve „slične“ korisnike i da radimo na preporukama za određenog korisnika. Razumno se čini preporučiti visoko ocijenjene filmove sličnih korisnika. Pošto u matrici još uvijek imamo praznih mjesta, trebamo pronaći način kako ih najpouzdanije popuniti. Jednostavan pristup je popuniti ih prosječnim vrijednostima ocjena filmova svakog korisnika. Nakon toga, ocjene „sličnih“ korisnika će biti skalirane faktorom sličnosti za svaka dva korisnika te će se izračunati njihova srednja vrijednost. Filtriranje neocijenjenih filmova nekog korisnika pokazuje najbolje rezultate. Analogno možemo napraviti i za sličnost između filmova.



## 3.1. Obrada podataka

Zbog velike količine podataka koja bi usporila naše modele, potrebno je odbaciti dio našeg dataframe-a za koji smatramo da nije previše utjecajan ili nije realan pokazatelj stanja. Logično je zato odbaciti filmove ocijenjene malen broj puta i korisnike koji su ostavili neznatan broj ocjena. U sljedećem kodu ograničili smo se na 70 % filmova s najviše ocjena, a isti kriterij iskoristit ćemo i za korisnike. Za korisnike koji nemaju dovoljan broj ocjena, ocjene predviđamo na osnovu generalnog „top score-a“.

1. f = ['count','mean']
3. df\_movie\_summary = df.groupby('Movie\_Id')['Rating'].agg(f)
4. df\_movie\_summary.index = df\_movie\_summary.index.map(int)
5. movie\_benchmark = round(df\_movie\_summary['count'].quantile(0.7),0)
6. drop\_movie\_list = df\_movie\_summary[df\_movie\_summary['count'] < movie\_benchmark].index
8. **print**('Minimalni broj ocjena filma: {}'.format(movie\_benchmark))
10. df\_cust\_summary = df.groupby('Cust\_Id')['Rating'].agg(f)
11. df\_cust\_summary.index = df\_cust\_summary.index.map(int)
12. cust\_benchmark = round(df\_cust\_summary['count'].quantile(0.7),0)
13. drop\_cust\_list = df\_cust\_summary[df\_cust\_summary['count'] < cust\_benchmark].index
15. **print**('Minimalni broj ocijena po korisniku: {}'.format(cust\_benchmark))

Minimalni broj ocjena filma: 1799.0

Minimalni broj ocjena po korisniku: 52.0

Sada primijenimo navedeni kriterij na naše podatke.

1. **print**('Početne dimenzije: {}'.format(df.shape))
2. df = df[~df['Movie\_Id'].isin(drop\_movie\_list)]
3. df = df[~df['Cust\_Id'].isin(drop\_cust\_list)]
4. **print**('Nakon obrade, dimenzije: {}'.format(df.shape))
5. **print**('-Dataset primjer-')
6. **print**(df.iloc[::5000000, :])
7. Početne dimenzije: (24053764, 3)
8. Nakon obrade, dimenzije: (17337458, 3)
9. -Dataset primjer-
10. Cust\_Id Rating Movie\_Id
11. 696 712664 5.0 3
12. 6932490 1299309 5.0 1384
13. 13860273 400155 3.0 2660
14. 20766530 466962 4.0 3923

Potrebno je još pridružiti Movie\_ID filmovima te godini njihovog izdavanja, to ćemo napraviti na sljedeći način.

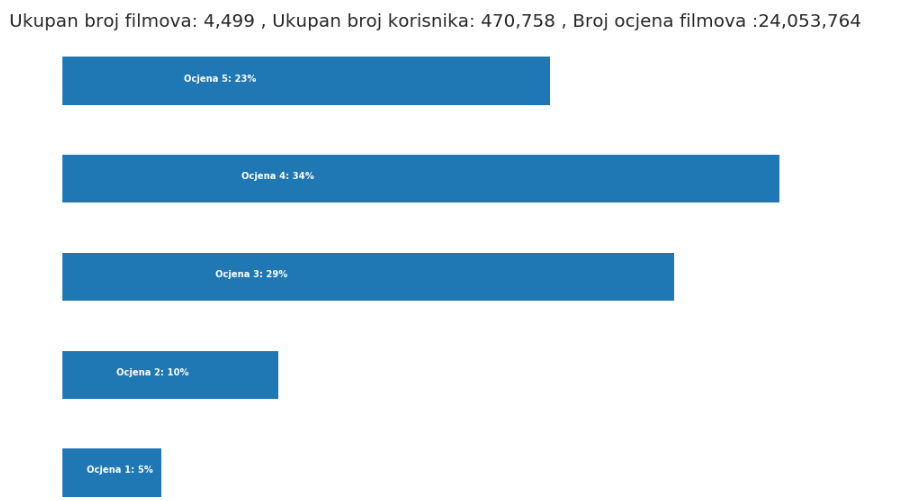
1. df\_title = pd.read\_csv('../input/movie\_titles.csv', encoding = "ISO-8859-1", header = None, names = ['Movie\_Id', 'Year', 'Name'])
2. df\_title.set\_index('Movie\_Id', inplace = True)
3. **print** (df\_title.head(10))
4. Year Name
5. Movie\_Id
6. 1 2003.0 Dinosaur Planet
7. 2 2004.0 Isle of Man TT 2004 Review
8. 3 1997.0 Character
9. 4 1994.0 Paula Abdul's Get Up & Dance
10. 5 2004.0 The Rise and Fall of ECW
11. 6 1997.0 Sick
12. 7 1992.0 8 Man
13. 8 2004.0 What the #$\*! Do We Know!?
14. 9 1991.0 Class of Nuke 'Em High 2
15. 10 2001.0 Fighter

