FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Doporučovací systém pro volbu předmětu na základě podobnosti studentů
Student:	Vít Kalianko
Vedoucí:	Ing. Tomáš Nováček
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Když si studenti vybírají předměty, které by si mohli zapsat do dalšího semestru, musí procházet Anketu, ptát se spolužáků a mnoho dalšího. To může být velmi časově náročné a nezáživné. Cílem této práce je zanalyzovat možnosti doporučování a vytvořit systém, který má studentům sloužit jako rádce při výběru předmětů. Důraz je kladen na různé metody porovnávání studentů, přesnost výsledků těchto metod a výběr nejlepší z nich pro co nejpřesnější doporučení.

Postupujte v těchto krocích:
1) Analyzujte data o předmětech a studentech, která využívá a zpracovává KOS, Datový sklad, případně Anketa ČVUT.
2) Proveďte rešerši stávajících metod využívaných doporučovacími systémy a aktuálně nejvíce používaných technologií k jejich vytváření.
3) Na základě analýzy zvolte potřebná data pro vaši úlohu a vhodně je předzpracujte.
4) Navrhněte systém na doporučování předmětů dle výstupů z rešerše z bodu 1.
5) Na základě rešerše dále vyberte vhodnou metriku na testování doporučovacích systémů a využijte ji pro ověření správnosti vámi zvolených řešení.
6) Implementujte a otestujte několik metrik pro porovnání podobnosti studentů, alespoň jedna metrika musí být vaše vlastní.
7) Navrhněte a implementujte systém na doporučování předmětů s využitím metriky z bodu 6, která dosáhla nejlepších výsledků.
8) Dosažené výsledky a vhodnost svého řešení diskutujte.

Elektronicky schválil/a Ing. Magda Friedjungová, Ph.D. dne 20. ledna 2023 v Praze.

Bakalářská práce

DOPORUČOVACÍ SYSTÉM PRO VOLBU PŘEDMĚTU NA ZÁKLADĚ PODOBNOSTI STUDENTŮ

Vít Kalianko

Fakulta informačních technologií
Katedra aplikované matematiky
Vedoucí: Ing. Tomáš Nováček
11. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Můj dík patří Bc. Adamovi Makarovi a Ing. Michalu Valentovi, Ph.D., za poskytnutí potřebných dat pro práci. Dále bych chtěl poděkovat Ing. Vojtěchu Vančurovi za cenné rady při implementaci systému a nejlepšímu vedoucímu Ing. Tomášovi Nováčkovi, který mi byl podporou nejen při práci a vždy mi poradil, kterým směrem se dát.
Dále bych chtěl poděkovat rodině, všem svým přátelům, organizátorům Seznamováku FIT, studijní skupince Tangus a týmu Platformeláků TA ČR za psychickou podporu.
Nakonec bych chtěl poslat speciální největší dík svému tatínkovi, který mě vždy maximálně podporoval ve všem, co dělám, ale tuto práci si už nikdy nepřečte.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2023 .....

Abstrakt

Práce se zaměřuje na analýzu metod, kterými jsou tvořeny doporučovací systémy, speciálně pak popisuje metodu nejbližších sousedů. Definuje metriky pro vyhodnocování doporučovacích systémů. Následně obsahuje analýzu dat dostupných ze systémů na Fakultě informačních technologií ČVUT v Praze, která předzpracovává. Výsledkem je dataset obsahující intrakce studentů s předměty, na kterém pomocí metody nejbližších sousedů každému studentovi doporučujeme volitelné předměty, které si zapisovali podobní studenti. Podobní studenti jsou hledáni pomocí manhattanské, euklidovské a cosinové vzdálenosti. Tyto metriky jsou v práci upraveny tak, že při porovnání dvou studentů berou v potaz pouze předměty, které oba studenti měli skutečně zapsané. Tyto metriky vzdálenosti jsou v práci porovnány s originálními. Pomocí různých metrik je vyhodnocena úspěšnost systému s různými hyperparametry, ze kterých je jako nejlepší kombinace vyhodnocena cosinová vzdálenost s 20 sousedy.

Klíčová slova doporučování předmětů, metrika vzdálenosti, doporučovací systém, metoda nejbližších sousedů, vyhodnocování doporučovacích systémů, předměty FIT ČVUT

Abstract

The thesis focuses on the analysis of methods used in recommender systems and specifically describes the k-nearest neighbors method. It defines metrics for evaluating recommender systems. Subsequently, it includes an analysis of data available from the systems at the Faculty of Information Technology, CTU in Prague. These data is preprocessed into a dataset containing student interactions with courses, on which the k-nearest neighbors method is used to recommend optional courses to each student based on similar students. Similar students are identified using Manhattan, Euclidean, and cosine distances. These distances are modified in the thesis to consider only the courses that both students actually enrolled in when comparing two students. Modified distance metrics are compared to original ones in the evaluation. The effectiveness of the system with different hyperparameters is evaluated using various metrics, with the cosine distance with 20 neighbors chosen the best combination.

Keywords course recommendation, distance metric, recommender system, k-nearest neighbors, recommender system evaluation, FIT CTU courses

Cíl bakalářské práce

Hlavním cílem práce je prozkoumat možnosti a implementovat doporučovací systém pro volbu předmětů studentů Fakulty informačních technologií.

