VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

JEDNODUCHÝ DOPORUČOVACÍ SYSTÉM

SIMPLE RECOMMENDATION SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **DAMIÁN GORČÁK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Gorčák Damián**
Program: Informační technologie
Název: **Jednoduchý doporučovací systém**
Simple Recommender System
Kategorie: Data mining

Zadání:

1. Seznamte se s problematikou doporučovacích systémů a prostudujte jejich jednotlivé typy.
2. Vyberte vhodnou aplikační doménu a vyhledejte vhodná data, která lze využít pro doporučování, a zvolte pro tuto doménu vhodný typ doporučovacího systému.
3. Po dohodě s vedoucím navrhněte aplikaci, která bude provádět doporučování zvolenou metodou.
4. Implementujte navrženou aplikaci a experimentálně ověřte její funkčnost.
5. Zhodnoťte použitelnost výsledné aplikace a dosažených výsledků. Diskutujte možná rozšíření tohoto projektu.

Literatura:

- Aggarwal, C.: Recommender Systems: The Textbook 1st ed. Springer Verlag, 2016. ISBN 978-3319296579.
- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 20. října 2021

Abstrakt

Odporúčacie systémy sú veľmi dôležité pri vyhľadávaní rôznych položiek naprieč internetom. Existuje mnoho algoritmov pre vytváranie odporučení. Cieľom tejto práce bolo nájsť vhodné dátové sady a vytvoriť aplikáciu, ktorá ich dokáže spracovať. Následne porovnať dátové sady spolu s vybranými algoritmami odporúčacích systémov.

Abstract

Recommender systems are very important in searching for items all over the internet. There are many algorithms for creating recommendations. The main goal of this thesis was to find suitable datasets and make application, which would process them. After that, chosen algorithms for recommender systems are compared with selected datasets

Kľúčové slová

odporúčiace systémy, svd, als, sgd, dátová sada, predikcia, strojové učenie, interakčná matica, python

Keywords

recommender systems, svd, als, sgd, dataset, prediction, machine learning, interaction matrix, python

Citácia

GORČÁK, Damián. *Jednoduchý doporučovací systém.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Jednoduchý doporučovací systém

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Vladimíra Bartíka, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Damián Gorčák
2. mája 2022

Poďakovanie

Ďakujem Ing. Vladimírovi Bartíkovi, Ph.D. za odbornú pomoc, poskytnuté rady a za vedenie pri vypracovávaní bakalárskej práce.

Obsah

Kapitola 1

Úvod

Počítačová veda, internet a veci, ktoré sú s týmito pojmami späté, sa stávajú neoddeliteľnou súčasťou nášho života. Vďaka internetu sú nám informácie čoraz viac dostupnejšie. S týmto sa spája aj problém nájdenia dát pre konkrétneho užívateľa. Preto informačné systémy, internetové obchody a mnoho dalších odvetví, čím ďalej tým viac začínajú využívať odporúčacie systémy. Odporúčacie systémy sú algoritmy, ktoré pracujú s veľkým množstvom dát (zväčša o užívateľoch a produktoch) a snažia sa odporučiť užívateľovi produkt, ktorý by mu vyhovoval. V niektorých odvetviach hrajú odporúčacie systémy veľmi významnú úlohu. Ako príklad uveďme firmu Netflix, ktorá organizovala výzvu, pričom jej cieľom bolo vytvoriť algoritmus pre odporúčací systém tak, aby bol efektívnejší ako ich vlastný. Hlavnou finančnou odmenou bolo jeden milión dolárov.

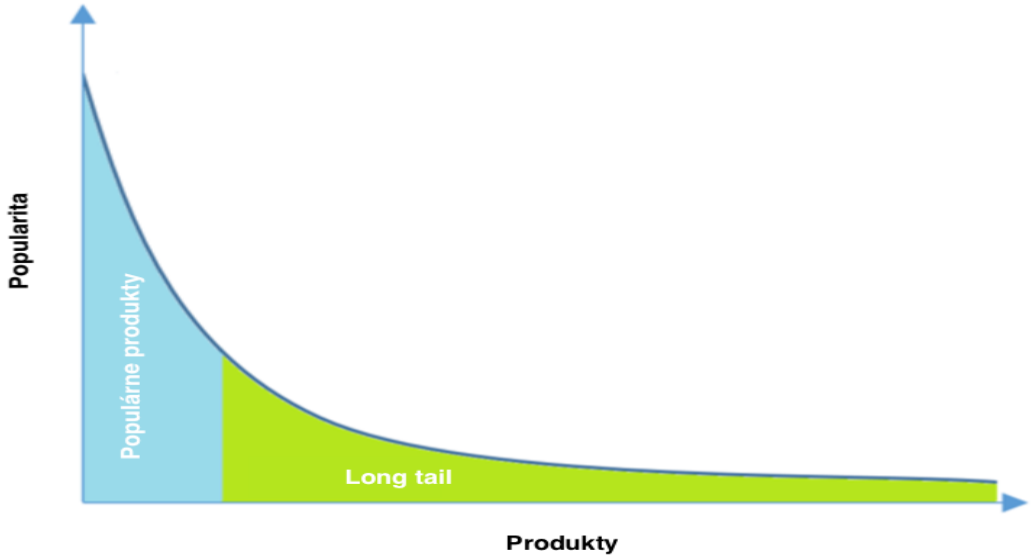
Táto bakalárska práca sa venuje metódam pre vytvorenie odporúčacích systémov. Konkrétne algoritmom kolaboratívneho filtrovania, ktoré vyžadujú model. Sú to často používané metódy, veľmi efektívne pre použitie. Cieľom práce je vytvoriť aplikáciu, ktorá dokáže spracovať dodanú dátovú sadu, vytvoriť k nej potrebné štatistiky a umožniť užívateľovi ohodnotiť položky. S týmito hodnoteniami môže užívateľ vytvárať modely niektorých z algoritmov pre odporúčacie systémy a porovnávať ich pomocou štatistík, ktoré mu vytvorená aplikácia poskytne.

V kapitole 2 sú zhrnuté prístupy odporúčacích systémov. Ďalej sú popísané jednotlivé problémy, ktoré nastávajú pri zavedení týchto systémov do praxe. Následne metriky, ktorými dokážeme merať a porovnávať ich presnosť a tiež možné prístupy k rozdeleniu dátovej sady na trénovaciu a testovaciu. Nakoniec je zosumarizovaná problematika optimalizácie hyperparametrov. Nasledujúca kapitola 3 stručne popisuje existujúce implementácie jednotlivých algoritmov v jazyku Python. Implementácia a návrh výslednej aplikácie sú podrobnejšie popísané v kapitole 4. Obsahom kapitoly 5 je výber dátových sád a experimentovanie s nimi na konkrétnych algoritmoch pomocou vytvorenej aplikácie.

Kapitola 2

Odporúčacie systémy

„Odporúčacie systémy sú systémy, ktoré sú navrhnuté tak, aby používateľovi odporúčali veci na základe mnohých faktorov. Tieto systémy predpovedajú najpravdepodobnejší produkt, ktorý si užívatelia s najväčšou pravdepodobnosťou kúpia a majú oň záujem.“ [27] Tu sa dostávame k dileme Long-tail popularite položiek. obrázok 2.1 zobrazuje tento problém.



Obr. 2.1: Grafické znázornenie Long tail problému. Prevzaté z [28]

Na obrázku 2.1 vidíme grafické znázornenie problému. Na osi y sa nachádza popularita a na osi x prvky. Z grafu je vidieť, že populárnych prvkov je menej než tých nepopulárnych. A práve úlohou odporúčacích systémov je posunúť hranicu medzi týmito dvoma skupinami produktov tým, že užívateľovi odporučia menej obľúbený prvok, ktorý je v nejakom smere buď podobný k jednotlivému populárnemu prvku, alebo patrí do skupiny prvkov, ktoré má užívateľ v obľube. Posunutím hranice tiež prispejeme k diverzifikácií prvkov (jednoduché systémy sú navrhnuté tak, aby odporúčali iba populárne prvky). Na základe tohto možno pozorovať, aké užitočné odporúčacie systémy sú.[28]

2.1 Delenie odporúčacích systémov

Podľa [10] sa odporúčacie systémy delia na: demograficky založené odporúčacie systémy, systémy založené na vedomosti, systémy založené na obsahu a Kolaboratívne filtrovanie.

2.1.1 Demograficky založené odporúčacie systémy

Demograficky založené odporúčacie systémy vytvárajú obraz o užívateľovi z jeho demografických informácií (vek, pohlavie, plat, krajina narodenia atď.). Na základe týchto informácií sa vytvoria skupiny užívateľov s podobnými demografickými poznatkami. Pri registrácií nového užívateľa a získaní daných informácií sa užívateľ, na základe jeho demografie, zaradí do príslušnej skupiny. Následne sú mu odporúčané položky patriace do tejto skupiny. Na rozdiel od kolaboratívnich a obsahovo založených systémov nemusia mať demograficky založené odporúčacie systémy žiadnu históriu interakcií medzi užívateľom a položkami, t.j netrpia studeným štartom 2.2. [7]

2.1.2 Systémy založené na vedomosti

Tieto systémy sa používajú pri užívateľoch a položkách, medzi ktorými nedochádza k častej interakcií, napríklad kúpa domu, hľadanie práce atď. Znamená to, že systém nemá žiadnu históriu vzájomných stretov, pretože užívateľ vyhľadáva danú skupinu produktov len zdanlivo. Preto na začiatku tieto systémy vyžadujú od užívateľa vyplniť vstupný formulár, v ktorom sa definuje daná položka ako napríklad typ práce, mesto v ktorom sa práca nachádza. Na základe týchto informácií sa algoritmus snaží vytvoriť tzv. „Domain Knowleage“. Táto vedomosť je knižnica (súbor) pravidiel, ktoré vytvoríme z každej získanej informácie. Z jednotlivých pravidiel sa vytvorí jeden dotaz na databázu, ktorý sa využíva na získanie položiek pre daného užívateľa. Ten taktiež môže zlepšovať výber tým, že ho zúži, alebo vyberie najlepšiu položku, na základe ktorej sa dotaz upraví. Podobne ako pri demograficky založených odporúčacích systémoch netrpia tieto metódy studeným štartom 2.2 .[15]

2.1.3 Systémy založené na obsahu (Content based)

Systémy založené na obsahu vyžadujú informácie buď o produkte (žáner, autor, herci), vtedy hovoríme o užívateľsko-orientovanom modeli, alebo o užívateľovi (vek, pohlavie, krajina), ktorý nazývame položkovo-orientovaný model.Ich cieľom je snaha vytvoriť model, ktorý predpovedá na základe dostupných informácií možné interakcie medzi užívateľmi a položkami[20]. Výhodou týchto systémov je, že si vedia poradiť s problémami, ktoré majú metódy kolaboratívneho filtrovania. Dokážu predikovať odporučenia položkám, dokonca aj takým, ktoré ešte nik neohodnotil. Ak aj konkrétny užívateľ neohodnotil žiadnu položku, nie je problém mu nejaké odporučiť. Ďalšou výhodou je, že systémy založené na obsahu sa dokážu vysporiadať so zmenou preferencií užívateľa v krátkom čase. Nevýhodou spomínaných metód je, že sú závislé na metadátach o položkách. To znamená, že vyžadujú atribúty položiek a niekedy aj atribúty užívateľov pre to, aby mohli vykonať odporučenie. [17]

Položkovo orientovaný (item centred) model

Je veľmi podobný s demograficky založenými systémami. Pri týchto modeloch si kladieme otázku „Koľkým užívateľom sa bude páčiť táto položka?“. Odpoveď sa zisťuje na základe atribútov, ktoré poznáme o užívateľoch. Napríklad vieme, že ženám sa viac páčia romantické

filmy a naopak mužom sú bližšie sci-fi filmy. Avšak tento model nie je príliš presný vo svojich odhadoch, pretože síce užívatelia môžu mať rovnaké atribúty, no nemusia mať rovnaké preferencie.[20]

Užívateľsky orientovaný (user centred) model

Užívateľsky orientovaný model pracuje s atribútmi, ktoré sú o produktoch známe. Na základe interakcie užívateľa s určitým produktom (či už zakúpenie alebo kladné ohodnotenie) sa vyhľadajú a odporúčia produkty, ktoré sú čo najpodobnejšie danému produktu. Na zistenie podobnosti sa používajú metriky na meranie vzdialenosti medzi produktami. Tieto metriky sú spomínané v 2.1.3. User centred model nepotrebuje dáta od žiadneho iného používateľa a je špecifický pre konkrétneho používateľa. Výhodou je, že model trpí výrazne menej studeným štartom než kolaboratívne filtrovanie. Avšak, ak dáta nie sú určené spoľahlivo, môže dôjsť k nesprávnym odporúčaniam.[20]

Algoritmi pre nájdenie podobnosti

Sú to jednoduché algoritmy, ktoré nachádzaju najbližšie body k bodu, pre ktorý ich hľadáme. V prípade odporúčacích systémov sa takto hľadá podobnosť prvkov, na základe ich vektorov. Stručne povedané, pomocou algoritmov pre nájdenie podobnosti, niektoré typy odporúčacích systémov dokážu nájsť podobné položky.

Jackardov index Jackardov index počíta podobnosť dvoch množín. Výsledok je desatinné číslo v rozmädzí od 0 po 1. V [13] je Jackardov index vyjadrený ako:

$$J(A,B) = |A \cap B|/|A \cup B|$$

Ak sa výsledok, po dosadení do tohto vzorca rovná 1 znamená to, že množiny sú identické (ich prvky sú rovnaké), 0 znamená, že sa tam žiadna podobnosť nenachádza. Metóda sa využíva, keď interakčná matica obsahuje binárne hodnoty (napr. iba reflektuje či si užívateľ zakúpil produkt). Keď obsahuje kompexnejšie dáta (napr. hodnotenia), tak je výsledok nepresný.[13][22]

Kosínusová vzdialenosť Meria vzdialenosť medzi vektormi užívateľov/položiek, pomocou kosínusu z uhla medzi nimi. V odporúčacích systémoch sa musia prázdne hodnoty nahradiť nulami vo vektoroch. V [6] je spomínaný vzorec pre nájdenie kosínusovej vzdialenosti:

$$\cos(\alpha) = \frac{A + B}{|A| * |B|}$$

Kde A a B sú vektory, ktoré porovnávame. Na rozdiel od Jackardoveho indexu sa tu menší výsledok rovná menšej vzdialenosti.