# Modeli

Predstavit ćemo nekoliko algoritama za predviđanje korisnikova ponašanja, odnosno sastaviti ćemo personalizirane preporuke filmova za svakog korisnika. U tu svrhu koristimo strojno učenje kako bi iz dostupnih podataka predvidjeli korisnikovo ponašanje. Prvo predstavljamo Pearson-ovu R-koleraciju. Radi jednostavnosti radimo s jednim dataset-om.

1. df = pd.read\_csv('../input/combined\_data\_1.txt', header = None, names = ['Cust\_Id', 'Rating'], usecols = [0,1])
3. df['Rating'] = df['Rating'].astype(float)
5. **print**('Dataset 1 dimenzije: {}'.format(df.shape))
6. **print**('-Dataset primjer-')
7. **print**(df.iloc[::5000000, :])
8. Dataset 1 dimenzije: (24058263, 2)
9. -Dataset primjer-
10. Cust\_Id Rating
11. 0 1: NaN
12. 5000000 2560324 4.0
13. 10000000 2271935 2.0
14. 15000000 1921803 2.0
15. 20000000 1933327 3.0

Prvi data set ima sljedeći grafički prikaz, kao i što smo prethodno opisali.



## 4.1. Pearsonova R-korelacija

Koristeći Pearsonovu R-koleraciju računamo lineranu korelaciju između ocjena svih filmova te dobivamo listu od 10 filmova sa najvećom korelacijom. Ako se korisniku svidio određeni film na taj način možemo, temeljem korelacije navedenog filma sa ostalim filmovima, korisniku predložiti listu filmova koji su “slični”, odnosno koji bi mu se mogli svidjeti. Definiramo funkciju preporuke (recommend) na sljedeći način.

1. **def** recommend(naziv\_filma):
2. min\_count = 0
3. **print**("Za film ({})".format(naziv\_filma))
4. **print**("- Top 10 preporuka temeljam Pearsonove R-koleracije - ")
5. i = int(df\_title.index[df\_title['Name'] == naziv\_filma][0])
6. target = df\_p[i]
7. similar\_to\_target = df\_p.corrwith(target)
8. corr\_target = pd.DataFrame(similar\_to\_target, columns = ['PearsonR'])
9. corr\_target.dropna(inplace = True)
10. corr\_target = corr\_target.sort\_values('PearsonR', ascending = False)
11. corr\_target.index = corr\_target.index.map(int)
12. corr\_target = corr\_target.join(df\_title).join(df\_movie\_summary)[['PearsonR', 'Name', 'count', 'mean']]
13. **print**(corr\_target[corr\_target['count']>min\_count][:10].to\_string(index=False))

Kada korisnik pogleda određeni film, npr. “Batman Beyond: The Movie”, dobivamo sljedeću listu preporuka za navedenog korisnika ( za upit: recommend("Batman Beyond: The Movie") ).

Za film (Batman Beyond: The Movie)

- Top 10 preporuka temeljam Pearsonove R-korelacije -

PearsonR Name

1.000000 Batman Beyond: The Movie

0.756045 Regular Guys

0.752263 Batman Beyond: Return of the Joker

0.693876 11:14

0.685877 Morvern Callar

0.680686 Look at Me

0.652110 Sordid Lives

0.651295 Fast

0.651104 Batman: The Animated Series: Tales of the Dark...

0.649219 Batman: Mask of the Phantasm

Iz navedenog vidimo Pearsonov koeficijent korelacije u prvom stupcu, očekivano koeficjent je 1.000 za sami film “Batman Beyond: The Movie”. Također za upit recommend("Justice League")   dobivamo sljedeću listu preporuka.

Za film (Justice League)

- Top 10 preporuka temeljam Pearsonove R-koleracije -

PearsonR Name

1.000000 Justice League

0.775745 Justice League: Justice on Trial

0.658475 Regular Guys

0.654393 Testosterone

0.643469 Batman: The Animated Series: Tales of the Dark...

0.631749 Batman Beyond: The Movie

0.614198 Batman Beyond: Return of the Joker

0.577765 Batman: Mask of the Phantasm

0.567673 Nine Dead Gay Guys

0.567410 Yi Yi

## 4.2 Random Forest i XGBoost modeli

Sada ćemo iskoristiti RandomForestRegressor i XGBRegressor kako bi napravili modele koji će određenom korisniku preporučiti filmove ako posjedujemo listu njegovih već pogledanih filmova zajedno s ocjenama koje im je pripisao. Cilj je predvidjeti koje bi od nepogledanih filmova korisnik ocijenio najvišom ocjenom te mu onda iste preporučiti.