Cíle teoretické části práce

Cílem je analýza řešení problematiky doporučování. Dále si zadefinujeme základní pojmy, které se v oblasti strojového učení využívají. Pomocí těchto pojmů pak vysvětlíme algoritmus metody nejbližších sousedů. V poslední části teoretické části zanalyzujeme metriky vyhodnocování doporučovacích systémů.

Cíle praktické části práce

Analýza a vhodné předzpracování dostupných dat. Z těchto dat následně vybereme příznaky, které využije doporučovací systém pro určení relevantních předmětů každému studentovi. Tento systém implementujeme pomocí metody nejbližších sousedů. Pro určení nejbližších sousedů využijeme jak již existující metriky, tak si zadefinujeme několik vlastních metrik vzdálenosti studentů. Následuje hledání nejlepších hyperparametrů metody nejbližších sousedů, které najdeme měřením evaluačních metrik. Systém poté pomocí nejlepších hyperparametrů implementujeme.

Body literární rešerše

- Doporučovací systémy
- Algoritmy strojového učení
  - Vymezení pojmů
  - Definice vzdálenost s příklady
  - Metoda nejbližších sousedů
- Metriky pro vyhodnocování doporučovacích systémů

Body praktické části

- Seznámení s technologiemi
- Průzkum dat
- Předzpracování dat
- Implementace doporučovacího systému
- Měření metrik doporučovacích systémů
- Vyhodnocení jednotlivých přístupů
- Implementace systému s nejlepšími hyperparametry

Přínos

Hlavním přínosem bakalářské práce je možnost využít implementovaný systém jako rádce pro studenty, kteří neví, jaký předmět si mají zapsat do dalšího semestru.

Seznam zkratek

CSV	Comma-separated values
ČVUT	České vysoké učení technické v Praze
ECTS	The European Credit Transfer and Accumulation System
FIT	Fakulta informačních technologií
GUI	Graphical user interface
kNN	k-nearest neighbors
NaN	Not a Number
NumPy	Numeric python
SciPy	Scientific python

Úvod

Na evropských vysokých školách mají studenti pro úspěšné absolvování studia povinnost získat určité množství ECTS kreditů. V České republice odpovídá 1 kredit 30 hodinám času, který by student u předmětu měl strávit. Každý předmět je ohodnocen kredity podle jeho časové náročnosti.

Na FIT ČVUT v Praze jsou tyto předměty, za které má student možnost získat kredity, rozděleny do tří kategorií podle povinnosti:

- PP – povinné předměty programu, které musí splnit každý student.
- PO – povinné předměty oboru, které musí splnit každý student, který chce absolvovat studium nějaké specializace.
- V – volitelné předměty, které student nemusí splnit pro úspěšné absolvování vysoké školy.

V požadovaném množství kreditů pro úspěšné zakončení mají zastoupení všechny tyto skupiny předmětů.

Když si student zapisuje volitelné předměty do dalšího semestru, má obrovskou nabídku předmětů, ze které není snadné si vybírat. Ve školním systému KOS má každý předmět vlastní popis, sylabus a další informace. V dalším systému Anketa ČVUT má zase každý předmět slovní a bodové hodnocení od jiných studentů. Všechny tyto informace by měly studentovi pomoci s výběrem správných volitelných předmětů.

Bohužel se setkáváme se dvěma základními nedostatky. Čtení popisů předmětů v Bílé knize a hodnocení v Anketě ČVUT sebere mnoho času a i přes velmi důkladnou analýzu se stává, že předmět nesplní studentova očekávání. Cílem této práce je navrhnout a implementovat systém, který studentovi na základě toho, jaké předměty už odstudoval, doporučí, co by mohl studovat dál, aby tak nemusel procházet všechny předměty v Bílé knize, ale vybral si pouze ty, které pro něj jsou relevantní.

V práci se nejdříve zaměříme na doporučovací systémy, jaké jsou možnosti jejich implementace a využití. Dále si zadefinujeme několik důležitých pojmů potřebných pro popis metody nejbližších sousedů a nakonec se v teoretické části seznámíme s různými metrikami vyhodnocování doporučovacích systémů a algoritmů strojového učení.

V praktické části nejdříve vyjmenujeme technologie, které následně využijeme při analýze a předzpracovávání dat. Poté vytvoříme doporučovací systém, pro který budeme hledat nejvhodnější nastavení hodnot hyperparametrů. Nakonec implementujeme doporučovací systém pomocí zjištěných nejlepších hodnot hyperparametrů.

V doporučovacím systému se zaměříme na metodu nebližších sousedů, pro kterou implementujeme několik vlastních metrik vzdálenosti dvou studentů. Tento přístup k doporučování předmětů poté srovnáme nejdříve napříč různými nastaveními, s obvykle používanými metrikami pro doporučování, a nakonec i s dosavadním nejlepším řešením problému.