Euklidovská vzdialenosť V tejto metóde sa počíta vzdialenosť pomocou pytagorovej vety (Obsah štvorca zostrojeného nad preponou (najdlhšou stranou) pravouhlého trojuholníka je rovný súčtu obsahov štvorcov zostrojených nad jeho odvesnami.). Vzdialenosť je tu daná ako priama čiara medzi bodmi, ktorých vzdialenosť počítame. V [8] je ukázaný vzorec pre výpočet euklidovskej vzdialenosti v n-dimenzionálnom priestore:

$$\text{vzdialenosť(A,B)} = \sqrt{\sum_{i=1}^{N}(A_i - B_i)}$$

Euklidovská vzdialenosť je pomerne často využívaná, dokonca v niektorých algoritmoch je predvolená. Výhodou sú všeobecne podávané dobré výsledky a spoľahlivosť. [8]

Pearsonov korelačný koeficient Výsledkom tejto metriky je číslo v rozmädzí -1 a 1, kde -1 znamená silne negatívnu koreláciu, 0 znamená žiadnu koreláciu a 1 znamená veľmi silnú koreláciu. V [9] sa spomína tento vzorec, ktorý počíta podobnosť medzi užívateľom U a užívateľom V:

$$\text{PCC(U,V)} = \frac{\sum_{i \varepsilon I_{uv}} (r_{ui-\overline{r_u}})(r_{vi-\overline{r_v}})}{\sqrt{\sum_{i \varepsilon I_{uv}} (r_{vi-\overline{r_v}})^2} \sqrt{\sum_{i \varepsilon I_{uv}} (r_{ui-\overline{r_u}})^2}}$$

Kde I_{uv} predstavuje pole položiek, ktoré sú spoločne hodnotené užívateľmi U a V. $\overline{r_u}$ a $\overline{r_v}$ predstavujú priemerné hodnotenie užívateľa U alebo V na položku i v I_{uv}. Hodnotenia daných užívateľov na tú istú položku i predstavujú vo vzorci premenné r_{vi} a r_{ui} . Taktiež v [9] sa spomína vzorec pre výpočet vzdialenosti dvoch položiek:

$$\text{PCC(i,j)} = \frac{\sum_{u \varepsilon U_{ij}} (r_{ui-\overline{r_i}})(r_{vj-\overline{r_j}})}{\sqrt{\sum_{u \varepsilon U_{ij}} (r_{ui-\overline{r_i}})^2} \sqrt{\sum_{u \varepsilon U_{ij}} (r_{uj-\overline{r_j}})^2}}$$

Kde U_{ij} predstavuje pole užívateľov, ktorí spoločne hodnotili položky i a j. $\overline{r_i}$ a $\overline{r_j}$ predstavuje priemerné hodnotenia na položky i a j v U_{ij}. Hodnotenia užívateľa na položky i a j sú premenné r_{ui} a r_{uj}. [9]

2.1.4 Kolaboratívne filtrovanie

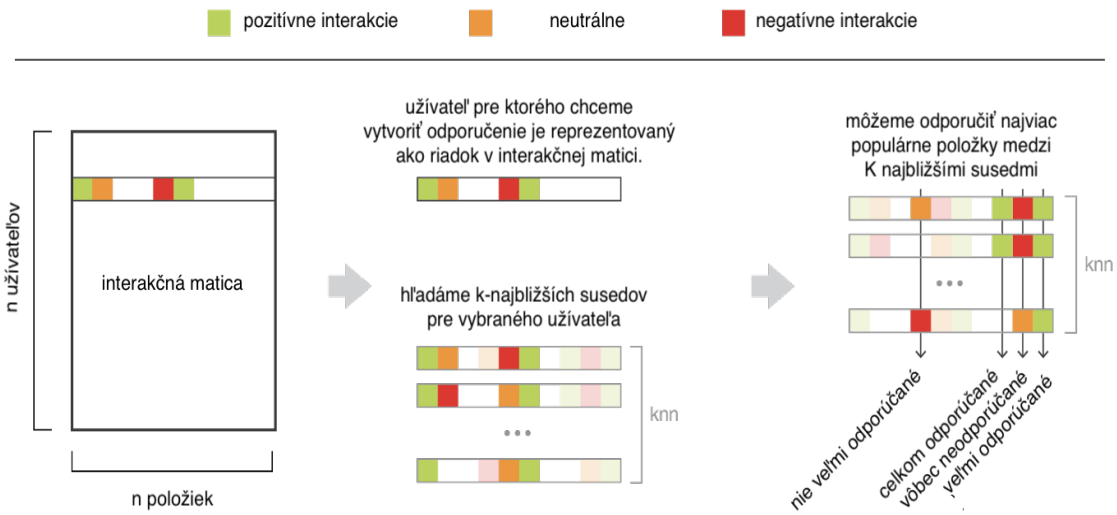
Kolaboratívne filtrovanie je metóda, ktorá je najviac používaná. Pracuje výlučne s minulými vzájomnými pôsobeniami medzi užívateľmi a položkami, ktoré sú zaznamenané v interakčnej matici. Tiež nevyžaduje žiadne dodatočné informácie o užívateľoch ani o položkách a vyhľadáva možné vzťahy medzi týmito položkami. Toto je výhodou oproti obsahovo založeným metódam, pretože užívateľ nemusí zadávať žiadne osobné informácie, stačia iba interakcie s položkami, vedie to k istej miere zachovania súkromia. Kolaboratívne filtrovanie sa rozdeľuje do dvoch ďalších metód a to: metóda založená na modeli a metóda založená na pamäti.

Metóda založená na pamäti

Algoritmy pre túto metódu pracujú s veľkými maticami, ktoré reprezentujú interakcie medzi užívateľmi a položkami. To znamená, že používajú iba dáta z interakčnej matice. Z týchto dát sa snaží nájsť určitú podobnosť či už medzi hodnoteniami viacerích užívateľov, alebo podobne hodnotených položiek k položke hodnotenej práve jedným užívateľom.

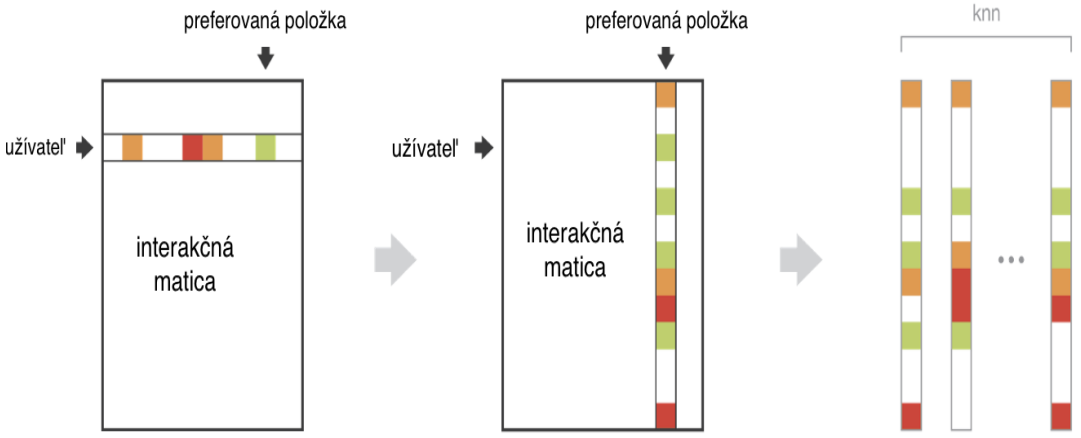
Užívateľ-užívateľ V tomto spôsobe odporúčania sa pracuje s vektorom každého užívateľa, v ktorom sú zaznamenané interakcie s rôznymi položkami (ohodnotenie položky). Medzi týmito užívateľmi sa snažíme nájsť podobnosť s využitím techník pre nájdenie najkratšej vzdialenosti 2.1.3. Po nájdení podobnosti sa ku konkrétnemu užívateľovi ponechá iba K-najbližších susedov a odporúčia sa najpopulárnejšie prvky medzi nimi.

Položka-položka Metóda slúži na hľadanie položiek, ktoré sú podobné k položke ohodnotenej užívateľom kladne. Podobnosť sa v tomto prípade neberie ako to, či sú dve položky



Obr. 2.2: Grafické znázornenie techniky užívateľ-užívateľ. Prevzaté a upravené z[20]

rovnaké na základe ich vlastností. Vyberá sa podľa toho, či iný užívateľ neohodnotil danú položku rovnako ako referenčný užívateľ, berie ohľad aj na podobné položky, ktoré hodnotil rovnako. V prípade ak sa také vyskytnú, tvrdíme, že tieto dve položky sú rovnaké. Po nájdení podobných položiek si ponecháme K-najbližších susedov.



Obr. 2.3: Grafické znázornenie techniky položka-položka. Prevzaté a upravené z[20]

V prvej časti obrázku 2.3 je vidieť, preferovanú položku užívateľa, pre ktorého sa vytvára odporúčanie. V druhej časti je znázornené, že každá položka je reprezentovaná vlastným stĺpcom v interakčnej matici. V tretej časti sa dá pozorovať k-najbližších susedov k preferovanej položke.

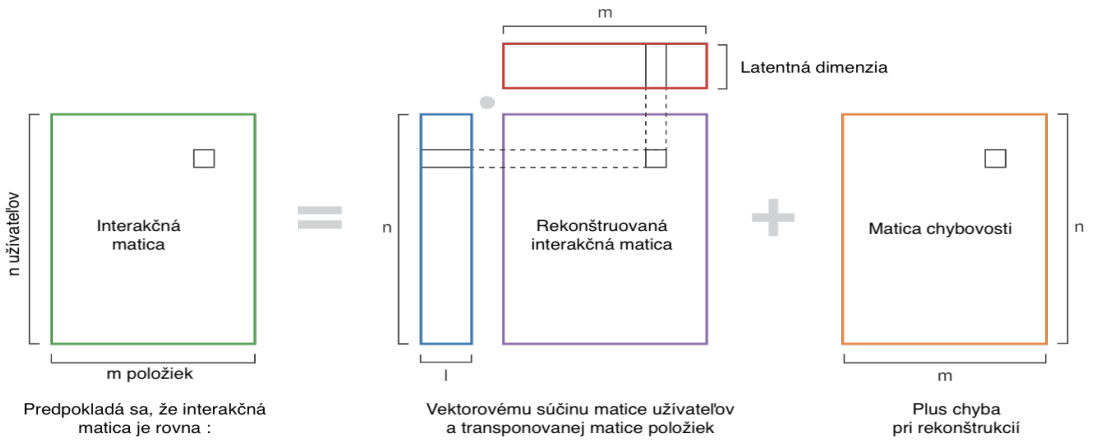
Metóda, ktorá vyžaduje model

Metóda, ktorá vyžaduje model vyžaduje iba informácie z interakčnej matice a snaží sa pomocou Faktorizácia matíc nájsť vzťahy medzi hodnoteniami.

Faktorizácia matíc Tento algoritmus sa snaží rozdeliť veľmi riedku interakčnú maticu na menšie matice. Jednu pre užívateľa a druhú pre položky – výsledkom skalárneho súčinu týchto dvoch matíc musí byť matica, na základe ktorej boli tieto 2 matice vytvorené . Nech M je matica interakcií. Potom chceme túto maticu rozdeliť na :

$$M \approx U \cdot I^T$$

Pričom U je matica užívateľov a I je matica položiek. Tieto matice môžu byť doplnené buď ručne a to tak, že spoločnosť má najatých ľudí, ktorí zadávajú položkám vlastnosti (napr. pri filme, vlastnosti ako dramatickosť, humor ...), alebo pomocou strojového učenia, kde systém vyplní maticu za nás, takzvanými skrytými vlastnosťami, „keďže sú naučené a nie dané, síce majú matematický zmysel, ale žiadnu intuitívnu interpretáciu (človeku je ťažké až nemožné tomu porozumieť).“ [20]



Obr. 2.4: Grafické znázornenie faktorizácie matíc. Prevzaté a upravené z[20]

Rozloženie podľa singulárnych hodnôt (Singular value decomposition (SVD)) Je metóda, ktorá je jednou z najpoužívanejších pre odporúčacie systémy. Víťazný tím súťaže organizovanou firmou Netflix, spomínanou v úvode, použil ako svoj výherný algoritmus práve SVD. Tento algoritmus sa vo všeobecnosti používa na zmenšovanie dimenzií v strojovom učení. V odporúčacích systémoch sa táto metóda využíva na faktorizáciu matíc 2.1.4. SVD sa snaží rozložiť veľkú interakčnú maticu na vektorový súčin troch menších matíc:

$$M_{[m \times n]} = U_{[m \times r]} \Sigma_{[r \times r]} (V_{[n \times r]})^T$$

kde:

- **U** je ortogonálna ľavá singulárna matica, **m** je počet užívateľov v interakčnej matici M a **r** je jej hodnosť. Táto matica udáva vzťah medzi užívateľmi a latentnými faktormi.

- **Σ** sú singulárne hodnoty, je to diagonálna matica (nenulové hodnoty má iba na diagonále). **Σ** opisuje silu nájdených latentných faktorov, r je hodnosť interakčnej matici M
- **V** sú pravé singulárne vektory, **n** je počet položiek v interakčnej matici M a **r**je jej hodnosť. Táto matica udáva podobnosť medzi položkami a latentnými faktormi.

Nevýhodou tejto metódy je, že oproti ALS a SGD je značne pomalá, no efektívnejšia.[26]

Stochastické gradientné klesanie (Stochastic gradient descent (SGD)) Je metóda faktorizácie matíc, ktorá iteratívne znižuje RMSE(root mean square error). Cieľom tejto metódy je iteratívne nájsť také matice užívateľov a položiek, kde RMSE bude čo najmenšie. V každej iterácií používame metódu gradient descent 2.5 . Iterujeme až do vtedy, dokiaľ chyba nedosiahne optimálnu hodnotu. Dá sa to vyjadriť vzorcom, použitým z [20]:

$$(U, I) = \arg\min_{U,I}(\sum_{i,j}((U_i)(I_j)^T - M_{ij})$$

Avšak pri tejto metóde može dôjsť k problémom overfitting alebo underfitting, pre riedkosť dát v interakčnej matici. Overfitting nastáva vtedy, keď sa model príliš naviaže na dáta, ktoré máme k dispozícií. Ak pribudne v matici nejaký nový prvok, vypočítaná hodnota nebude vyhovujúca, pretože matice sú príliš naviazané na predošlé dáta. Underfitting vzniká pri neschopností systému nájsť v interakčnej matici súvislosti medzi dátami. Pre vyhnutie sa tejto chybe existuje doplnok, takzvaná regularizácia, taktiež spomenutá v [20], k predošlému vzorcu:

$$(U, I) = \arg\min_{U,I} \frac{1}{2}(\sum_{i,j}((U_i)(I_j)^T - M_{ij}) + \frac{\alpha}{2}(\sum_{i,k}(U_{i,k})^2 + \sum_{j,k}(I_{i,k})^2)$$

Kde α predstavuje parameter učenia sa, v ktorom určujeme chybu. Ak chceme, aby bola chyba čo najnižšia nastavíme regularizačný parameter na 0. Ak je potrebné, aby bol systém čo najviac objektívny (nie zameraný iba na vopred dané hodnoty) je potrebné ho zvýšiť. Našim cieľom je nastaviť tento parameter niekde medzi. Výhodou tohto algoritmu je, že je rýchly a pomerne jednoduchý.[20]

SGD založené na zaujatosti (Bias Stochastic Gradient Descent (B-SGD)) Je to veľmi podobná metóda ako Stochastické gradientné klesanie (Stochastic gradient descent (SGD)). Rozdiel je v tom, že B-SGD využíva inú cieľovú funkciu pre minimalizáciu. Algoritmus sa snaží využiť zaujatosť užívateľov a položiek. Toto dokážeme transformovať do vzorca, ktorý sa spomína [5]:

$$b_{i,j} = \mu + b_i + b_j$$

Kde zaujatosť hodnotenia užívateľa i a položky j ($b_{i,j}$) sa počíta ako priemerné hodnotenie naprieč celým algoritmom (μ) pripočítané k súčtu priemerného hodnotenia užívateľa i a položky j vzhľadom k priemernému hodnoteniu (b_i a b_j). Ako príklad použijeme hodnotenie filmu F užívateľom U. Priemerné hodnotenie v dátovej sade je 3.5. Priemerné hodnotenie F je o 0.5 vyššie ako priemer. U hodnotí kriticky, a teda jeho priemerné hodnotenia sú o 0.3 menej ako priemer. Celková zaujatosť teda bude 3.7 (3.5 + 0.5 - 0.3). Vzorec pre výpočet hodnotenia sa zväčší o $b_{i,j}$ a teda celkový vzorec pre faktorizáciu matíc bude:

$$(U, I) = \arg\min_{U,I} \sum_{i,j}[(U_i)(I_j)^T - b_{ij} - M_{ij}] + \frac{\alpha}{2}(\sum_{i,k}(U_i, k)^2 + (I_j, k)^2) + b_i^2 + b_j^2$$

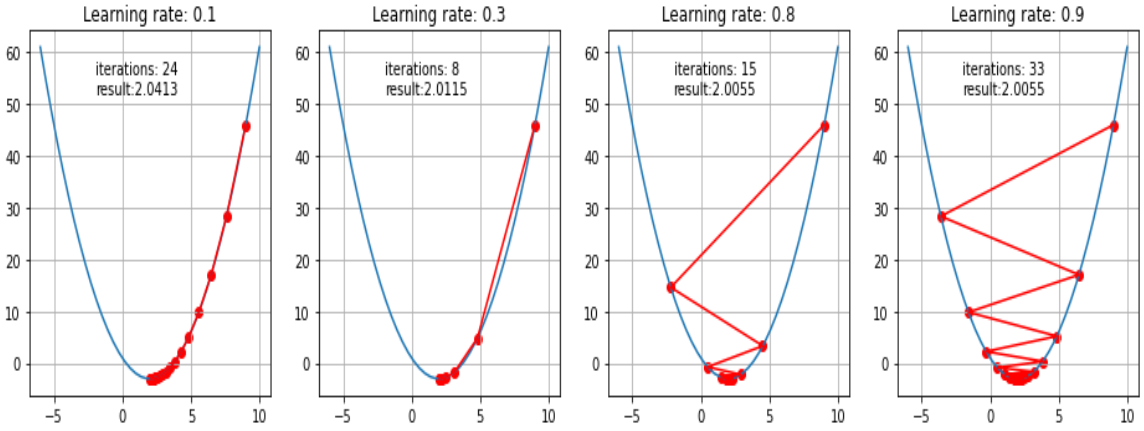
[5][3]

Gradientné klesanie (gradient descent) Je algoritmus využívaný pre nájdenie lokálneho minima diferenciovatelnej funkcie. Funguje tak, že iteratívne nájde gradient daného bodu, potom ho vyškáluje pomocou koeficientu učenia a odčíta výslednú hodnotu od aktuálneho bodu. V [19] je použitá nasledovná rovnica :

$$P_{(n+1)} = P_n - \eta \Delta f(P_n)$$

Kde P je krok, n je počítadlo pre kroky, η je koeficient učenia. V [19] sa spomína tento algoritmus:

1. Vyber si začiatočný bod.
2. Vypočítaj gradient v tomto bode.
3. Urob krok v opačnom smere gradientu.
4. Opakuj krok 2 a 3 pokiaľ nie je dosiahnuté maximálny počet krokov, alebo dĺžka kroku je menšia ako tolerancia.