Budući da dataset koji koristimo ne sadrži previše informacija o svakom filmu, samo njegov naslov, godinu i ID u sustavu, moramo modele bazirati na ID-u i godini (ovdje treba primijetiti da koristim općeniti izraz 'godina' jer za neke filmove to znači godina izlaska u kino dvorane, dok za druge predstavlja dolazak na platformu ili izlazak DVD-a na tržište). Mogli smo ovdje koristiti i datum korisnikovog gledanja filma, ali taj stupac sadrži previše kategorijalnih varijabli da bi se efikasno primijenio One-Hot Encoder (dataframe postaje previše masivan). Također, svaki pokušaj label encodinga ne uspijeva jer se u testnim setovima javljaju datumi neprisutni u trening setovima pa nam tako ne ostaje drugo nego sasvim izbaciti 'Date' stupac.

Dakle predviđanja će se uglavnom bazirati na godini izlaska filma, dok i predviđanje s obzirom na ID može biti korisno ako se pretpostavi da streaming služba dodaje nove filmove u hrpama s određenim značajkama, recimo da odjednom doda pet akcijskih filmova ili tri filma strave. Tada se ID može gotovo shvatiti kao žanr.

Do ocjena korisnika, u ovom slučaju njih 785314, lako je doći iz našeg df dataframea. Zatim izbacimo 'Cust\_Id', jer ga već znamo i svugdje je jednak, te iz df\_title pridružimo godinu svakom ID-u. Pomoću train\_test\_split stvorimo naše trening i test setove. Ove setove koristimo za kreiranje naših Random Forest i XGBoost modela.

1. df\_785314 = df[(df['Cust\_Id'] == 785314)]
2. df\_785314=df\_785314.drop('Cust\_Id', axis=1)
3. year=df\_title.copy()
4. year=year.reset\_index()
5. year = year[~year['Movie\_Id'].isin(drop\_movie\_list)]
6. year = year[year['Movie\_Id'].isin(df\_785314.Movie\_Id)]
7. year = year[year['Movie\_Id'].isin(df.Movie\_Id)]
8. col = np.asarray(year['Year'])
9. df\_785314['Year']=col
11. y=df\_785314.Rating
12. x=df\_785314.drop('Rating', axis=1)
13. x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2, random\_state=2)
15. model=RandomForestRegressor(criterion='mae',random\_state=0)
16. model.fit(x\_train, y\_train)
18. my\_model=XGBRegressor(n\_estimators=500, learning\_rate=0.05, random\_state=0)
19. my\_model.fit(x\_train, y\_train, early\_stopping\_rounds=5, eval\_set=[(x\_test, y\_test)], verbose=False)

Sada moramo doći do filmova koje korisnik 785314 nije ocijenio (dakle niti pogledao). Kopiramo dataframe df\_title u kojem se nalaze podaci o filmovima pa ga prvo očistimo svih filmova koji nisu u našem df-u kao i onih koje smo odlučili izbaciti zbog premalo ocjena. Sada izbacujemo sve filmove koje je korisnik 785314 već pogledao te napokon dobivamo popis svih filmova koje možemo preporučiti. Ostaje nam izbrisati stupac s naslovom i provesti predviđanja koja onda spremamo u posebne stupce.

1. user\_785314=df\_title.copy()
2. user\_785314 = user\_785314.reset\_index()
3. user\_785314 = user\_785314[~user\_785314['Movie\_Id'].isin(drop\_movie\_list)]
4. user\_785314 = user\_785314[~user\_785314['Movie\_Id'].isin(df\_785314.Movie\_Id)]
5. user\_785314 = user\_785314[user\_785314['Movie\_Id'].isin(df.Movie\_Id)]
6. user\_=user\_785314.drop('Name', axis=1)
8. user\_785314['Estimate\_Score'] = model.predict(user\_)
9. user\_785314['XGB\_Estimate'] = my\_model.predict(user\_)

Preostaje nam poredati filmove silazno po predviđenim ocjenama. Time smo dobili najbolje preporuke za korisnika 785314.

1. user\_785314 = user\_785314.drop('Movie\_Id', axis = 1)
3. user\_785314 = user\_785314.sort\_values('Estimate\_Score', ascending=False)
4. **print**(user\_785314.head(10))

Na kraju za našeg korisnika 785314 dobivamo preporuke:

Year Name Estimate\_Score \

392 1998.0 The Replacement Killers 5.0

329 1998.0 Wild Things 5.0

359 1999.0 Lies 5.0

344 1998.0 Star Trek: Voyager: Season 5 5.0

44 1999.0 The Love Letter 4.9

288 1998.0 The Avengers 4.9

303 1997.0 King of the Hill: Season 1 4.9

284 1997.0 The Devil's Own 4.9

361 2000.0 The Flintstones in Viva Rock Vegas 4.8

315 1999.0 Futurama: Monster Robot Maniac Fun Collection 4.8

XGB\_Estimate

392 3.794509

329 3.794509

359 3.794509

344 3.794509

44 3.794509

288 3.794509

303 3.794509

284 3.794509

361 3.572643

315 3.794509