V několika pracech předchozích studentů se můžeme setkat s různými řešeními. Jednou z nich je diplomová práce Ondřeje Nového, která implementuje doporučovací systém pro předměty na FIT ČVUT pomocí metody asociačních pravidel a poté testuje metody deep learningu v kontextu doporučování – tyto dva přístupy se snaží srovnat a posléze nasadit. [1]

Druhou prací, na kterou budeme navazovat, je doporučovací systém založený na metodě nejbližších sousedů, který je implementován nad doménou předmětů na středních školách v Nizozemsku. [2]

Poslední práce, která stojí za zmínění, je práce Anny Kapitánové. Ta se musela ve své práci vypořádat s daty ze školního systému KOS ČVUT. [3]

Kapitola 1
Doporučovací systémy

V aplikacích získáváme od uživatelů mnoho informací o tom, co dělají, kam se dívají, jak hodnotí různé položky atd. V první kapitole se seznámíme s obecnou teorií doporučovacích systémů, zjistíme, jak fungují a jaké jsou jejich silné a slabé stránky.

1.1 Co je doporučovací systém

V rámci některých aplikací, se kterými se každý setkává, se může nacházet modul, který je na této aplikaci závislý, bere z ní data o uživatelích a položkách, se kterými uživatel interaguje. Na základě těchto dat modul vyhodnocuje pomocí různých metod, o které položky by mohl mít daný uživatel největší zájem. Tento modul nazveme *doporučovací systém*. [4]

Na vstupu doporučovací systém očekává

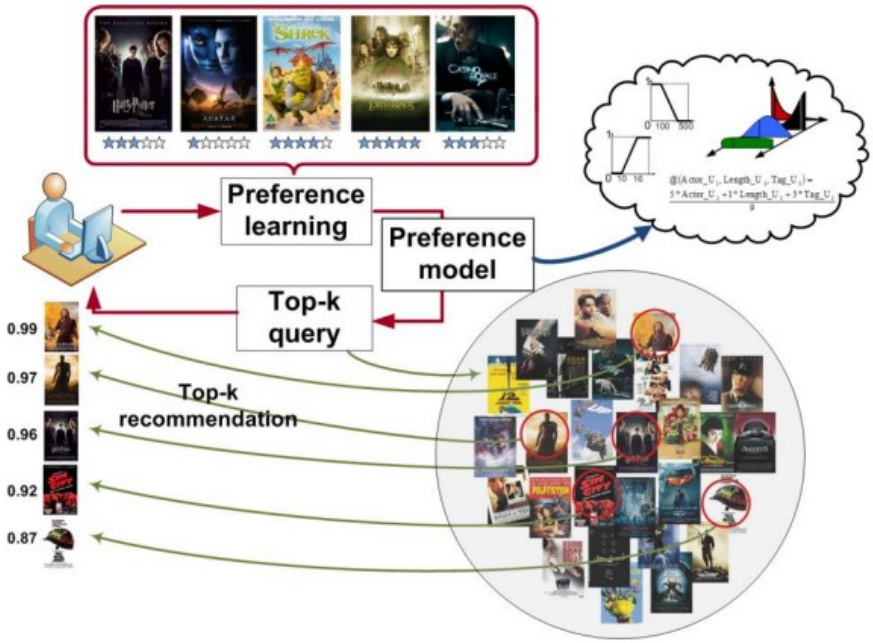
- údaje o uživateli,
- historii uživatele,
- položky, které má doporučovat,
- atributy položek, podle kterých máme doporučovat ty nejvhodnější,
- kontext (místo, roční období, aktuálně zobrazenou položku atd.).

Na výstup nám dává

- seřazený seznam nejvíce relevantních položek pro uživatele,
- hodnocení relevance těchto položek pro uživatele.

1.2 Jak systém pracuje

Na obrázku 1.1 je vidět cesta dat od uživatele do doporučovacího systému a zase zpátky k uživateli. *Preference learning* odpovídá získávání dat a *preference model* a *top-k query* zastupuje různé způsoby jejich zpracování. Jednotlivé kroky si postupně vysvětlíme.



Obrázek 1.1 Nákres workflow doporučovacího systému [4]

1.2.1 Zisk dat

Data se dají získat několika způsoby, které dělíme na *explicitní* a *implicitní*. Způsob získávání dat musíme zvolit v závislosti na tom, jakým způsobem budeme následně data zpracovávat.

Explicitní sběr dat znamená, že uživatel musí vyvinout nějakou aktivitu navíc pro to, abychom od něj data dostali. Jde například o psaní recenzí, hodnocení hvězdičkami apod. Explicitně sebraná data mají hlavní výhodu v tom, že jsou snadno interpretovatelná. Problémem však je, že jsou subjektivní, nedostáváme je od všech uživatelů a mohou být snadno podvrhnuta.

Implicitní data jsou naopak data taková, která sleduje naše aplikace, aniž by si toho uživatel všiml. Jedná se zejména o chování v aplikaci, kam uživatel kliká, kolikrát obrazovku navštívil... Tato data sice můžeme snadno získat, ale často jsou těžko interpretovatelná nebo obsahují šum. Šum jsou například data získána tím, že uživatelé omylem kliknou, kam nechtějí, prohlíží si položky jen pro zajímavost bez cíle s nimi jakkoliv interagovat, sdílí stejný profil v aplikaci ve více lidech a mnoho dalšího.

Je velmi těžké určit, která data implicitního typu bychom měli sledovat, aby systém dostal relevantní informace, podle kterých se může správně rozhodovat při doporučování.