Obr. 2.5: Grafické znázornenie gradientného klesania s rôznymi hodnotami koeficientu učenia. Prevzaté z [19]

Alternovanie najmenších štvorcov (alternating least squares (ALS)) Je to metóda veľmi podobná SGD algoritmu. Rovnako ako SGD, sa s maticou užívateľov a maticou položiek, ktoré sú na začiatku náhodne naplnené, snaží iteratívne dôjsť k najlepšej faktorizovanej podobe interakčnej matice. Rozdiel oproti SGD metóde nastáva v tom, že ALS v každej iterácií najprv optimalizuje jednu z matíc buď užívateľov alebo položiek s tým, že tá druhá matica ostane zafixovaná. V [2] sa spomína všeobecný postup tejto metódy:

1. Inicializuj maticu pre užívateľov a maticu pre položky.
2. Zafixuj maticu položiek, a optimalizuj maticu užívateľov znižovaním RMSE.
3. Zafixuj maticu užívateľov, a optimalizuj maticu položiek znižovaním RMSE.
4. Opakuj krok 2 a 3 až k oprimálnemu výsledku.

Výhodou alternovania najmenších štvorcov je, že pretvára problém, ktorý bol nekonvexný (pretože matica užívateľa aj položky sú neznáme) na konvexný (pretože jedna z matíc buď užívateľa alebo položky je fixná) a ten môže byť vyriešený optimálne. To znamená, že algoritmus má iba jedno lokálne minimum, ktoré je aj globálnym, takže sa nemôže stať, ako pri nekonvexnom probléme, že by algoritmus našiel lokálne minimum a to by ale nebolo globálnym. Ďalšou z výhod je, že ALS dokáže rátať paralelne, pretože ráta vlastnosti každej položky nezávisle od iných položiek a taktiež vlastnosti užívateľa od iných užívateľov.[5]

Váhove alternovanie najmenších štvorcov (WALS) Vo svojej podstate je Wals veľmi podobná Alternovanie najmenších štvorcov (alternating least squares (ALS)), používa ale inú cieľovú funkciu. Táto metóda sa snaží optimalizovať kolaboratívne filtrovacie algoritmy pre dátové sady, ktoré obsahujú implicitné hodnotenia[1]. Kedže z týchto hodnotení nie je jasné či sa uživateľovi položka páčila (užívateľ mohol napríklad kúpiť darček niekomu aj napriek tomu, že sa mu daná položka nepáči) Wals využíva rôzne úrovne spoľahlivosti tak, že je daná položka preferovaná užívateľom. v [3] je spomenutý tento vzorec :

$$(U, I) = \arg\min \sum_{i,j} [c_{ij}((U_i)(I_j)^T - M_{ij})] + \frac{\alpha}{2}(\sum_{i,k}(U_i, k)^2 + (I_j, k)^2)$$

Kde c_{ij} meria spoľahlivosť predikovanej hodnoty. V [4] sa okrem iných spomína aj tento vzorec pre výpočet spoľahlivosti:

$$c_{ij} = 1 + \alpha r_{ij}$$

v ktorom sa dôvera každým hodnotením zvyšuje. Zvyšovanie dôvery kontroluje konštanta α, ktorá podľa experimentov z toho istého zdroja mala dobré výsleky pri hodnote 40.[3][4]

2.1.5 Hybridné prístupy

Hybridné prístupy kombinujú 2 a viac metód za účelom získania čo najviac personifikovaného výsledku. Kombinujú sa, lebo každý prístup má svoje plusy a mínusy. Preto sa tento prístup snaží použiť len silné stránky každej z metód, ktoré spája. Podľa [10] sa hybridné systémy delia na: váhové, prepinajúce, zmiešané, kombinácia súvislostí, kaskádové, rozšírenie súvislostí a meta-levelové.

Váhový Hybridný prístup Pre tento prístup sa zadefinujú algoritmy, ktoré sú schopné mať najlepšie výsledky pri odporúčaní daného datasetu. Váhový prístup pomocou čísla presnosti algoritmov vypočíta váhu. Následne zoberie výstupy oboch algoritmov a pomocou váhy ich kombinuje. V [24] je spomenutý tento vzorec:

$$p_{u,i} = \sum_{f}^{c} (\sigma_f * p_{u,i}^{(f)})$$

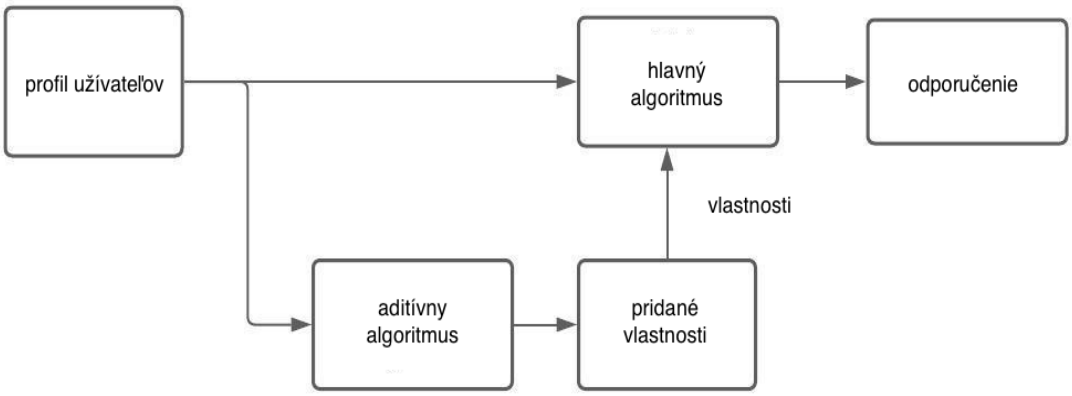
Kde p je predikcia, u a i sú indexy pre užívateľov a položky, c je počet použitých algoritmov a σ_f je váha pre konkrétny algoritmus.

[1] Implicitné hodnotenia sú hodnotenia ktoré užívateľ udelí nepriamo. Napríklad nejaká televízna stanica vysiela rôzne programy a ako hodnotenie od užívateľa berie dĺžku sledovania daného programu. Opakom Implicitných hodnotení su explicitné, kde užívateľ priamo zadá hodnotenie (zadá hodnotenie na nejakej škále)

Prepínajúci Hybridný prístup Táto metóda musí mať vopred zadefinované pravidlá, aký algoritmus sa má použiť pri danej situácií. Príkladom je prihlásenie sa nového užívateľa. Využíva sa preto, aby sme sa vyhli problému tzv. studeného štartu pri kolaboratívnom filtrovaní, preto použijeme nejakú z obsahovo založených metód.

Zmiešaný Hybridný prístup V tomto prístupe sa rozdelí dátová sada na základe užívateľov a ich preferencií na menšie kandidátne dátové sady. Pre každú z týchto sád sa vyberie najlepšia metóda, ktorá pre ňu bude mať najlepší výkon. V podstate to znamená, že tento prístup vezme ako vstup sadu kandidátnych dátových sád a jeho výstupom bude predikcia. [16]

Prístup kombináciou súvislostí Kombinácia súvislostí znamená, že kombinujeme algoritmy. Funguje tak, že jeden (hlavný) algoritmus vyžaduje súvislosti napr.obsahovo založené odporúčacie systémy a sekundárny algortimus bude hlavnému tieto súvislosti dodávať - budú mu injektované. Napríklad súvislosti získané pomocou faktorizácie matíc budú dodané hlavnému algoritmu, ktorý s nimi bude pracovať spolu s vlastne získanými.[16]



Obr. 2.6: Grafické znázornenie hybridného prístupu Kombináciou súvislostí. Prevzaté a upravené z [16]

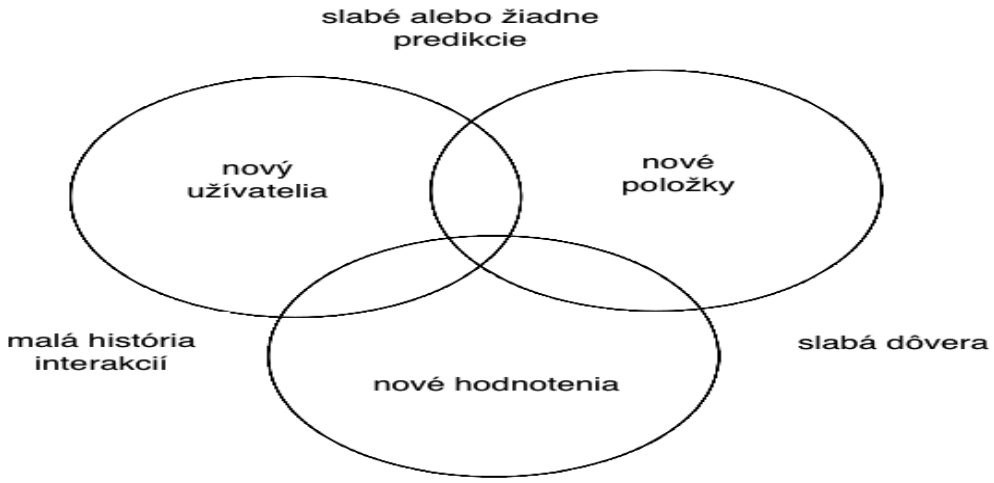
Kaskádový hybridný prístup Kaskádový hybridný prístup definuje hierarchiu odporúčacích systémov tak, že hlavný odporúčací systém produkuje primárný výsledok - hrubý zoznam odporúčaní a pomocou sekundárneho odporúčacieho systému sa tieto odporúčania spresňujú. [17]

Rozšírenie súvislostí Tento prístup zreťazuje 2 metódy odporúčacích systémov tak, že sekundárny model generuje hodnotenia užívateľov, ktoré sa následne použijú v hlavnom systéme na predikovanie finálnych odporučení. Rozšírenie súvislostí dokáže zlepšiť výkon odporúčacieho systému bez potreby meniť hlavný model.[16]

Meta-levelový prístup Podobne ako Rozšírenie súvislostí meta-levelový prístup zreťazuje 2 metódy do jednej. Rozdiel je ale v tom, že hlavný model tejto metódy neberie ako vstup iba hodnotenia vyprodukované sekundárnym modelom ale celý model. [17]

2.2 Problémy pri vytváraní odporúčacích systémov

Rôzne metódy v odporúčacích systémoch pracujú s rôznymi typmi dát. Niektoré používajú metadáta o produktoch a užívateľoch a iné si zas vystačia iba s interakčnou maticou. Toto prináša aj problémy, ktoré niektoré algoritmy zvládajú lepšie a iné horšie. Často sa to dá vyriešiť pomocou hybridných odporúčacích systémov. V [10] sú spomínané tieto problémy: problém chladného štartu, problém riedkosti dát, škálovateľnosť, rôznorodosť a habituačný efekt.



Obr. 2.7: Grafické znázornenie troch hlavných príčin problémov ktoré vznikajú v odporučovacích systémoch. Prevzaté a upravené z [30]

Problém chladného štartu Pojem chladného štartu je prevzatý z automobilového priemyslu.Súvisí to so situáciou, keď motor nie je správne zahriaty z čoho vznikajú problémy. Akonáhle sa zahreje na určitú teplotu, začne pracovať bez problémov. V podstate rovnaký problém nastáva aj v odporúčacích systémoch, keď je do interakčnej matice pridaný buď nový užívateľ alebo položka. Nemajú žiadne interakcie a pre metódy ako kolaboratívne filtrovanie je problém nájsť adekvátne odporučenie.[10]

Problém riedkosti dát Tento problém nastáva pre nedostatok dát v interakčnej matici, pretože priemerní používatelia majú tendenciu ohodnotiť málo položiek. Toto vedie k prázdnym alebo neznámym interakciám (až 99% takýchto interakcií). Takto prázdne matice sú jedným z hlavných problémov v odporúčacích systémoch a sú príčinou neschopnosti nájsť suseda, tým pádom vedú k množstvu slabých odporúčení. Tiež sa to spája s problémom pokrytia dát, keď je málo percent ohodnotených položiek v systéme, pre ktoré môže byť vytvorené odporúčenie.[17]

Problém škálovateľnosti V dnešnej dobe je tento problém oveľa citelnejší, pretože komerčné produkty, ktoré využívajú odporúčacie systémy obsahujú množstvo užívateľov a položiek. Problém škálovateľnosti spočíva v tom, že algoritmus sa s narastajúcim počtom užívateľov začne chovať inak, ako je od neho očakávané. Keď má systém problém so ška-

lovateľnosťou, začne byť presnosť jeho odporúčaní čoraz horšia. Tiež nastanú problémy, aké systém nemal keď bolo užívateľov menej. V [21] sa spomína delenie tohto problému na: hardverový problém škálovateľnosti a softvérový problém škálovateľnosti. Hardvérový problém sa dá vyriešiť zvýšením výpočetnej sily na strane serveru. Softvérový sa dá opraviť tým, že sa použije iný algoritmus, ktorý by to zvládal lepšie. Pravdou ale je, že nie vždy na daný problém existuje riešenie. Je to problém, pre ktorého riešenie bude v budúcnosti potrebné vynájdenie nových algoritmov.[21]

Problém s rôznorodosťou odporúčaní Tento problém súvisí s položkami, ktoré sú príliš podobné. V [21] je spomenutý príklad s cukrom - ak užívateľ kupuje pravidelne na stránke cukor, automaticky ako danú stránku otvorí, mu bude odporúčaný cukor (iných značiek atd.). Ak sa toto stáva vždy, hrozí, že zákazník prejde na inú stránku, kde mu budú položky odporúčané rôznorodejšie. Jedným z riešení, je odporučiť zákazníkovi produkty, ktoré nie sú podobné, ale sú pre užívateľa zaujímavé.[21]

Habituačný efekt Je problém, ktorý vzniká pri marketigových stratégiach. Keď v užívateľských rozhraniach, za účelom čo najväčšej predajnosti produktov, má užívateľ dostupných množstvo informácií o položkách napr. počet hodnotení, obrázky položky, informácie o položke atd. Akonáhle užívateľ príde do kontaktu s takýmto masívnym množstvom informácií, často dochádza k tomuto efektu, ktorý väčšinou končí tzv. fenoménom bannerovej slepoty. Čo znamená, že hoci je odporúčenie optimálne z hľadiska algoritmu, algoritmus môže produkovať nesprávne výsledky.[10]

2.3 Meranie presnosti algoritmov pre odporúčacie systémy

Dôležitou časťou odporúčacích systémov je meranie presnosti (kvality) jednotlivých metód. Každá dátová sada je rozdielna. Líši sa v mnohých aspektoch ako sú veľkosť, hustota hodnotení (pri odporúčacích systémoch) a iné. Z toho dôvodu nemá každý algoritmus rovnakú presnosť na jednej dátovej sade. Je preto nesmierne dôležité ohodnotiť ich výkon. Práve k tomu sa používa meranie ich kvality. Často používané meranie je priemer. Priemer je zlomok správných odporučení v pomere k všetkým možným odporučeniam.

Podľa [21] sa metriky pre meranie priemeru odporúčacích systémov delia na štatistické a metriky presnosti podpory rozhodovania sa. Vhodnosť použitia týchto metrík zavisí na vlastnostiach dátovej sady a typu úlohy, ktoré dané odporúčacie systémy plnia.[21]

2.3.1 Rozdelenie dátovej sady

Pre predikovanie hodnotení pomocou metód kolaboratívneho filtrovania, kde je potrebný model, sa musí dátová sada rozdeliť na trénovaciu sadu a testovaciu sadu. Trénovacia sada slúži pre vytváranie - "trénovanie"modelu. Testovacia slúží na otestovanie presnosti daného modelu. Pomer v akom treba rozdeliť dátovú sadu záleží na algoritme a na mnoho aspektoch, ktoré súvisia s danou dátovou sadou. Trénovacia sada obsahuje všetky položky, len z niektorých sú zámerne odobrané hodnotenia - tieto hodnotenia sa nachádzajú v testovacej sade. Ako z názvu testovacej sady vyplýva, táto sada slúži na trénovanie (vytvárania modelu) na danných dátach, a pomocou konkrétnych metrík sa testuje ích presnosť na testovacej sade. V [23] sa spôsoby delenia dátovej sady delia na: ponechanie posledného, užívateľské-časové/globálne rozdelenie (Temporal User/Global Split), náhodné rozdelenie a užívateľské rozdelenie.