Další nevýhodou těchto dat je, že obsahují většinou pouze pozitivní interakce s obsahem (např. klikání, nákup,...) narozdíl od explicitního způsobu, kde můžeme určit hodnotu reakce (1 hvězdička = špatné hodnocení, 5 hvězdiček = dobré hodnocení). [4]

1.2.2 Způsob zpracování dat

Opět máme několik způsobů, které si postupně vysvětlíme v následující části. Jejich rychlý přehled je vidět v tabulce 1.1.

Tabulka 1.1 Způsoby zpracování dat

Způsob	Struktura dat	Zdroj dat	Metoda zpracování
Kolaborativní filtrování	matice hodnocení položek uživateli	uživatelské hodnocení položek	identifikace podobných uživatelů
Na základě obsahu	vlastnosti položek	uživatelské hodnocení položek	identifikace podobných položek
Na základě znalostí	vlastnosti položek	potřeby uživatele	spojení vlastností položek a potřeb uživatele
Demograficky	demografická data	demografická data	filtry demografických atributů

1.2.2.1 Kolaborativní filtrování

Data jsou shromážděna v matici, která má v řádcích všechny uživatele a ve sloupcích všechny položky, které můžeme doporučit. Matice může být buď binární v případě, že máme k dispozici pouze data typu *účastnil se/neúčastnil se* nebo *líbilo se/nelíbilo se*, a nebo může obsahovat reálná čísla například pro vztah *uživatel dal položce x hvězdiček*.

Na základě těchto dat je aktivnímu uživateli[1] vypočtena podobnost s ostatními uživateli. Následně ze všech uživatelů vybereme ty nejpodobnější aktivnímu uživateli. Vytvoříme množinu položek, které aktivní uživatel ještě nehodnotil (nemá o nich záznam v matici) a zároveň je skupina jemu podobných uživatelů už ohodnotila. Z této množiny položek aktivnímu uživateli doporučíme tu s nejlepším a nejrelevantnějším hodnocením. [4]

Kolaborativní filtrování na základě uživatele Pro každou položku spočítáme skóre $p_{a,j}$, které říká, jestli aktivnímu uživateli a doporučíme položku j pomocí vztahu:

$$p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i).$$

- $\overline{v}_a$ (resp. $\overline{v}_i$) – průměrné hodnocení od uživatele a (resp. i), které je definováno vztahem:[2]

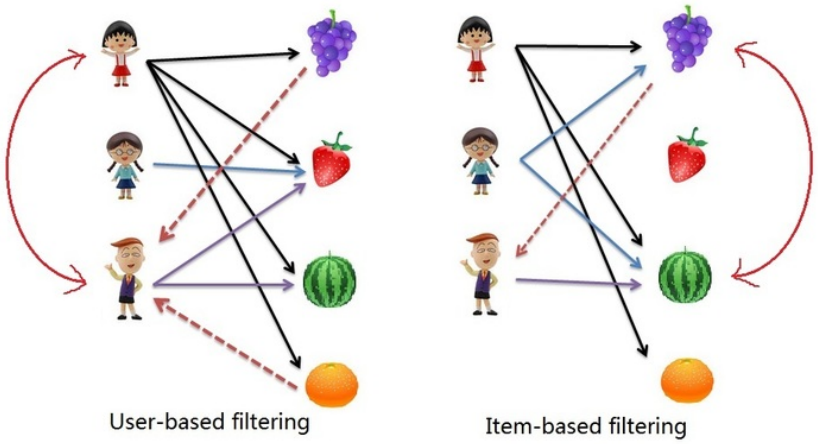
$$\overline{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j},$$

- κ – hodnota, která slouží k normalizaci sumy, jestliže nechceme normalizovat, pak $\kappa = 1$,
- n – počet uživatelů, které chceme brát v potaz při výpočtu $p_{a,j}$
- $w(a,i)$ – koeficient podobnosti uživatele i vůči aktivnímu uživateli a,
- $(v_{i,j} - \overline{v}_i)$ – rozdíl hodnocení *j-té* položky *i-tým* uživatelem a průměrného hodnocení od uživatele i.

Poté tato skóre seřadíme a vybereme z nich k nejlepších položek, které uživateli a doporučíme.

Kolaborativní filtrování na základě položky Měříme podobnost položek a podle ní doporučujeme uživateli další položky. Aktivní uživatel a ohodnotil položku x. Položka y má podobné hodnocení různými uživateli jako položka x a zároveň položka y nebyla hodnocena aktivním uživatelem a. Pokud je podle ostatních uživatelů položka y podobná položce x, doporučíme tuto položku y uživateli a.

[1] Uživatel, kterému právě teď chceme něco doporučit.
[2] Součet hodnocení všech položek, které uživatel i hodnotil, vydělený počtem položek, které uživatel i hodnotil.



Obrázek 1.2 Rozdíl mezi kolaborativním doporučováním na základě uživatele a na základě položky [5]

Příklad rozdílů mezi přístupy ke kolaborativnímu filtrování Na obrázku 1.2 jsme při doporučování na základě uživatele vyhodnotili, že chlapec vespod je podobný dívce nahoře a nabídli mu ovoce, které ještě neměl, ale dívka už ano.

U doporučování na základě položky jsme chlapci doporučili víno, protože všichni, co měli meloun – stejně jako chlapec dole – si dali i víno.