Ponechanie posledného

Metóda ponechania na posledného, ako vidíme v názve, ponecháva posledné hodnotenie užívateľa pre testovanie, ostatné hodnotenia sú ponechavané na trénovanie. Výhodou tejto metódy je, že ponecháva maximálny počet hodnotení pre trénovanie. Na opačnej strane je dát na testovanie málo, pretože pri testovaní s danou testovacou sadou sa nemusí reflektovať efektivita pre užívateľa.[23]

Užívateľske/globálne rozdelenie (Temporal User/Global Split)

Tento postup rozdeľuje interakcie podľa percent na základe časovej značky (napr. 20 percent hodnotení užívateľa je použitých pre testovanie). Avšak podľa [23] sa tento postup delí na: užívateľské-časové a globálne rozdelenie.[23]

Užívateľské-časové rozdelenie Užívateľské-časové rozdelenie je značne podobné metóde ponechania na posledného. Rozdiel nastáva v tom, že toto rozdelenie ponecháva na testovanie viac ako jedného posledného.[23]

Globálne rozdelenie Globálne rozdelenie definuje časovú značku, ktorá je rovnaká pre všetkých užívateľov. Všetky hodnotenia ktoré boli uskutočnené za týmto bodom sa ponechávajú ako testovacie dáta.[23]

Náhodné rozdelenie

Tento prístup rozdeľuje dátovú sadu na testovaciu a trénovaciu náhodne. V minulých odporúčacích systémoch sa pre testovaciu sadu vyberala práve jedna položka, teraz sa ich vyberá viac.[23]

Užívateľské rozdelenie

Užívateľské rozdelenie je menej používaná metóda, ktorá rozdeľuje dátovú sadu na základe užívateľov, nie položiek. Princíp tohto prístupu spočíva v tom, že niektorí užívateľia sú (spolu so všetkými hodnoteniami) ponechaní pre testovanie a ostatní sú použití na trénovanie. Táto stratégia sa dá použiť iba pri metódach, ktoré netrpia studeným štartom a teda vedia urobiť predikciu aj pre nového užívateľa.[23]

2.3.2 Štatistické metriky

Často používané metriky sú RMSE a MAE. Pracuje sa s nimi dobre, pretože obe tieto metódy pracujú s hodnoteniami na tej istej škále ako pôvodné hodnotenie. [18]

Mean average error (MAE)

Jednou z hlavných charakteristík MAE je, že ak sa vyskytnú nejaké extrémne chybové hodnoty, tak ich rovnomerne rozloží na ostatné hodnotenia. Pre toto sa MAE využíva práve vtedy, keď sa hľadá presnosť odporúčacích systémov a veľmi sa neprikladá na dôležitosti odľahlých hodnôt.[18]

$$\text{MAE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

Kde $\hat{R}$ je časť interakčnej matice (testovacie dáta), r_{ui} je predikované odporučenie a $\hat{r}_{ui}$ je reálne hodnotenie.

Root mean square error (RMSE)

Táto metrika oproti MAE je omnoho viac náchylnejšia na ovplyvnenie odľahlými hodnotami. Dôvodom je, že nevyužíva absolútne hodnoty a teda ich umocňuje na druhú. Odmocnina sa následne vyráta až z výsledneho čísla.[18]

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}.$$

Kde $\hat{R}$ je časť interakčnej matice (testovacie dáta), r_{ui} je predikované odporučenie a $\hat{r}_{ui}$ je reálne hodnotenie. RMSE je v porovnaní s MAE využívaná viac, pretože pracuje so zapornými hodnotami, teda ich nerobí absolútnymi.

2.3.3 Metriky presnosti podpory rozhodovania sa

Medzi často používané metriky presnosti podpory rozhodovania sa, patria podľa [21] tieto: spätné hodnodenie (reversal rate), vážené chyby, Receiver Operating Characteristics (ROC) a Precision Recall Curve (PRC), presnosť, recall a f-meranie(meassure). Tieto metódy klasifikujú položky tak, že vnímajú postup predikcie ako binárne operácie, ktoré odlišujú "dobré"predpovede od tých zlých. Pre príklad sa môže použiť užívateľ, pre ktorého je zadefinované dobré hodnotenie položky 4 a viac, pri škále od 0 do 5. Pomocou nejakeho algoritmu odporúčacích systémov sa zistí predikcia, ktorá je 4.5, v tom prípade sa berie predikcia ako dobrá, v prípade ak by predikcia bola menšia ako 4 sa berie ako zlá. Presnosť sa dá vypočítať ako:

$$\text{Presnosť} = \frac{\text{Správne ohodnotené položky}}{\text{Celkový počet položiek}}$$

Recall sa dá vyjadriť ako:

$$\text{Recall} = \frac{\text{Správne ohodnotené položky}}{\text{Celkový počet užitočných položiek}}$$

F-meranie pomáha jednoducho zlúčiť presnosť a recall do jednej metriky. Výsledok porovnáva algoritmy s dátovými sadami veľmi ľahko. F-meranie sa dá definovať ako:[21]

$$\text{F-meranie} = \frac{2PR}{P + R}$$

2.4 Optimalizácia hyperparametrov

Cieľom optimalizácie hyperparametrov v strojovom učení je nájdenie hyperparametrov pre vytvorenie nejakého modelu, ktorých výsledkom je najlepšia presnosť na overovacej dátovej sade. V [12] je Optimalizácia hyperparametrov znázornená týmto vzorcom:

$$x^\star = \arg\min_{x \in X} f(x)$$

Kde f(x) reprezentuje hodnotu, ktorú algoritmus mimimalizuje (V prípade odporúčacích systémov ide najčastejšie o 2.3.2 alebo 2.3.2), x* reprezentuje sadu hyperparametrov, ktoré sa ladia pričom x patrí množine X. Z pohľadu odporúčacích systémov ide hlavne o: parameter učenia sa (learning rate), regularizačný parameter a počet faktorov v maticiach užívateľov a položiek. Výsledkom tohto algoritmu je najlepšia kombinácia parametrov. Problémom v hľadaní optimálnych hyperparametrov je, že pri každej kombinácií vstupných parametrov musíme vytrénovať model na trenovacej sade a následne vyrátať presnosť na testovacej sade. Je to časovo náročný proces. Metódy pre ladenie hyperparametrov sú napríklad: Mriežkové hľadanie (grid search) a náhodné hľadanie, tieto 2 sú síce o niečo lepšie ako manuálne hľadanie, avšak ešte lepšou metódou je Bayesovská optimalizácia pre hľadanie hyperparametrov[12]

2.4.1 Bayesovská optimalizácia pre hľadanie hyperparametrov

Tento prístup si oproti mriežkovému hľadaniu (grid search) a náhodnému hľadaniu uchováva výsledky minulých meraní, ktoré využíva pre nájdenie nových hyperparametrov, čo budú spustené v dalšej iterácií. Práve preto je potrebých na nájdenie optimálnych hyperparametrov oveľa menej iterácií trénovania modelu[12]. V [12] sa spomínajú tieto kroky fungovania Bayesovskej optimalizácie.

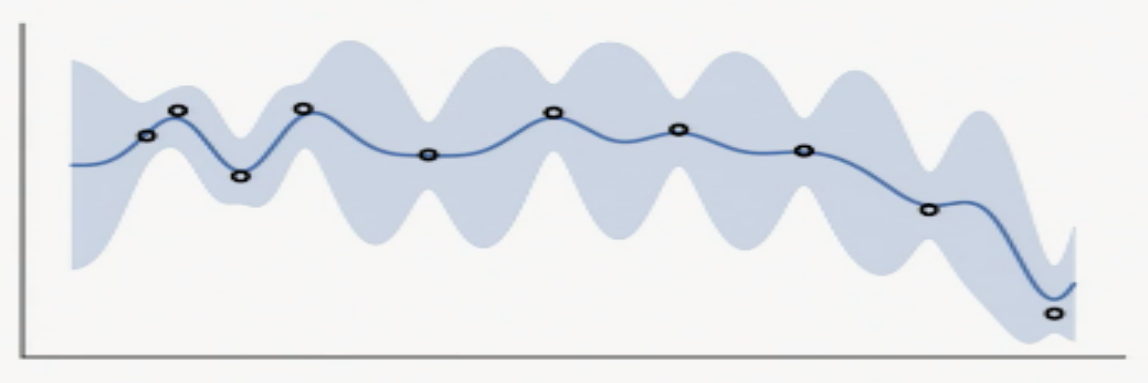
1. Vytvor dočasný model cieľovej funkcie.
2. Nájdi hyperparametre, ktoré majú najlepšie výsledky na dočasnom modeli.
3. Aplikuj tieto hyperparametre na cieľovú funkciu,
4. Aktualizuj náhradný model zahŕňajúci nové výsledky.
5. opakuj kroky 2-4 pokiaľ nie je dosiahnutých maximálnych iterácií alebo nevypršal čas.

Ako je možné vidieť na algoritme spomenutom vyššie, na začiatku sa vytvorí tzv. dočasný model. V [25] je definovaný tento model ako: “pravdepodobnostná reprezentácia cieľovej funkcie”. Čo znamená, že model je trénovaný na hyperparametroch a skóre cieľovej funkcie na nich. Matematicky zapísané:

$$P(\text{skóre cieľovej funkcie} \mid \text{hyperparameter})$$

Formalizácia Bayesovskej optimalizácie sa nazýva Sekvenčná modelovo-založená optimalizácia. Sekvenčná v názve znamená, že v tejto metóde bežia pokusy jeden po druhom. Každý raz bežia s vylepšenými hyperparametrami pomocou použitia Bayesovskej optimalizácie[12]. V [12] je spomenutých 5 aspektov sekvenčnej modelovo-založenej optimalizácie:

- Doména hyperparametrov, v ktorých sa má vyhľadávať.
- Cieľová funkcia, ktorá berie ako vstup hyperparametre a jej výstupom je hodnota, ktorú chceme minimalizovať.
- Dočasný model cieľovej funcie
- Kritéria, pre rozhodovanie,ktoré hyperparametre sa vyberú z dočasného modelu
- Záznam, ktorý pozostáva z párov v tvare (hyperparameter, výsledok), ktoré sú využívané algoritmom pre aktualizáciu dočasného modelu



Obr. 2.8: Grafické znázornenie náhradnej funkcie (modré pozadie) na cieľovej funkcií (modrá čiara). Prevzaté z [25]

Kapitola 3

Existujúce riešenia

Ako sa odporúčacie systémy začali čoraz viac dostávať do popredia, zároveň vznikali aj mnohé implementácie metód v nej obsiahnutej. Tieto implementácie sú naprogramované v rôznych jazykoch. V pythone je naimplementovaných niekoľko takýchto knižníc.

3.1 Surprise

Surprise je open-source knižnica pre vytváranie odporúčacích systémov. Pracuje s dátami s explicitnými hodnoteniami. Názov Surprise je odvodený z názvu Jednoduchý pythonovský engine pre odporúčacie systémy (Simple Python RecommendatIon System Engine). V [11] sa spomínaju tieto účely prečo bol tento balík vytvorený:

- Dať užívateľom kontrolu nad ich experimentami. Veľku rolu v tomto zohráva ich dokumentácia. Snažili sa ju vytvoriť precízne a čisto, aby vytkli každý detail algortmu.
- Ľahká práca s dátovou sadou.
- Ľahká implementácia nápadov nových algoritmov.
- Kritéria, pre rozhodovanie ktoré hyperparametre sa vyberú z dočasného modelu
- Sprístupniť prostriedky na meranie, analyzovanie a porovnávanie výkonu algoritmov. Krížová validácia sa spúšta veľmi jednoducho.

Výhodou Surprise knižnice, tak ako je spomenuté vyššie, je práca s dátovou sadou. Načítanie sady je intuitívne a pracuje sa s ňou dobre. Pre vytvorenie enginu pre odporúčací systém je táto knižnica dobrou voľbou, pretože okrem iného niektoré kritické časti algoritmov, ktoré má v sebe implementované sú písané v jazyku C (Je všeobecne známe, že ak projekty, ktorých implementácia je písaná v tomto jazyku sú napísané kvalitne, tak bežia veľmi rýchlo). Ďalšou z výhod tejto knižnice je, že sú v nej implementované hlavné algoritmy pre odporúčacie systémy ako vidíme v tabuľke 5.1. Z používateľského hľadiska je dokumentácia, ako je spomenuté v bodoch vyššie, veľmi dobre a kvalitne spracovaná. Obsahuje nielen informácie o používaní danej knižnice ale aj detailne spracované funngovanie jednotlivých algoritmov z matematického a teoretického hľadiska.

3.2 Pyspark

Ako sa píše v jeho dokumentácií[1] Pyspark je rozhranie pre Apache Spark v Pythone. Nielenže umožňuje užívateľovi písať aplikácie v Sparku s použitím pythonu, ale poskytuje mu celý ekosystém Sparku ako Spark SQL, DataDrame, Streaming, MLlib (Strojové učenie) a Spark Core. Veľkou výhodou Pysparku je, že syntax pri práci s dátovou sadou je veľmi podobná Pandas syntaxi. To znamená, že ak užívateľ pozná syntax Pandas balíka, tak prakticky vie hneď pracovať v Pysparku s dátovou sadou. Ďalšou výhodou je, že v priebehu programu sa dá ľahko pracovať a konvertovať Pyspark dátovú sadu na Pandas dátovú sadu a naopak.

	ALS	SVD	SGD	Pearsonov korelačný koeficient	Euklidovská vzdialenosť	Kosínusová vzdialenosť
Pyspark	✓	✗	✗	✓	✓	✗
Surprise	✓	✓	✓	✓	✗	✓

Tabuľka 3.1: implementované metódy Kolaboratívneho filtrovania a časť implementovaných metód pre nájdenie podobnosti v knižniciach Pyspark a Surprise

3.3 Recmetrics

Z dokumentácie[2] je zjavné, že Recmetrics je Python knižnica hodnotiacich metrík a diagnostických nástrojov pre odporúčacie systémy. To znamená, že pracuje skôr na vizualizáciu dát a štatistiky dátovej sady pred aj po predikovaní odporučení. Recmetrics dokáže spracovať dátovú sadu, aby bol program schopný vizualizovať veci ako: Long tail problém, pokrytie, štatistiky o tom koľko položiek nebolo odporučených užívateľovi a mnoho ďalších.

3.4 TensorFlow Recommenders

TensorFlow Recommenders je knižnica[3] pre vytváranie odporúčacích systémov pomocou TensorFlow. Dokáže pomôcť s celou tvorbou odporúčacieho systému, čo zahŕňa: prípravu dát, formuláciu modelu, trénovanie dát, testovanie modelu a jeho nasadenie. Je postavený na systéme Keras. Cieľom je mať jemnú krivku učenia sa, a zároveň stále dokáže poskytovať flexibilitu pri vytváraní zložitých modelov ako hlboká faktorizácia matíc.

[1] https://spark.apache.org/docs/latest/api/python/
[2] https://github.com/statisticianinstilettos/recmetrics
[3] https://www.tensorflow.org/recommenders/examples/quickstart

Kapitola 4

Návrh a implementácia aplikácie

Jednou z častí tejto práce bola prezentácia výsledkov pomocou aplikácie. Ešte pred začatím návrhu aplikácie bolo potrebné naštudovať si informácie ohľadom odporúčacích systémov. Tieto poznatky sú zhrnuté v 2. Táto kapitola sa bude zaoberať návrhom a implementáciou aplikácie a budú predstavené knižnice, ktoré boli využité pre jej implementáciu. Aplikácia je vytvorená ako webové rozhranie. Pomocou webového rozhrania sa ľahko pracuje s transformáciou dát a ich následnou vizualizáciou. Čo je veľmi dôležitá vlastnosť, ktorú využíva aj táto aplikácia.

4.1 Využité technológie

Knižnica pandas Pandas[1] je open-source knižnica v prostredí python. Umožňuje prácu s dátami, ktoré sú uložené v rôznych formátoch. Práca s dátami je možná pomocou objektu *Dataframe*, ktorý má v sebe integrované indexovanie. Pomocou tohto objektu je s dátami možné robiť veci ako ich agregovanie, spájanie objektov, meniť jej tvar a mnoho ďalších. Veľkou výhodou je, že je veľmi dobre optimalizovaný pre výkon tak, že kritické pasáže kódov sú písane v Cythone alebo v jazyku C.