Problémy kolaborativního filtrování

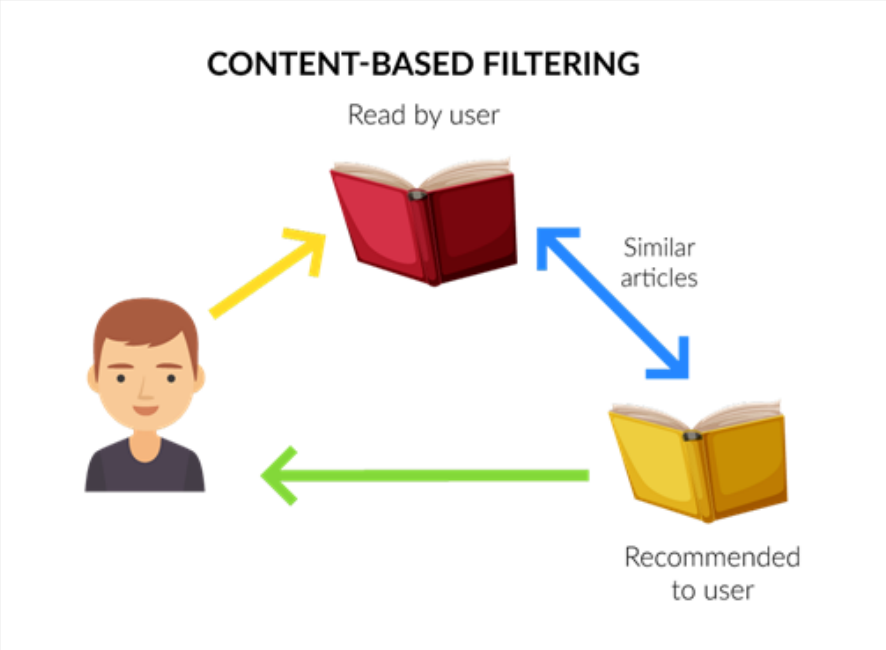
- Často máme velmi řídkou matici, ve které uživatelé interagují s položkami.
- Doporučování probíhá až po získání nějakých dat z aplikace. Při první zkušenosti uživatele se systémem nejsme schopni pomocí kolaborativního filtrování cokoliv doporučit správně, to platí i pokud uživatel ještě neprovedl dostatek hodnocení položek. Zároveň je kladen požadavek na to, aby se systém rychle učil, když mu přijde nové hodnocení.
- Těžko se interpretuje, na základě čeho systém rozhodl položku doporučit.
- Ve velkých systémech obvykle bývá problém s výpočetní náročností určování podobnosti uživatelů nebo položek. Dá se řešit pomocí shlukování. [4]

Řešení některých problémů Některé položky, které do aplikace zadáváme, si jsou velmi podobné a dokážeme to určit ještě před vstupem uživatelů. Tyto položky můžeme sloučit do kategorií a místo samotných položek se v matici zabývat těmito shluky.

1.2.2.2 Filtrování na základě obsahu

Kromě sledování hodnocení položek uživateli si všímáme i vztahů a souvislostí mezi jednotlivými položkami nebo uživateli. Vstupem jsou atributy každé položky, na základě kterých klasifikujeme, jestli položku doporučíme nebo ne (případně určíme nějaké její skóre pro doporučování). [4]

Metoda nejbližších sousedů Pokud uživatele zaujme některý předmět, určíme pomocí vybraných metrik vzdálenosti (*viz. část 2.2*) k nejbližších položek, které uživateli doporučíme. Více o metodě k-nejbližších sousedů nás čeká v části 2.3 kapitoly o strojovém učení.



Obrázek 1.3 Nákres filtrování na základě obsahu [6]

Nacházíme tak položky, které mají něco společného s těmi, které už uživatel viděl. Mezi nimi jsme schopni vybrat takové, které ještě neviděl a ty mu ukázat. Spoléháme se na to, že spolu položky souvisí a tím pádem by pro uživatele mohly být relevantní.

Výhody a nevýhody doporučování na základě obsahu

- Nezávislost na uživatelském hodnocení položek.
- Snadná interpretovatelnost, proč systém rozhodl, jak rozhodl.
- Položky se dají do systému přidávat a nemusí se čekat, až je někdo ohodnotí.
- Nedá se používat univerzálně, systémy jsou závislé na zkoumané doméně a nedají se snadno přenášet.
- *Overspecialization* [3] – Pokud máme v systému mnoho sobě velmi blízkých položek, nebudeme schopni doporučovat nic jiného kromě těchto blízkých položek. Ukázka s popisem obrázku 1.3:
  - Pokud bychom měli mezi položkami více červených knih, které považujeme nejpodobnější červené knize, kterou si přečetl uživatel, protože jsou červené. Nejsme schopni doporučit knihu žlutou i přes to, že by se věnovala stejnému tématu, a tím pádem by měla být původní červené knize podobnější.
- Když přijde nový uživatel, musíme nejdříve zjistit, co ho zajímá, než mu začneme něco doporučovat.

[3] Dalo by se volně říci, že jde o problém uváznutí v lokáním extrému.

1.2.2.3 Filtrování na základě znalostní báze

V některých doménách nejsme schopni nasbírat dostatek zpětné vazby pro využití kolaborativního filtrování a zároveň spolu položky málo souvisejí, abychom mohli nasadit doporučování na základě obsahu. [4]

Zpravidla jde o doporučování položek, které uživatel nenakupuje často nebo kde je pro správné doporučení[4] potřeba přímá pomoc od uživatele.

Znalostní báze, na základě které doporučujeme, může být vytvořena za pomoci splňování zadefinovaných požadavků (*constraint-based*). Pokud nejsme schopni určit požadavky, můžeme se pokusit doporučit položky ze známých případů správného doporučení, které jsme již v minulosti provedli (*case-based*). Takovou bázi můžeme s postupem času rozšiřovat.

Znalostní báze je definována základními komponentami:

- vlastnostmi uživatelů,
- vlastnostmi položek,
- požadavky na vlastnosti uživatelů,
- filtrovacími podmínkami mezi uživateli a položkami,
- možnou skladbou doporučených položek.

Hlavní výhodou tohoto přístupu je velmi snadná interpretace, jak systém položky doporučuje.

Při tvorbě báze znalostí se ale potýkáme s mnohými problémy. Zkonstruovat bázi znalostí je složité a časově náročné. Je třeba používat specifikované nástroje. Zpočátku navržená omezení nemusí být později platná a hledat je v bázi znalostí, abychom je mohli smazat, je obtížné. Některá omezení mohou být nesplnitelná a vracet prázdnou množinu. Často musíme s uživatelem provádět nějaký dialog. Podle tohoto dialogu se pak musíme rozhodovat, jakým způsobem se budeme ptát dál, aby nám splňování podmínek upřesňovalo informaci, ke které se systém snaží dobrat.

1.2.2.4 Demografické filtrování

Jde pouze o filtry, které ořezávají množinu možných doporučení. Například v Africe, kde bývá zpravidla teplé počasí, nebudeme doporučovat nákup zimních bund, nebo v případě sociálních sítí během období předvolebních kampaní v určité zemi budeme doporučovat pouze politický obsah, který je pro tuto zemi relevantní. [4]

1.3 Využití doporučovacích systémů

V dnešní době je kolem nás spoustu informací, ve kterých se kvůli jejich obroskému množství často není možné vyznat. Proto využíváme doporučovací systémy, abychom odfiltrovali informace, které nás nezajímají nebo pro nás nejsou relevantní. Pomocí doporučovacích systémů vytváříme tzv. *personalisovaný obsah*.

S doporučovacími algoritmy se setkáváme například při nakupování na internetu, sledování obsahu na sociálních sítích nebo ve streamovacích službách. Chování doporučovacích algoritmů vychází z informací, které získává naší aktivitou v aplikacích.

Doporučovat se dá cokoliv. V této práci se dále budeme věnovat doporučování předmětů, které by si mohl chtít zapsat student na Fakultě informačních technologií ČVUT.

Doporučovací systém nám může dobře posloužit, ale nesmíme se jím nechat ovládat [5].

[4] Například vyplnění nějakých filtrů na e-shopech.

[5] Pokud systém studentovi doporučí zapsat si nějaký předmět, student by si měl nejdříve ověřit, co mu bylo doporučeno, a až poté zvážit zápis předmětu.

Kapitola 2
Strojové učení

V následující kapitole se dostaneme k teorii algoritmu, který následně budeme využívat při modelování doporučovacího systému. Nejdříve budou zmíněny potřebné pojmy z oblasti strojového učení, poté samotný algoritmus a nakonec jeho detaily.

2.1 Vymezení pojmů

Než se pustíme do vlastního vysvětlení metody, musíme se napřed vybavit pojmy, které jsou pro její pochopení důležité.

Strojové učení se dělí na dva základní proudy – učení bez učitele (*angl. unsupervised learning*) a učení s učitelem (*angl. supervised learning*).

2.1.1 Učení bez učitele

Učení bez učitele má za úkol nám dát představu o struktuře dat a vzájemných vazbách mezi nimi. Narozdíl od učení s učitelem nemáme žádnou vysvětlovanou proměnnou, jejíž hodnotu bychom měli předpovídat. Jeho nejběžnějším využitím je shlukování, díky kterému v datech identifikujeme body, které si jsou podobné, a diskutujeme, které příznaky nejvíce přispívají k tomu, aby se data na tyto shluky dala rozdělit. [7]

2.1.2 Učení s učitelem

Učení s učitelem využíváme při snaze předpovídat hodnotu vysvětlované proměnné na základě hodnot uložených v jednotlivých příznacích, které data na vstupu mají. Nemusí vždy jít o data uložená v tabulkách, ale můžeme tímto způsobem zpracovávat i obrázky, videa nebo časové řady.

Mějme data, která mají p příznaků $X_0, X_1, \ldots, X_{p-1}$ a vysvětlovanou proměnnou Y, kterou dokážeme pomocí funkce odhadnout na základě hodnot příznaků $X_0, X_1, \ldots, X_{p-1}$.

$$Y \approx f(X_0, X_1, \ldots, X_{p-1})$$

Funkce f zastupuje model, kterým predikujeme hodnotu Y. Takovým modelem mohou být rozhodovací stromy, neuronové sítě nebo takovou funkci může zastupovat metoda k-nejbližších sousedů a mnoho dalších.