Knižnica flask Flask je mikro webový framework napísaný v jazyku Python[2]. Mikro neznamená, že má slabú funkcionalitu. Znamená to, že cieľom je udržať jeho jadro ľahké a udržateľné. Výhodou je, že podporuje prácu s knižnicami, ktoré ľahko umožňujú perzistenciu dát (napr. sqlalchemy). Výhodou tiež je možnosť využívať šablóny pri práci s html súborom (napr. šablóna pre navigačnú lištu). Šablóny je možné použiť vďaka enginu *Ninja*.

Chart.js Charts.js[3] je knižnica napísaná v jazyku Javascript. Umožňuje z dodaných dát vykresľovať interaktívne grafy. Dáva možnosť výberu z rôznych typov grafov ako koláčový, lineárny, stĺpcový a mnoho ďalších. Tiež je možnosť prispôsobovať zmeny grafov na základe rôznych interakcií od užívateľa.

Surprise Je knižnica písaná v jazyku Python pre odporúčacie systémy. Viac je o nej písané v kapitole 3.1. Knižnica je použitá z [11].

[1] https://pandas.pydata.org/about/index.html
[2] https://flask.palletsprojects.com/en/2.1.x/
[3] https://www.chartjs.org/docs/latest/

BayesOptimalization Knižnica[4], napísaná v jazyku python. Umožňuje hľadať optimálne hodnoty pre hyperparametre v niekoľkých krokoch. Využíva Bayesovskú optimalizáciu.

4.2 Štruktúra aplikácie

Aplikácia je štruktúrovaná podľa vzoru *MVC* (*model view controller*).

View je časť, ktorá je zodpovedná za vizualizáciu dát, inak povedané je to užívateľské rozhranie. V tejto aplikácií je implementovaný pomocou značkovacieho jazyka HTML spolu enginom, ktorý umožňuje použiť flask, Ninja. Pomocou Ninja je vytvorená šablóna pre navigačnú lištu. Na prácu s interakciami užívateľa, ktoré boli potrebné pri určitých stránkach bol využitý jazyk *Javascript*. Prehľadnosť a krajší vzhľad je dotvorený prevažne pomocou frameworku *Bootstrap*. Niektoré detaily boli dotvárané jazykom *CSS*.

Model má na starosti celú logiku dát. To znamená, že vstupy od užívateľa, transformuje na dáta, ktoré majú nejakú výpovednú hodnotu. Flask je zvolený ako nástroj, s ktorým je implementovaný *model*. Jeho vstupom sú v tejto aplikácií primárne súbory, v ktorých je uložená vybraná dátová sada. Perzistenciu dát má na starosti trieda *datasets*, v ktorej sú uložené názov súboru, názvy potrebných stĺpcov v dátovej sade a dalšie rôzne interné záležitosti, ako dáta potrebné pre vykreslenie stránky so štatistikami. Z tejto dátovej sady sa následne robia rôzne štatistiky, ako hustota alebo veľkosť. Pripravujú sa dáta pre zobrazovanie grafov a vytvárajú sa odporúčania pre danú sadu.

Controller prepája vstupy od užívateľa s modelom. V aplikácií sa na to používajú protokoly *GET* a *POST*. Pomocou formulárov užívateľ zadá požadované parametre, ktoré sú následne pomocou spomenutých protokolov predané modelu.

4.3 Implementácia

4.3.1 Spracovanie dát od užívateľa

Na začiatku je od užívateľa vyžadovaný vstupný formulár, v ktorom je potrebné vyplniť informácie o dátovej sade a samozrejme samotný súbor, ktorý ju obsahuje(ak je potrebné aj súbor s metadátami o stĺpci pre položky). Formulár je znázornený na obrázku 4.1. Po vyplnení a odoslaní tohto formulára sa dáta spracuvávajú nasledovne:

1. Najskôr sa skontroluje, či boli vyplnené všetky povinné polia (názov pre stĺpec užívateľov, položiek, hodnotení a súbor s hodnoteniami). Ak sú zadané aj iné, ako napríklad separátor pre súbor s metadátami o položkách, skontroluje sa či bol zadaný aj daný súbor. Ak je všetko v poriadku urobí sa dočasná inštancia triedy *datasets* (nie je to trieda *Dataset*) a uložia sa tam všetky premenné.

2. Otestuje sa, či sú vložené súbory formátu json alebo csv. Test prebieha tak, že sa z objektu pre súbor, ktorý bol vyžiadaný z formuláru, vezme atribút *content_type*. Ten vráti typ súboru. Ak daného typu nie sú, vyvolá sa chybová hláška.

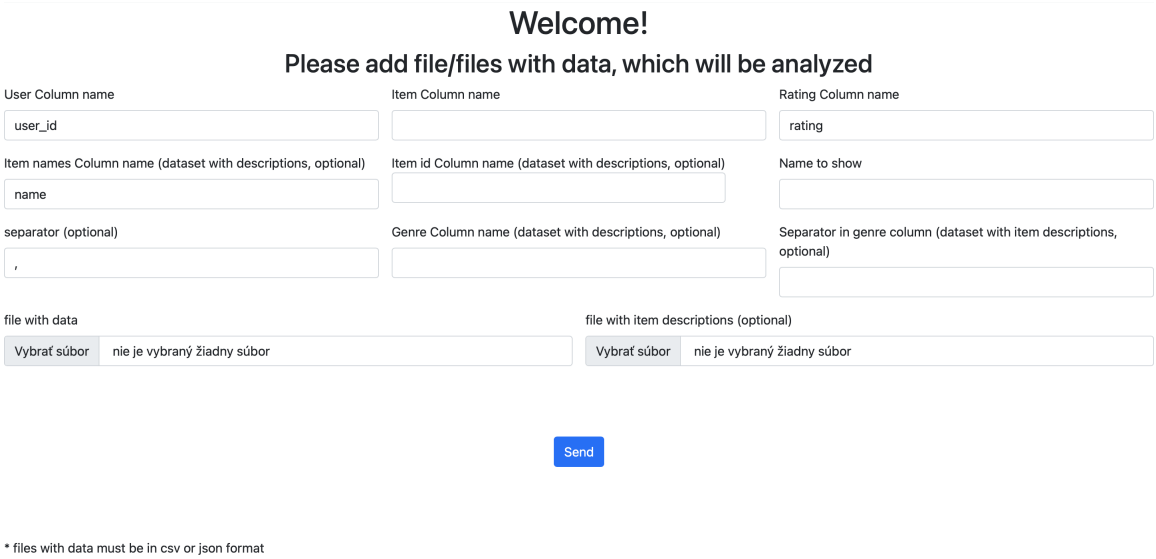
3. Urobí sa pomocný *Dataframe* zo súboru s hodnoteniami, pomocou metódy z knižice pandas *read_csv* alebo *read_json* záleží na type súboru. Zavolá sa jej atribút *columns*, ktorý vráti zoznam stĺpcov a otestuje sa či sedia názvy, ktoré užívateľ zadal.

[4] https://github.com/fmfn/BayesianOptimization

4. Ak je zadaný aj súbor pre metadáta o položkách, urobí sa identická kontrola ako v kroku 3. Rozdiel je len v kontrolovaných stĺpcoch (v tomto súbore sa kontroluje stĺpec s id položiek, stĺpec s názvami položiek a ak je potrebné aj stĺpec so žánrami).

5. Ak je všetko v poriadku vytvorí sa z daných dátových sád inštancia triedy *Dataset*. Táto trieda slúži na prácu s dátovou sadou. Dokáže vytvárať štatistiky o sade, pridávať hodnotenia, aj užívateľov a taktiež vytvárať modely odporúčacích systémov. Táto trieda sa uloží pomocou dátovej serializácie *pickle*. Jej názov sa uloží.

6. Nakoniec sa dočasná inštancia triedy *datasets* uloží. Pomocou vygenerovaného id danej inštancie sa vytvorí názov, ktorý má tvar: názov súboru + id.



Obr. 4.1: Ukážka formulára vo výslednej aplikácií.

4.3.2 Trieda Dataset

Dataset je interná trieda, ktorá slúží na spracovanie dátovej sady, reprezentáciu dát a vytváranie modelu.

Tvorba inštancie triedy sa rieši pomocou konštruktora danej triedy. Pri zavolaní konštruktora je potrebné zadať paramentre, ktorými sú :

- Dátový rámec s interakčnou maticou (povinný).
- Názvy stĺpcov v danom dátovom rámci. Konkrétne názov pre stĺpec užívateľov, položiek a hodnotení (povinné). V prípade, že je v danom dátovom rámci stĺpec s názvami položiek, tak je požadovaný jeho názov (ak by nebol zadaný, vymazal by sa a teda by sa zobrazovalo id položky namiesto jej názvu).
- Dátový rámec s metadátami o položkách (nepovinné).

- Názvy stĺpcov v dátovom rámci s metadátami. Ide o názov stĺpca s id jednotlivých položiek a názvami položiek (povinné v prípade, že bol uvedený daný dátový rámec). Voliteľnou položkou je stĺpec so žánrom, alebo inou vlastnosťou danej položky.

Algoritmus prebieha tak, že v prvom rade sa zmenia názvy stĺpcov v dátovom rámci s interakčnou maticou (pretože každá dátová sada ma iné názvy, a teda pre lepšiu prácu s nimi ich premenujeme všeobecne). Premenujú sa nasledovne: názov stĺpca s užívateľmi sa premenuje na *user_id*, s položkami na *item_id*, s hodnoteniami na *rating_id* a ak je v danom rámci aj stĺpec s názvami položiek, tak sa premenuje na *name*. Výsledok sa uloží do atribútu triedy *dataset*. Ak je zadaný aj rámec s metadátami premenuje sa jeho stĺpec s názvami na *name* a stĺpec žánrov (ak sú zadané) na *genre*. Ďalej sa tieto dva rámce (ak sú zadané oba) musia spojiť do jedného a to pomocou metódy v knižnici pandas *merge*, na základe id položiek. Takto spojený rámec sa uloží do atribútu *name_and_id_item*. Nakoniec sa predpripravia atribúty *trained_result_dict* a *hyperparams_result*, ktoré sú použité pre uloženie výsledkov z hľadania hyperparametrov a výsledkov predikovaných odporučení.

4.3.3 Pridávanie hodnotení do dátovej sady

Na stránke sa zobrazí sto náhodne vygenerovaných položiek, ktoré užívateľ dokáže ohodnotiť pomocou posuvnej lišty. Dáta sa na pozadí zbierajú do slovníka, ktorý má tvar:

```
"id_polozky" : [meno_polozky, hodnota_ohodnotenia]
```

Ak je v rámci aj stĺpec pre žáner tvar je takýto :

```
"id_polozky" : [meno_polozky, hodnota_ohodnotenia, zaner]
```

Takýto slovník sa posiela do modelu, ktorý buď iba pridá hodnotenia do rámca, alebo rovno aj vytvorí model na základe, ktorého predikuje odporučenie. Záleží, akú možnosť zvolí užívateľ pri odoslaní dát. Pridanie do slovníka prebieha v metóde triedy *Dataset* *make_dat_with_name* tak, že sa vytvorí užívateľ (ak už nieje vytvorený). Id užívateľa je uložené v atribúte danej triedy *user_id*. Id položiek, hodnotenie, id užívateľa a ak je potrebný aj názov položky, tak sa na začiatku ukladajú do pomocného dátového rámca, ktorý sa následne pomocou funkcie *concat* spojí s atribútom danej inštancie *Dataset*. Zadávanie hodnotení v aplikácií je zobrazené na obrázku 4.2. Vytváranie modelu odporúčacieho systému je bližšie popísané v 4.3.5.

Ak je zvolená možnosť tvorby modelu, je potrebné ho najskôr vytrénovať na aktuálnych dátach (aj pridaných). Následne sa zavolá metóda *find_predictions* z triedy *Dataset*, pomocou ktorej sa predikujú dáta. Táto funkcia berie ako vstupné parametre: cestu kde sa majú uložiť predikované dáta[5], model , ktorý je uložený v inštancií *datasets* a názov použitého algoritmu (pre uloženie do slovníka). Funkcia sa vykonáva v nasledovných krokoch :

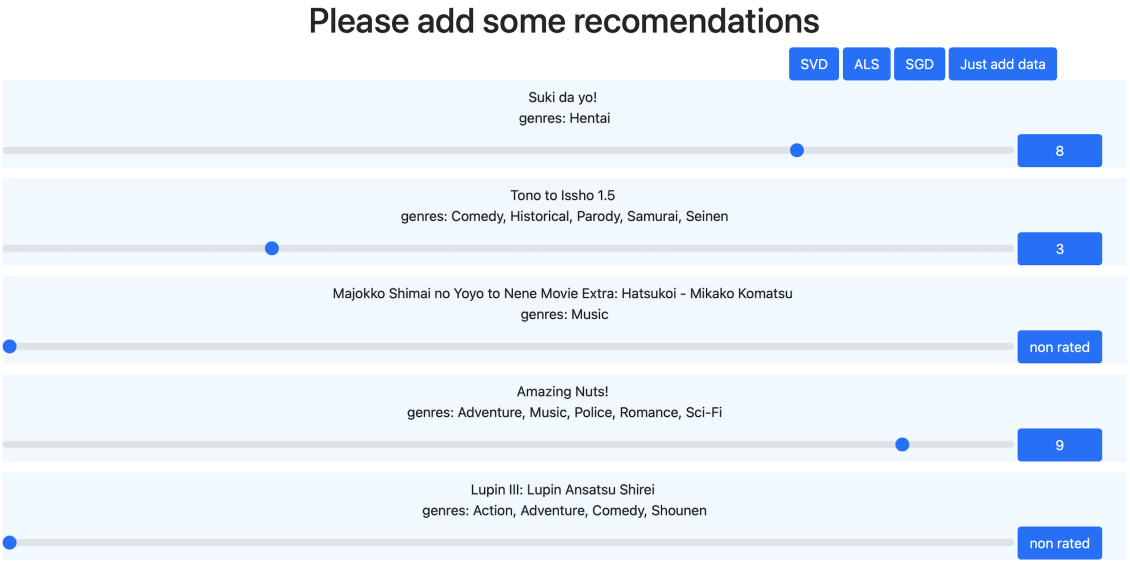
1. Na začiatku sa vezme aktuálny dátový rámec a vezmú sa z neho všetky neohodnotené položky referenčného užívateľa. (Ten bol pridaný vtedy, keď prvýkrát niečo ohodnotil a každé nasledovné hodnotenie sa pridávalo už iba jemu.)

2. Následne sa pridá k dátovej sade s neohodnotenými položkami stĺpec s predikovanými hodnoteniami. Do tohto stĺpca sa postupne pridávajú hodnotenia, ktoré sú vytvárané pomocou metódy daného modelu s názvom *predict*.

[5]Pri väčších dátových sadách je predikovanie dát zdĺhavé, a preto je výhodné ich uložiť a pri rovnakých dátach zbytočne nepredikovať znova.

3. Z dátovej sady s predikciami sa vytvoria štatistiky, ktoré sa uložia do slovníka. Ukladajú sa tam tieto dáta: top 10 odporučení (jeho hodnoty sú názov danej položky, predikovaná hodnota a ak je zadaný žáner, potom aj jeho názov). Ďalej je uložená štatistika o počte predikovaných hodnôt v rozmedzí hodnotenia (jej hodnoty sú škála hodnotenia a počet predikcií v nej obsiahnutej). Slovník sa napokon uloží do pamäte (pod názvom, ktorý je ako argument metódy) pomocou funkcie *pickle*. Tieto dáta sú aj výstupom metódy.

Po získaní dát z tejto metódy sa zobrazia štatistiky predikovaných odporučení. Vykreslí sa tabuľka s top 10 odporučeniami. Ďalej sa zobrazí tabuľka s počtom predikovaných hodnôt na škálach hodnotenní. Následne je vyobrazený stĺpcový graf reprezentujúci danú tabuľku. Ak je zadaný aj žáner, tak sa vykreslia grafy s počtom predikovaných hodnotení pre žáner v danom rozmedzí a aj percentuálne zastúpenie počtu hodnotení každého žánra naprieč škálou. Tieto grafy sa menia na základe kliknutia užívateľa na stĺpec v grafe, ktorý zobrazuje počet hodnotení v konkrétnom rozmedzí. Výstup môžeme vidieť na obrázku 4.3.



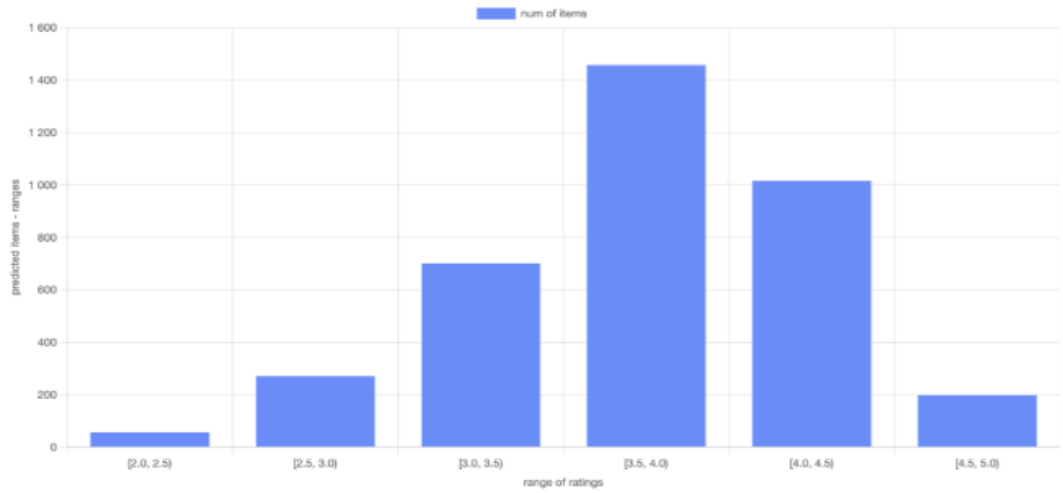
Obr. 4.2: Ukážka zadávania hodnotení v aplikácií.

4.3.4 Kopírovanie hodnotení

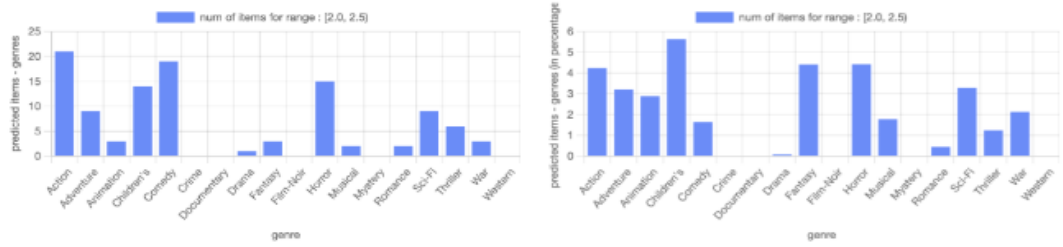
Funguje podobne ako hodnotenie položiek. Rozdiel je v tom, že pri výbere odkopírovania hodnotení sa na začiatku užívateľovi zobrazí formulár, v ktorom má vložiť id človeka, ktorého hodnotenia chce odkopírovať. Užívateľ má na výber tlačidlá s náhodne vybranými identifikátormi ľudí, ktorí hodnotili 29 až 89 položiek. Po kliknutí na niektoré z týchto tlačidiel sa do formulára vyplní identifikačné číslo zvoleného užívateľa. Následne sa zobrazí okno s položkami, ktoré boli ohodnotené daným užívateľom. Kopírovanie funguje rovnako ako klasické zadávanie hodnotení, len sa namiesto žánru zobrazí hodnotenie. Taktiež je možnosť kliknúť na tlačidlo, ktoré zobrazuje vlastné aktuálne hodnotenie. Po kliknutí naň sa odkopíruje hodnotenie. Následne sa zašlú hodnotenia a užívateľovi sa zobrazí okno s tabuľkou so stĺpcami: meno položky (zobrazia sa iba položky, ktoré sa neohodnotili), hodnotenie vybraného id, predikované hodnotenie a RMSE predikcie.

#	name	prediction	genres
1	Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)	4.935571601916348	Action|Drama
2	Shawshank Redemption, The (1994)	4.925198006120964	Drama
3	Sanjuro (1962)	4.910478186610473	Action|Adventure
4	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	4.901539152074782	Film-Noir
5	Usual Suspects, The (1995)	4.8990553239945696	Crime|Thriller
6	Godfather, The (1972)	4.89796730014007	Action|Crime|Drama
7	Close Shave, A (1995)	4.890571242315474	Animation|Comedy|Thriller
8	Wrong Trousers, The (1993)	4.88961896288659	Animation|Comedy
9	Schindler's List (1993)	4.883154417132196	Drama|War
10	Rear Window (1954)	4.866272918531939	Mystery|Thriller

range	number of items
[2.0, 2.5)	58
[2.5, 3.0)	272
[3.0, 3.5)	701
[3.5, 4.0)	1458
[4.0, 4.5)	1016
[4.5, 5.0)	201



in this graph we can see how many items rated how many users and how many users rated item



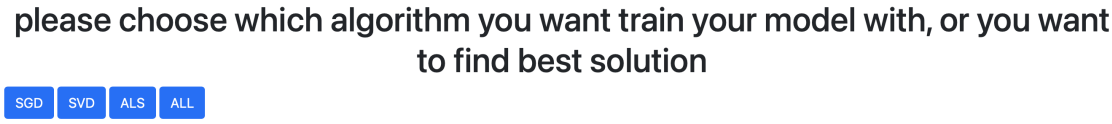
Obr. 4.3: Ukážka štatistiky vytvorenej pomocou aplikácie použitej v tejto práci.

4.3.5 Vytváranie modelu

Za účelom vytvorenia modelu, pomocou niektorého z algoritmov, slúžia metódy implementované v triede *Dataset* a to : *train_model_svd*, *train_model_als* a *train_model_sgd*. V týchto metódach sa volajú funkcie pre tvorbu modelov pomocou požadovaných algoritmov z knižnice *surprise*. Všetky tieto metódy berú ako vstupné parametre: koeficient učenia a počet krokov, ktoré sa majú vykonať. Tieto funkcie môžu byť volané buď z užívateľského rozhrania, kde užívateľ zadá aký algoritmus chce použiť pre vytvorenie modelu, a sám nastaviť parametre s akými chce vytrénovať model. Alebo sa funkcia volá pri zadávaní hodnotení, ak uživateľ zadá tlačidlo s algoritmom. Vtedy sa táto metóda volá s parametrom učenia, sa ktorý má hodnotu 0.005 a počtom krokov 10. Tieto hodnoty boli usúdené z experimentov. Dáta sa trénujú vždy z aktuálneho rámca v inštancií triedy *Dataset*. Po vytrénovaní sa uloží model daného algoritmu do inštancie triedy *datasets* (ak je už vytrénovaný iný model daného algoritmu, tak sa jednoducho prepíše).

4.3.6 Hľadanie hyperparametrov

Hľadanie hyperparametrov v tejto aplikácií slúži na nájdenie (odhadnutie) najlepších hyperparametrov, ktoré sa dajú zadať pri trénovaní modelu. Jedná sa o hyperparameter učenia sa a počet evaluácií pri jeho vytváraní. V užívateľskom rozhraní je na výber hľadanie optimálnych hyperparametrov, buď iba pre jeden konkrétny algoritmus (SGD, ALS, SVD) alebo najlepšia kombinácia zo všetkých algoritmov. Ukážka výberu pre hľadanie hyperparametrov je na obrázku 4.4



Obr. 4.4: Ukážka výberu algoritmu pre hľadanie hyperparametrov.

Implementácia je v metóde *find_hyperparams*, ktorá berie ako parametre: textovú skratku algoritmu, pre ktorý chceme ladiť parametre (sgd, als, svd ak chceme pre všetky tak použijeme all) a škálu hodnotení od a do. Škála hodnotení sa získava tak, že zavoláme v dátovom rámci na stĺpci hodnotení metódu *min*, pre zistenie hodnotení od a *max* pre zistenie hodnotení do. Funkcia pre nájdenie hyperparametrov funguje tak, že na začiatku sa rozdelí aktuálny dátový rámec v danej inštancií na testovaciu a trénovaciu. Následne sa vytvorí inštancia triedy *BayesianOptimization* z knižnice *BayesianOptimization*. Ako atribúty tejto funkcie je potrebné nastaviť parametre, ktorými sú: black box funkcia, tá sa bude opätovne volať v iteráciách pri hľadaní, ďalej parametre, ktoré sa budú ladiť a počet iterácií potrebných pre vykonanie. Úlohou black box funcie je vytrénovať model s parametrami, ktoré chceme ladiť (závisí na algoritme) a jej výstupom je RMSE daného modelu. Predvolený počet iterácií pre *BayesianOptimization* je 10 (proces hľadania je veľmi zdĺhavý a viac iterácií by to zbytočne predĺžovalo). Výsledkom funkcie pre nájdenie hyperparametrov je slovník daného algoritmu, v ktorom sú hodnoty parametrov, čas, RMSE z každej iterácie. Tento slovník sa uloží interne pomocou *pkl* pre spätné nahliadnutie k výsledkom. Po nájdení hyperparametrov sa užívateľovi zobrazí okno s tabuľkou hodnôt najlepšej iterácie a grafy, kde je v prvom vykreslená závislosť parametru učenia sa voči RMSE, v druhom závislosť počtu

krokov voči RMSE a v treťom závislosť času voči RMSE (ak bola vybratá možnosť hľadania pre všetky algoritmy, tak sa v rovnakých grafoch vykreslia závislosti všetkých algortimov rozlíšené farbami).

4.3.7 Vymazanie hodnotení

V aplikácií je možnosť vymazania hodnotení referenčného užívateľa. Logika vymazania hodnotení funguje tak, že sa atribút *to_train* inštancie triedy *Dataset*, v ktorom je uložená interakčná matica a hodnotenia užívateľa, prepíše na hodnoty atribútu *name_and_id_item*, kde je uložená čisto len interakčná matica (bez hodnotení daného užívateľa).

4.3.8 Funkcia aplikácie

Cieľom aplikácie je spracovať užívateľom zadanú dátovú sadu a taktiež mu umožniť vytvoriť predikciu odporučenia pomocou 3 metód kolaboratívneho filtorvania, a to ALS, SGD a SVD. Užívateľ môže zadať rôzne dátové sady a v priebehu práce s aplikáciou ich môže ľubovoľne meniť. Štatistiky o predikovaných odporučeniach a dátových sadách si môže spätne prezerať a analyzovať získané dáta.

Kapitola 5

Experimenty

Po implementácií sa aplikácia mohla použiť na experimenty s algoritmami, ktoré využíva a taktiež s vybranými dátovými sadami.

5.1 Dátové sady a ich predspracovanie

Nosnou časťou tejto práce bolo vybrať si správne dátové sady. Všeobecne dátová sada pre odporúčacie systémy je interakčná matica[1]. To či okrem interakčnej matice obsahuje dátová sada aj iné informácie, ako metadáta o užívateľoch, alebo položkách, explicitné resp. implicitné dáta záleží na konkrétnom algoritme, ktorý sa používa na odporúčanie položiek.

Táto práca je zameraná na 3 algoritmy a to: stochastické gradientné klesanie (SGD), rozloženie podľa singulárnych hodnôt (SVD), alternovanie najmenších štvorcov (ALS). Tieto algoritmy patria do skupiny metód kolaboratívneho filtrovania, ktoré vyžadujú model, a teda potrebujú iba interakčnú maticu. Cieľom pri hľadaní dátových sád boli 2 kritéria a to :

- Veľkosť dátovej sady.
- Hustota hodnotení dátovej sady.

Na základe týchto 2 zreteľov sa následne hľadali dátové sady, podľa toho, aký majú vplyv na daný algoritmus atď. Boli vybrané tieto dátové sady:

MovieLens 1M Táto dátová sada pochádza zo zdroja [1]. Obsahuje hodnotenia užívateľov na filmy. Pozostáva z troch súborov, kde prvý obsahuje interakčnú maticu, druhý obsahuje metadáta o položkách a tretí metadáta o užívateľoch. V tejto aplikácií sú využité iba dva z nich a to interakčná matica a metadáta o položkách. Ako je zjavné z názvu, táto dátová sada obsahuje niečo viac ako milión hodnotení.

Aby vytvorená aplikácia dokázala spracovať dáta obsiahnuté v týchto súboroch, bolo predom nutné urobiť pár zmien. Prvá zmena sa týkala zmeny typu súborov. Ich pôvodný typ bol dat. Avšak aplikácia dokáže spravovať iba súbory typu buď csv, alebo json. Zmena bola vykonaná pomocou knižnice Pandas, kde sa z daného súboru vytvoril dátový rámec, ktorý sa následne uložil ako csv súbor. Toto súviselo aj s pridaním hlavičky s názvami jednotlivých stĺpcov, ktoré bolo nutné pridať do daného dátového rámca.

[1] Pod pojmom interakčná matica v súbore je myslený taký súbor, ktorý pozostáva zo stĺpcov pre užívateľa, položky a hodnotenia. Samozrejme môže obsahovať aj iné stĺpce ako žáner a názov položky.

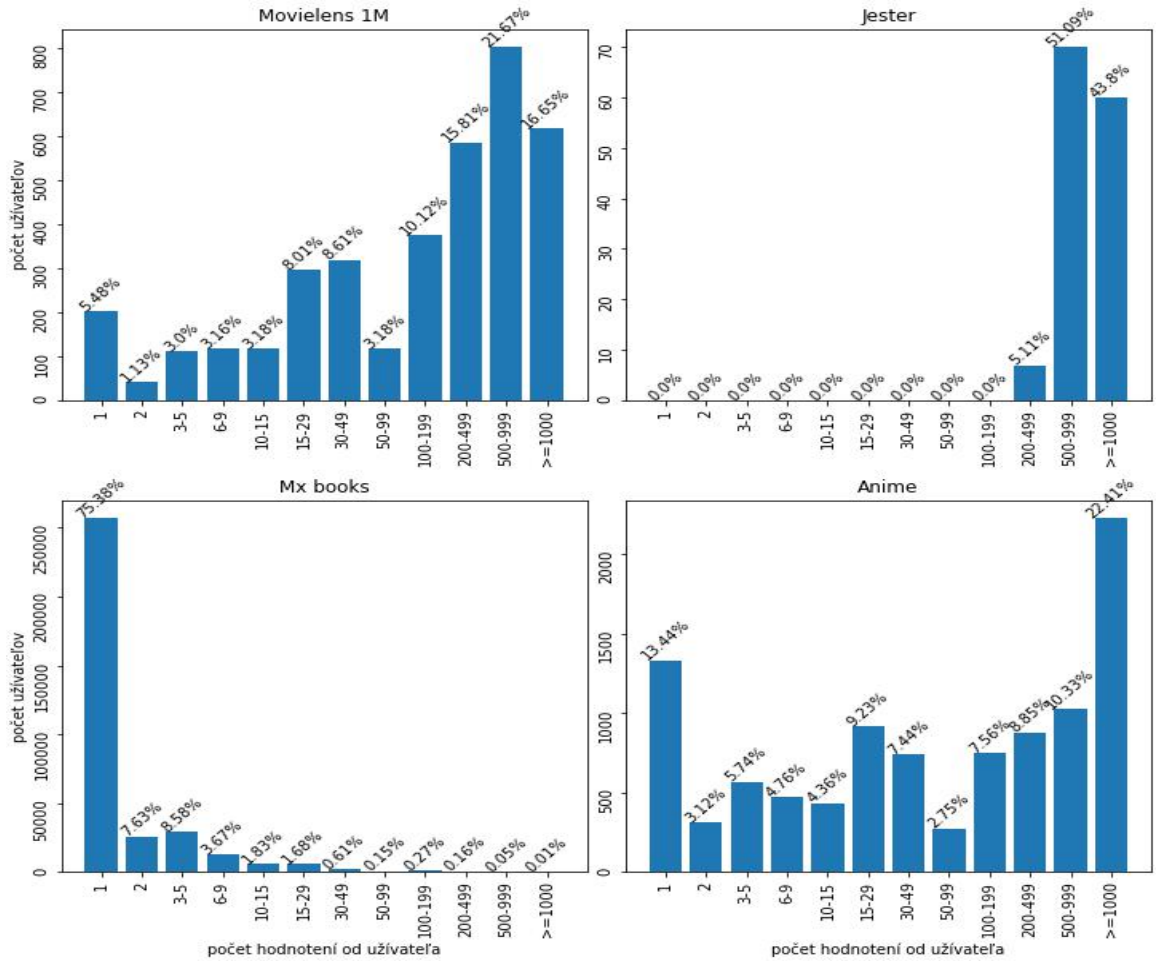
Anime databáza hodnotení Je prevzatá zo stránky kaggle[2]. Obsahuje hodnotenia užívateľov na rôzne anime postavy. Pozostáva z jedného súboru, ktorý obsahuje interakčnú maticu, žáner a aj názov položky. Toto je výhoda, pretože aplikácia nestráca čas so spájaním dátových rámcov z rôznych súborov.

Pre túto dátovú sadu neboli potrebné žiadne úpravy.

Yester Yester je dátová sada prevzatá z [14]. Obsahuje hodnotenia užívateľov na vtipy.

V súbore boli potrebné úpravy, ako zmeniť formát, pretože formát súborov je xls. Zmena formátu bola vykonaná podobne ako pri MovieLens 1M. Ako dalšie sa musel upraviť text vtipov, ktorý bol v inom súbore, agregovať k identifikátoru vtipu. Taktiež bolo potrebné vymazať niektoré hodnotenia, keďže boli irelevantné. Napríklad číslo bolo mimo škály hodnotenia.