Vytváření modelu probíhá tak, že data rozdělíme na trénovací a testovací množinu. Trénovací množina zpravidla obsahuje více dat než testovací. Obvykle 70–85 % dat náleží trénovací

množině a 15–30 % dat potom množině testovací. Z testovací množiny odebereme vysvětlovanou proměnnou, kterou si uložíme pro následné určení přesnosti modelu.

V rámci trénovací množiny má model za úkol na základě dat predikovat hodnoty správné hodnoty. Model se tak v datech snaží identifikovat vzorce, podle kterých je možné správně předpovídat hodnotu vysvětlované proměnné, zatímco její výslednou hodnotu zná a může podle ní adaptovat své vnitřní parametry, podle kterých se rozhoduje.

Po trénování model spustíme na testovací množině a porovnáváme hodnoty vysvětlované proměnné z původních dat s hodnotami, které model určil. Následně podle porovnání těchto dvou množin hodnot vyhodnotíme hodnotu metrik vyhodnocení modelu, kterými určujeme, o jak dobrý model se jedná. Více o vyhodnocování modelu si ukážeme v kapitole 3.

Učení s učitelem můžeme rozdělit podle problému, který pomocí něj řešíme, na *klasifikaci* a *regresi*.

Problém klasifikace řešíme v momentě, kdy predikovaná hodnota může nabývat pouze několika diskrétních hodnot[1]. Dále pokud rozhodujeme pouze o tom, zdali něco platí nebo ne, mluvíme o binární klasifikaci.

Problém regrese nazýváme úlohu, kdy jde o hledání řešení spojitého problému. K vyhodnocování modelu se využívají jiné metriky, protože ve spojitém prostoru řešení téměř nikdy nenarazíme na přesnou shodu predikce s výsledkem.

2.1.3 Hyperparametry algoritmu

Každý algoritmus má nějaké hyperparametry. Jedná se o hodnoty, které jsou algoritmu dopředu známé, které musí zadat uživatel, pokud algoritmus chce používat. Uživatel zpravidla před použitím správné hodnoty hyperparametrů nezná, protože každá úloha vyžaduje jiné nastavení.

Procesu hledání správných hodnot říkáme *ladění hyperparametrů* a často probíhá tak, že uživatel trénuje model vždy od začátku s různým nastavením hyperparametrů a posléze zkoumá úspěšnost modelu na testovacích datech. Na základě tohoto průzkumu vybere nejlepší hodnoty a s těmito hyperparametry provede trénování ještě jednou. Výsledkem tohoto procesu je finální funkční model strojového učení.

Při ladění parametrů se často setkáváme s pojmem *bias-variance tradeoff*, který se snažíme vyvážit tak, abychom měli co nejlepší model.

2.1.4 Bias-variance tradeoff

Při trénování modelu hledáme hodnoty hyperparametrů tak, abychom předešli podučení a zároveň zamezili přeučení modelu.

Podučení (*angl. underfitting*) odpovídá nedostatečné adaptaci modelu na trénovací data. To znamená, že model se z dat nenaučil žádné vzorce, na základě kterých by se měl rozhodovat, a tím pádem je příliš obecný. Model tak dosahuje vysoké nepřesnosti jak na trénovacích, tak na testovacích datech.

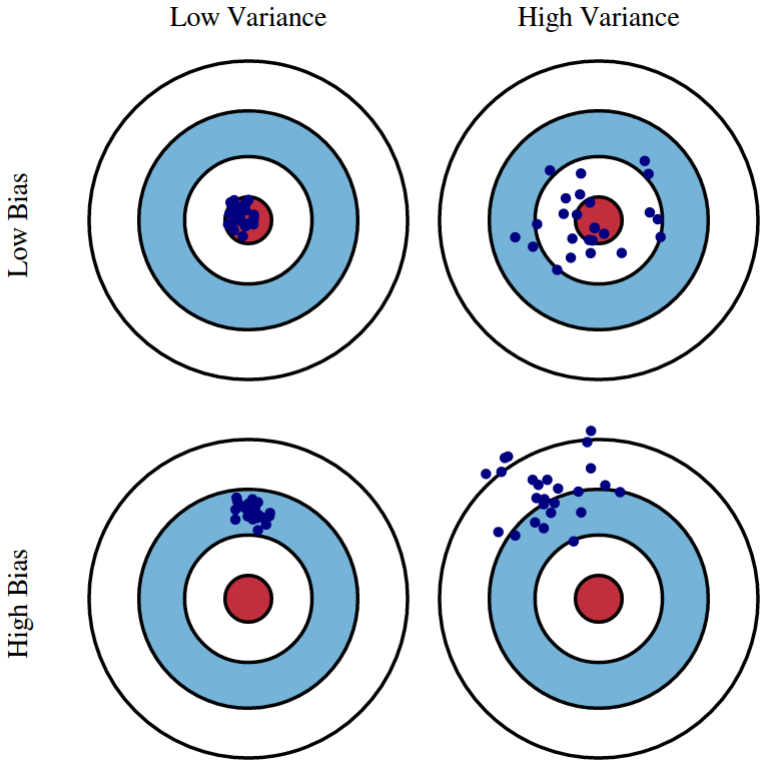
Přeučení (*angl. overfitting*) je naopak přílišná adaptace modelu na trénovací data. Ta nastává, když je model málo obecný a přesně dokáže predikovat hodnoty vysvětlované proměnné na trénovací množině dat (včetně šumu a náhodných chyb). To vede k vysoké přesnosti na trénovacích datech, ale k chybám na datech testovacích. [8]

[1] Můžeme určit jejich počet, i když to nemusí být málo.