Book-Crossing dátová sada Book-Crossing Dataset je použitý zo zdroja [29]. Obsahuje hodnotenia užívateľov na knihy. Okrem zmeny typu súborov, ktorú bolo potrebné vykonať, sa museli vykonať iba zmeny ako v MovieLens 1M.



Obr. 5.1: Graf znázorňujúci počet užívateľov, majúcich daný počet hodnotení.

[2]https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database

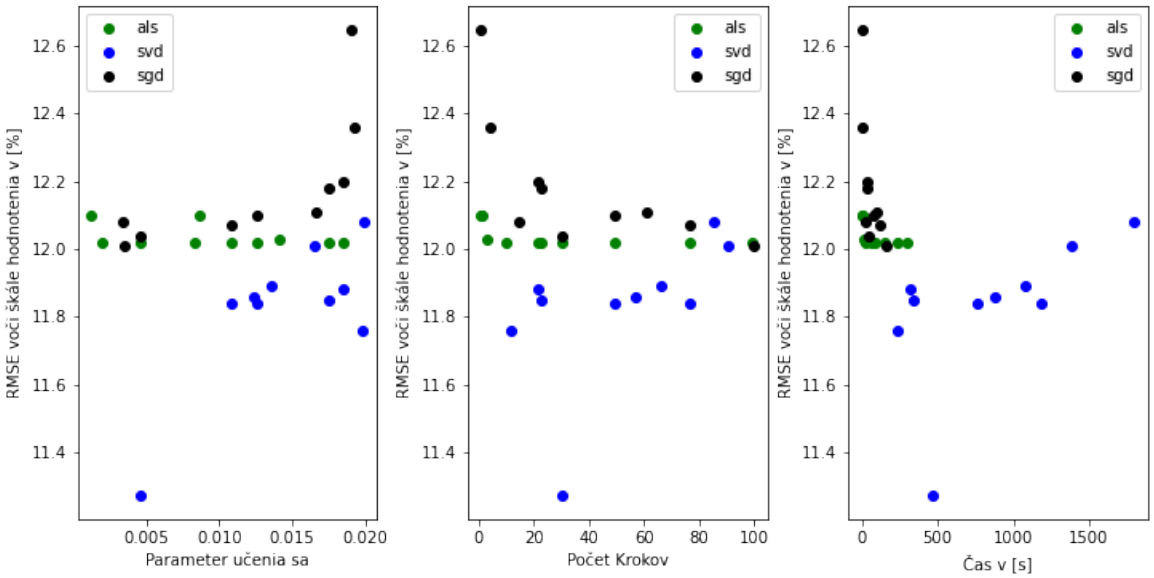
	počet hodnotení	hustota	užívateľov užívateľov	počet položiek	škála hodnotenia
MovieLens 1M	1 000 209	4.47%	6040	3952	1 - 5
Anime databáza hodnotení	6 337 241	0.92%	73 516	12 294	1 - 10
Yester	114 181	10.83%	7 699	158	-10 - 10
Book-Crossing dátová sada	1 149 780	0.000015%	278 858	271 379	1 - 10

Tabuľka 5.1: Porovnanie dátových sád s ich veľkosťami a hustotami.

5.2 Hľadanie hyperparametrov

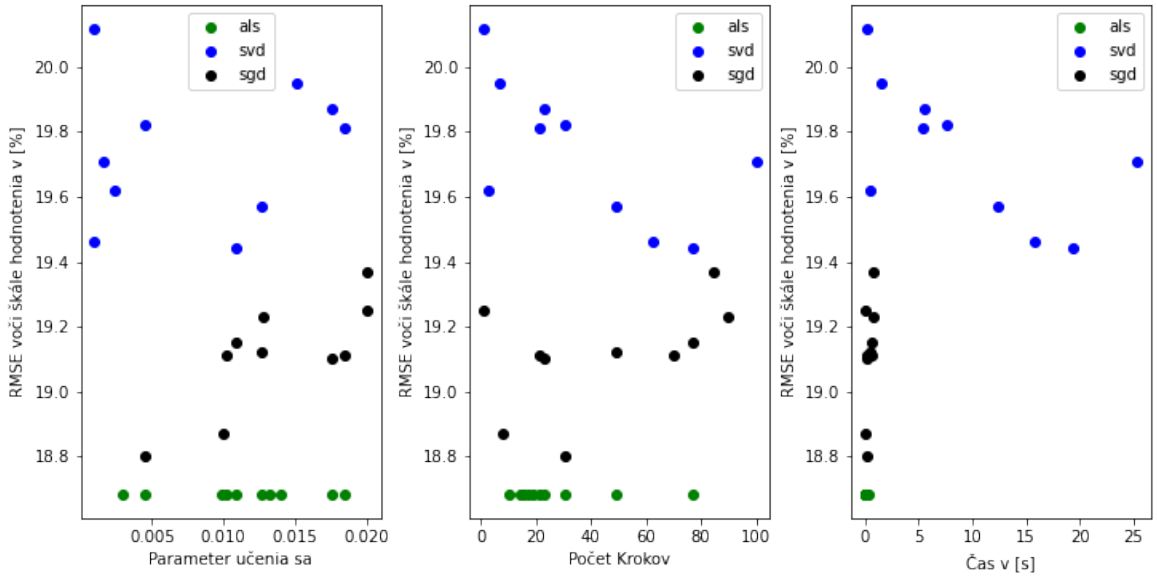
Nemenej dôležitá vec pri trénovaní modelu je nájsť tie správne hyperparametre. Pri tomto hľadaní sa v aplikácií využila knižnica *BayesOpt*, jej použitie v aplikácií je podrobnejšie popísané v 4.3.6.

Ako s prvou sadou sa experimentovalo s Anime databáza hodnotení. Najlepšie výsledky pre trénovanie pri tejto sade mal SVD algoritmus. Je vidieť, že hustota má obrovský vplyv na RMSE algoritmov. Ako pozorujeme na obrázku 5.2 v druhom grafe, k najlepšiemu výsledku sa dostal v 35 kroku. Na treťom grafe badáme, že SVD algoritmus je extrémne pomalý. Keď porovnáme niečo okolo 100 iterácií ALS a SGD oproti 90 iteráciám SVD vidíme, že SVD je časovo zložitejšie. Zaujímavosťou je, že parameter učenia sa má najlepšie výsledky pri malej hodnote ako 0.0005. Taktiež je vidieť, že pri vyšších dochádza buď k overfittingu, alebo underfittingu.



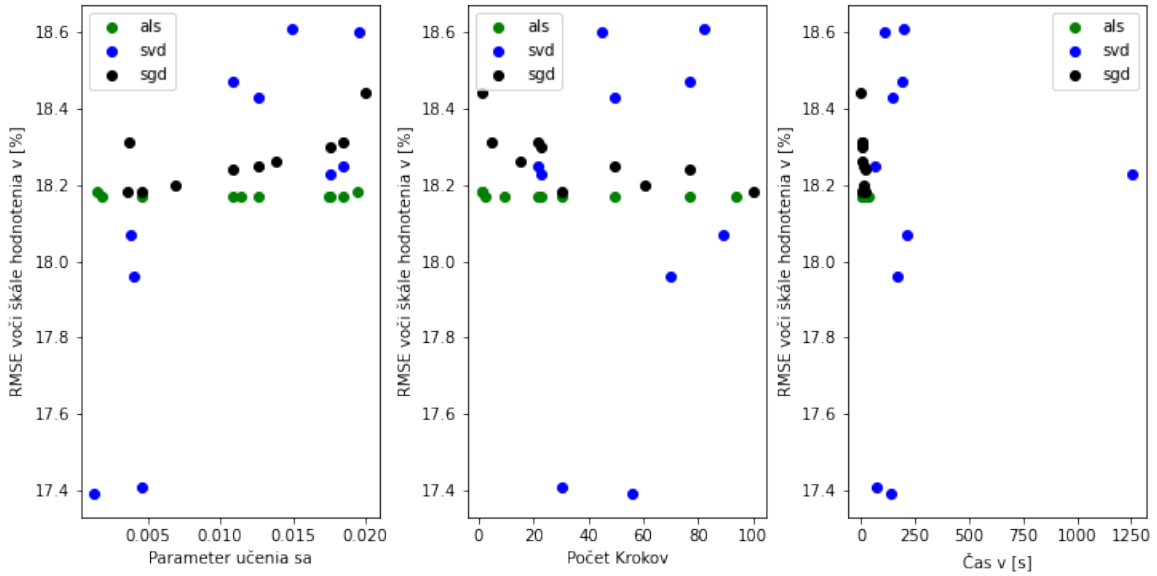
Obr. 5.2: Rozptylový graf znázorňujúci skúmané parametre voči RMSE na Anime databáza hodnotení.

Ako druhá sada bola skúmaná Yester. Als bol algoritmus, ktorý mal najlepšie výsledky. Je zrejmé, že ani počet evaluácií a ani počet krokov nemá nejaký signifikantný dopad. Na oboch grafoch (1. a 2. zľava), ktoré sú znázornené na obrázku 5.3, sú body ALS algoritmu takmer lineárne



Obr. 5.3: Rozptylový graf znázorňujúci skúmané parametre voči RMSE na dátovej sade Yester.

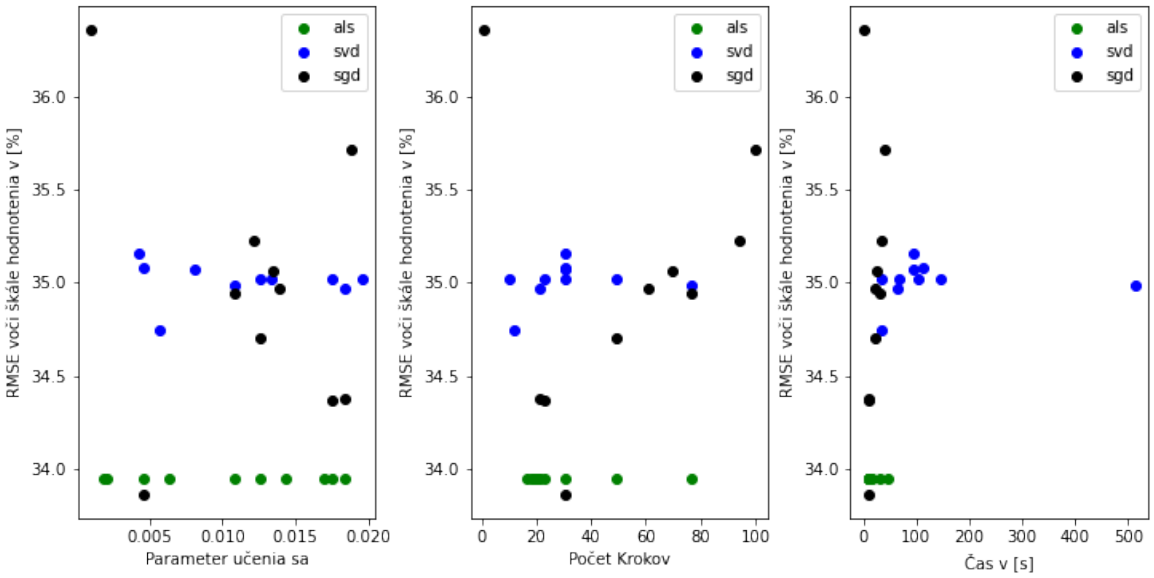
Ďalej sa skúmali hyperparametre MovieLens 1M sady. Tu bol opäť najlepším algoritmom SVD. Tak ako v prvom pokuse, vieme posúdiť rovnaké veci. A teda, parameter učenia sa je najmenší pre optimálny výsledok. Tiež nie je potrebné vykonať príliš veľa evaluácií. V danom prípade, ich stačí 58, dokonca pri 35 krokoch nie je veľký rozdiel v RMSE (niečo cez 0,1). Výsledky sú znázornené na obrázku 5.4.



Obr. 5.4: Rozptylový graf znázorňujúci skúmané parametre voči RMSE na dátovej sade MovieLens 1M.

Posledná skúmaná dátová sada bola Book-Crossing dátová sada. Najlepším algoritmom bol SGD. Táto metóda je veľmi rýchla, čo pozorujeme aj v predošlých experimentoch. Z

grafov, ktoré sú vyobrazené na obrázku 5.5 je vidieť, že optimálne hodnoty parametrov sú ideálne. Keď je hodnota nastavenia parametrov menšia, graf má tendenciu klesať. Naopak ak ju zvýšime, tak má graf tendenciu rásť. Je možné usúdiť, že pomocou bayesvského hľadania hyperparametrov sa našlo lokálne minimum.



Obr. 5.5: Rozptylový graf znázorňujúci skúmané parametre voči RMSE na dátovej sade Book-Crossing dátová sada.

	veľkosť	hustota	najlepšia metóda	parameter učenia sa	kroky	RMSE percentuálne voči škále hodnotenia
MovieLens 1M	1 000 209	4.47%	SVD	0.0013	56	17.2%
Anime	6 337 241	0.92%	SVD	0.005	30	11.3%
Yester	114 181	10.83%	ALS	0.01	77	18.7%
Book-Crossing	1 149 780	0.000015%	SGD	0.005	30	33.8%

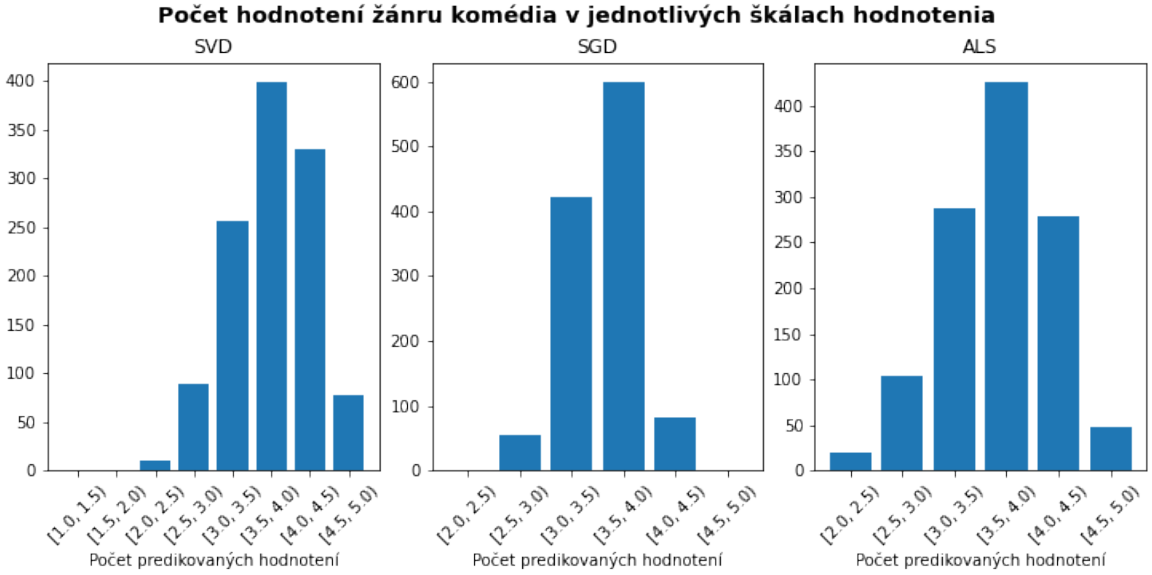
Tabuľka 5.2: Porovnanie výsledkov najlepších metód a kombinácie paramterov s ich veľkosťami a hustotami.

Ako je vidieť v tabuľke 5.2, dá sa vo všeobecnosti povedať, že kombinácia 30 krokov a hodnota 0.005 pre parameter učenia sa, je v týchto pokusoch dominujúca. Taktiež je z grafov zrejmé, že výsledky algoritmu ALS sú od hodnoty 0.004, takmer konštantné. Znamená to, že počet evaluácií a hodnota parametru učenia sa nemajú na túto metódu veľký vplyv. SGD algoritmus je extrémne rýchly. SVD metóda je spoľahlivá, avšak veľmi zdĺhavá na výpočet. Preto je na zváženie jej použitie oproti napríklad ALS, ktoré podáva kvalitné výsledky za zlomok času. Ak je dátová sada príliš riedka, tieto algoritmy nedokážu dobre predikovať, ako vidíme pri Book-Crossing sade (Aj keď pravdou je, že táto sada ma extrémne nízku hustotu).

5.3 Predikcia hodnotení

Pri vytváraní predikcií sa testovali dve veci, a to: samotná predikcia hodnotenia na základe položiek, ktoré užívateľ vloží (testovalo sa, či dokážu algoritmy detekovať zámerné hodnotenie jedného žánru). V druhom experimente sa skúmalo, či nejaký algoritmus dokáže zaznamenať zámerné kopírovanie hodnotení iného užívateľa. Hodnoty hyperparametrov pre trénovanie modelu boli použité z merania spomínaného vyššie.

1. V tomto experimente boli použité dátové sady anime 5.1 , filmová databáza Movielens 5.1 a databáza hodnotenia kníh 5.1 (pretože tieto sady obsahujú aj žaner položky). Ako prvá sa testovala filmová databáza. Ohodnotilo sa dvadsať náhodných filmov so žánrom komédia. Všetky boli ohodnotené piatimi, čo predstavuje najlepšie možné hodnotenie. Ako je možné vidieť na grafoch 5.6 algoritmus SVD vedel najlepšie zdetekovať hodnotenie položky jedného žánru a predikoval najviac hodnotení na škále 5 pre žáner komédia.

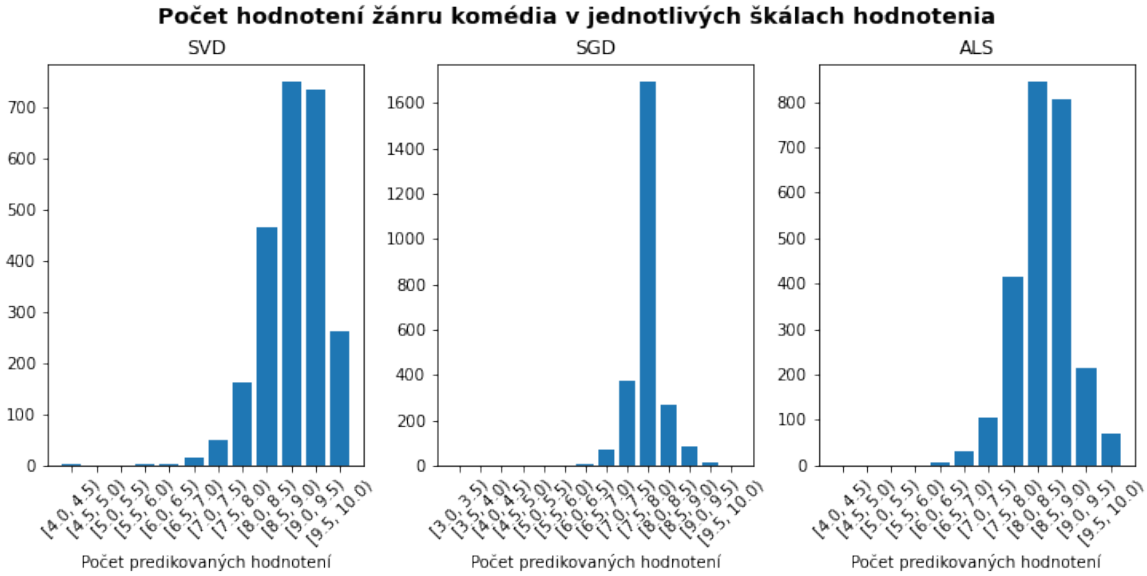


Obr. 5.6: graf znázorňujúci predikcie hodnôt pre žáner komédia na sade MovieLens 1M.

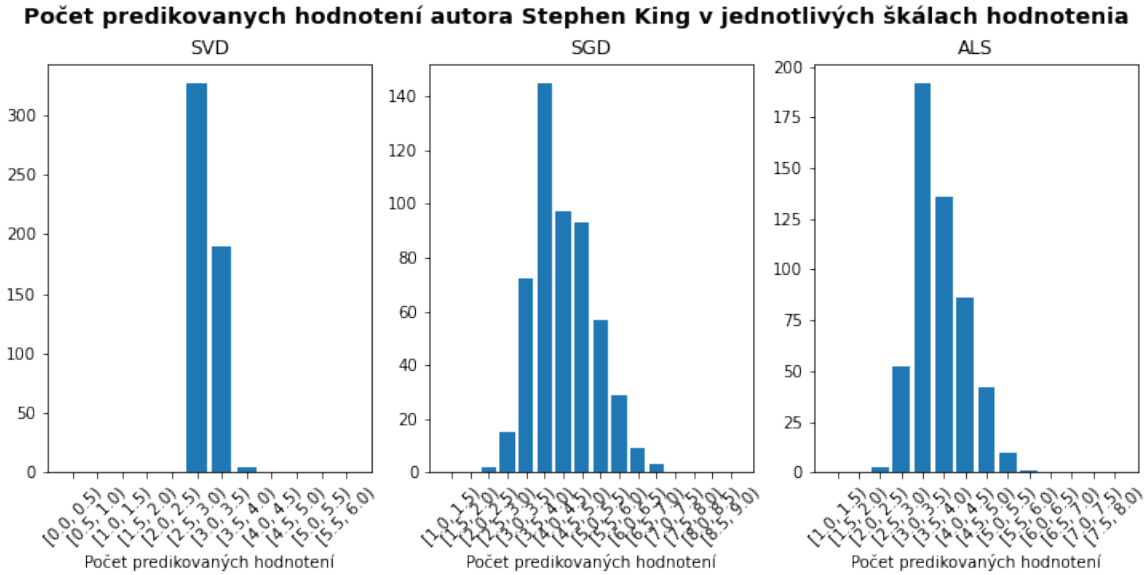
Identický test bol vykonaný na anime dátovú sadu, kde bolo ohodnotených 20 hodnotení taktiež žánru komédia. Ohodnotené boli hodnotou 10, čiže najväčším možným. Na obrázku 5.7 badáme, že najlepšie predikoval hodnotenia SVD algoritmus.

Tretí pokus bol experimentovaný na dátovej sade MX Books. Kde sa nehodnotil žáner, ale autor Stephen King. Taktiež sa hodnotilo 20 jeho diel s maximálnym možným ohodnotením 10. SGD algoritmus vedel najlepšie zdetekovať dané hodnotenia a predikoval hodnotenia najbližšie k maximu pre daného autora. Výsledky tohto pokusu môžeme vidieť na grafe 5.8

Ako je zjavné z pokusov, potvrdili sa výsledky hľadania hyperparametrov, ktoré sú zhrnuté v tabuľke 5.2. Tiez je zrejmé, že takéto výsledky by nebolo možné zosumarizovať, ak by sme neohodnotili viac položiek, pretože tieto algoritmy trpia chladným štartom.



Obr. 5.7: graf znázorňujúci predikcie hodnôt pre žáner komédia na sade Anime databáza hodnotení



Obr. 5.8: graf znázorňujúci predikcie hodnôt pre autora Stephen Kinga na sade Book-Crossing dátová sada.

2. Ďalším experimentom bolo skopírovanie hodnotení niektorého z užívateľov (pár sa vynechalo na otestovanie). Pre tento experiment boli použité sady s hodnoteniami na filmy a na anime postavy. Kopírovali sa užívatelia, ktorí ohodnotili 90 položiek. V experimente sa ohodnotilo 70 položiek a 20 sa zámerne vynechalo, následne sa vypočítala chybovosť v percentách tak, že sa spočítala RMSE každého algoritmu a tá

sa predelila počtom celých čísel v škále hodnotenia. Výsledky sú zhrnuté v tabuľke 5.3. Pri dátovej sade s filmami boli všetky algoritmy rovnako úspešné. V prípade dátovej sady anime mal ALS problémy, výrazne zaostal za SGD a SVD algoritmami. Ďalej je z tabuľky zrejmé, že hustota sád ovlplyvnila ich presnosť. Pri hodnoteniach filmov je priemerná chyba 22%, narozdiel od anime, kde je chyba 16%. Pri tomto experimente je, ale potreba zvážiť, že nie každý užívateľ hodnotí vždy všetko čo sa mu páči kladne.

	movies dataset	anime dataset
ALS	22.0%	20.3%
SGD	22.29%	14.2%
SVD	22.25%	14.1%

Tabuľka 5.3: tabuľka znázorňujúca chybovosť v percentách jednotlivých algoritmoch na dátových sadách pre hodnotenie filmov a hodnotení anime. Pri kopírovaní užívateľa, ktorý hodnotil 90 položiek.

5.4 Zhrnutie experimentov

Ako je pochopiteľné z vykonaných experimentov, každý zo skúmaných algoritmov má lepšie výsledky v iných situáciách. SGD algoritmus je veľmi rýchly a dokonca mal najlepšie výsledky RMSE pri sade s hodoteniami kníh. Dá sa povedať, že algoritmus ALS je univerzálny, to znamená, že je pomerne rýchly, dosahuje kvalitné výsledky, ktoré boli pri rôznych kombináciach hyperparametrov na rôznych dátových sadách lineárne. SVD metóda mala kvalitné výsledky pri sadách s vysokou hustotou. Slabá stránka tejto metódy je, že je extrémne pomalá.

Pri výbere algoritmu je potrebné hľadať správny algoritmus pre vykonanie predikcií. ALS algoritmus je univerzálnym a dosahuje kvalitné výsledky za rôznych okolností, a preto ho považujem za najlepší zo skúmaných.

Kapitola 6

Záver

Táto bakalárska práca sa venovala opisu a porovnaniu metód pre vytvorenie odporúčacích systémov. Cieľom bolo vytvoriť aplikáciu, ktorá umožní užívateľovi spracovať dátovú sadu a vytvoriť predikciu hodnotení pomocou niektorých z algoritmov. Taktiež poskytnúť možnosť porovnať výsledky týchto algoritmov pomocou rôznych štatistík. Pozornosť bola upriamená na algoritmy kolaboratívneho filtorovania, ktoré vyžadujú model.

V prvej časti tejto práce sú zhrnuté jednotlivé algoritmy odporúčacích systémov. Prevažne sa rozoberajú metódy kolaboratívneho filtrovania. Okrem algoritmov sú tu zhrnuté metriky pre meranie presností predikcií, prístupy k rozdeleniu dátovej sady pre testovanie a trénovanie. Tatiež sú spomenuté problémy, ktoré vznikajú pri odporúčacích systémoch, princíp pre hľadanie optimálnej kombinácie hyperparamterov a nakoniec existujúce implementácie týchto metód v jazyku Python.

V druhej časti je popísaný návrh a implementácia výslednej aplikácie, výber dátových sád a experimenty. Experimenty boli rozdelené do dvoch častí, kde v prvej časti sa porovnávali kombinácie dátových sád s algoritmami pomocou výsledkov hľadania hyperparametrov. V druhej časti sa testovali vytvorené predikcie hodnotení.

Každý algoritmus má svoje výhody, no taktiež aj slabé stránky. Preto je prácu možné rozšíriť o hybridné prístupy, ktoré využívajú kladné časti z použitých metód a eliminujú tie záporné. Ďalej je prácu možné prepísať do programovacieho jazyka R, ktorý je pre prácu s dátami vytvorený.

Literatúra

[1] Harper, F. M. a Konstan, J. A. *The MovieLens Datasets: History and Context.* ACM Trans. Interact. Intell. Syst., 2015.

[2] Zhou, Y., Wilkinson, D., Schreiber, R. a Pan, R. *Large-Scale Parallel Collaborative Filtering for the Netflix Prize.* 337-348 s. 2008. ISBN 978-3-540-68865-5.

[3] Aberger, C. R. a caberger. *Recommender : An Analysis of Collaborative Filtering Techniques.* 2014.

[4] Hu, Y., Koren, Y. a Volinsky, C. *Collaborative Filtering for Implicit Feedback Datasets.* IEE, 2008.

[5] Koren, Y., Bell, R. a Volinsky, C. *Matrix Factorization Techniques for Recommender Systems.* Computer, 2009.

[6] Han, J., Kamber, M. a Pei, J. *Data Mining.* 3, 77-78 s. 2012. ISBN 978-0-12-381479-1.

[7] Ryngksai, I. a Chameikho, L. *Recommender Systems: Types of Filtering Techniques* [online]. 2014 [cit. 2021-11-26]. Dostupné z: https://www.ijert.org/research/recommender-systems-types-of-filtering-techniques-IJERTV3IS110197.pdf.

[8] Zornoza, J. *Distance Metrics for Machine Learning* [online]. 2020 [cit. 2022-03-31]. Dostupné z: https://aigents.co/data-science-blog/publication/distance-metrics-for-machine-learningf.

[9] Fkih, F. *Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison.* Journal of King Saud University - Computer and Information Sciences, 2021. ISSN 1319-1578.

[10] Fayyaz, Z., Ebrahimian, M., Nawara, D., Ibrahim, A. a Kashef, R. *Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities.* Applied Sciences, 2020 [cit. 2021-11-25]. ISSN 2076-3417.

[11] Hug, N. Surprise: A Python library for recommender systems. *Journal of Open Source Software.* The Open Journal. 2020, zv. 5, č. 52, s. 2174. DOI: 10.21105/joss.02174. Dostupné z: https://doi.org/10.21105/joss.02174.

[12] Koehrsen, W. *A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning* [online]. 2018 [cit. 2022-03-23]. Dostupné z:

https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f.

[13] ZACH. *A Simple Explanation of the Jaccard Similarity Index* [online]. 2020 [cit. 2021-11-26]. Dostupné z: https://www.statology.org/jaccard-similarity.

[14] Goldberg, K., Roeder, T., Gupta, D. a Perkins, C. *Eigentaste: A Constant Time Collaborative Filtering Algorithm.* Information Retrieval, 2001.

[15] Wu, J. *Knowledge-Based Recommender Systems: An Overview* [online]. 2019 [cit. 2021-11-26]. Dostupné z: https://medium.com/@jwu2/knowledge-based-recommender-systems-an-overview-536b63721dba.

[16] chiang, J. *7 Types of Hybrid Recommendation System* [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8.

[17] Isinkaye, F., Folajimi, Y. a Ojokoh, B. *Recommendation systems: Principles, methods and evaluation.* 261-273 s. Egyptian Informatics Journal, 2015. ISBN 1110-8665.

[18] Rackaitis, T. *Evaluating Recommender Systems: Root Means Squared Error or Mean Absolute Error?* [online]. 2019 [cit. 2022-03-23]. Dostupné z: https://towardsdatascience.com/evaluating-recommender-systems-root-means-squared-error-or-mean-absolute-error-1744abc2beac.

[19] Kwiatkowski, R. *Gradient Descent Algorithm — a deep dive* [online]. 2021 [cit. 2021-11-27]. Dostupné z: https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21.

[20] Rocca, B. *Introduction to recommender systems ?* [online]. 2019 [cit. 2021-11-10]. Dostupné z: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada.

[21] Mishra, N., Chaturvedi, S., Vij, A. a Tripathi, S. *Research Problems in Recommender systems.* 7-8 s. Journal of Physics: Conference Series, 2021.

[22] Jure Leskovec, A. R. a Ullman, J. D. *Mining of Massive Datasets.* 2, 333 s. Cambridge University Press, 2014. ISBN 9781139924801.

[23] Meng, Z., McCreadie, R., Macdonald, C. a Ounis, I. *Exploring Data Splitting Strategies for the Evaluation of Recommendation Models.* Fourteenth ACM Conference on Recommender Systems, 2020.

[24] Suriati, S. *Weighted hybrid technique for recommender system* [online]. 2017 [cit. 2022-03-25]. Dostupné z: https://iopscience.iop.org/article/10.1088/1742-6596/930/1/012050.

[25] Wang, W. *Bayesian Optimization Concept Explained in Layman Terms* [online]. 2020 [cit. 2022-03-31]. Dostupné z: https://towardsdatascience.com/bayesian-optimization-concept-explained-in-layman-terms-1d2bcdeaf12f.

[26] KUMAR, D. V. *Singular Value Decomposition (SVD) & Its Application In Recommender System* [online]. 2020 [cit. 2022-03-22]. Dostupné z: https://analyticsindiamag.com/singular-value-decomposition-svd-application-recommender-system/.

[27] DWIVEDI, R. *What Are Recommendation Systems in Machine Learning?* [online]. 2021 [cit. 2022-04-15]. Dostupné z: https://www.analyticssteps.com/blogs/what-are-recommendation-systems-machine-learning.

[28] YASAR, K. *Recommender Systems: What Long-Tail tells ?* [online]. 2018 [cit. 2021-11-25]. Dostupné z: https://medium.com/@kyasar.mail/recommender-systems-what-long-tail-tells-91680f10a5b2.

[29] ZIEGLER, CAI NICOLAS, CAI NICOLAS, SEAN, M., SEAN, M. et al. Improving recommendation lists through topic diversification. In:. Január 2005. DOI: 10.1145/1060745.1060754.

[30] ZIESEMER, A., MÜLLER, L., SILVEIRA a MILENE. Just Rate It! Gamification as Part of Recommendation. In:. Jún 2014. ISBN 978-3-319-07226-5.