V obou případech jde s největší pravděpodobností o nesprávné nastavení hodnot hyperparametrů před tréninkem. Řešení spočívá ve volbě jiných hodnot a opětovném trénování modelu.

Odchylka (*angl. bias*) nám v kontextu strojového učení udává rozdíl mezi naším modelem, předpovídanou hodnotou a správnou hodnotou, kterou měl model předpovědět. Modely, které mají velkou odchylku od skutečných dat, dosahují obrovské chybovosti jak na trénovacích, tak na testovacích datech. Pokud máme velkou odchylku od skutečných dat, znamená to, že model se málo přizpůsobil trénovacím datům během procesu učení. Model tak dosahuje velkých nepřesností jak na trénovací, tak na testovací množině dat. [9]

Rozptyl (*angl. variance*) nám říká, jestli je model schopný generalizovat data. Pokud je rozptyl velký, model je příliš specifický na data, na kterých se trénoval. Můžeme tak tvrdit, že se model nenaučil vzorce v datech, nýbrž data samotná. Model je přeučený, na testovací množině dosahuje obrovské chybovosti a nemůžeme ho dále použít pro predikce.

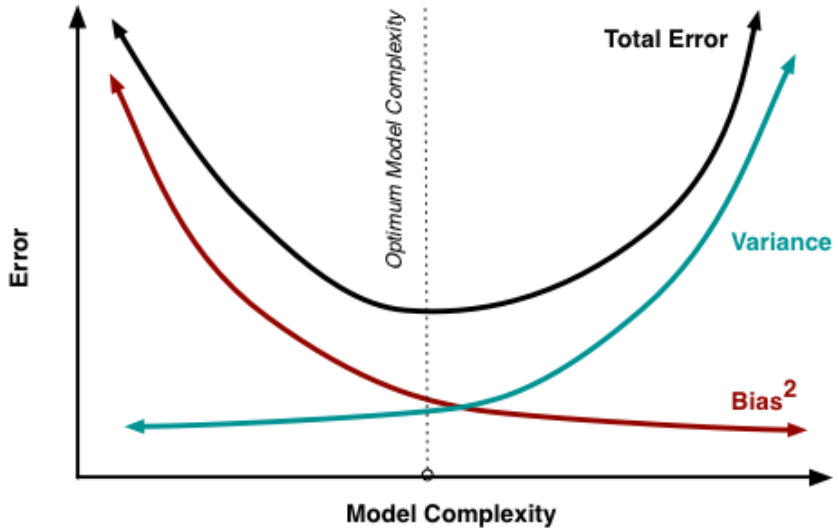


Obrázek 2.1 Přesnost predikce při nízké/vysoké odchylce/rozptylu [9]

Při učení modelu s časem roste rozptyl – model se snaží pokrýt data co nejvíce – a zároveň klesá odchylka – model se datům přibližuje. Na obrázku 2.1 si jako model, který trénujeme, můžeme představit červený kruh, kterým se snažíme predikovat data, která jsou představována modrými tečkami. Bias-variance tradeoff se dá shrnout formulí:

$$chyba\_na\_testovaci\_mnozine = odchylka^2 + rozptyl + irreducibilni\_chyba.$$

Formule[2] nám říká, že chyba se skládá z odchylky, rozptylu a irreducibilní chyby[3]. Naším cílem během trénování modelů je najít rovnováhu mezi odchylkou a rozptylem jako je vidět na obrázku 2.2 tak, aby model nebyl podučený a zároveň přeučený. Irreduciblilní chyba je zapříčiněna daty, které nejsme modelem vysvětlit (data si mohou například protiřečit). Zároveň není naším cílem takovou chybu odstraňovat, protože model, který ji odstraňuje je náchylný k přeučení.



Obrázek 2.2 Graf ukazující závislost chyby modelu na jeho složitosti [9]

2.1.5 Normalizace dat

Proč se něčím takovým máme zabývat vysvětlíme na příkladu a posléze ukážeme, jak k normalizaci přistupovat. [11]

▶ **Příklad 2.1.** Nacházíme se na doméně prodeje domů. Mějme spojité numerické příznaky, které určují:

- velikost zahrady (v m^2),
- velikost kuchyně (v m^2),
- velikost televize danou úhlopříčkou (v palcích),
- počet místností.

Kdybychom všechny rozměry převedli na m^2, měli bychom je sice ve stejných jednotkách, ovšem špatně naškálované. K tomu všemu do tohoto systému nejsme schopni převést příznak počtu místností.

Problém z příkladu nazýváme *nedosažitelná souměřitelnost příznaků* a jeho správné řešení vyžaduje rozsáhlejší analýzu zkoumané problematiky. Zároveň se ale můžeme pokusit data *normalizovat* a vyzkoušet, jestli tudy cesta nevede.

[2] Její odvození je dostupné z [10].
[3] Irreducibilní chybou je šum nebo náhodné chyby v datech.

Normalizace dat znamená, že každý příznak naškálujeme do intervalu $[0, 1]$. Tato operace se týká pouze numerických příznaků a v žádném případě to není univerzální způsob, jak tento problém řešit.

Škálování provedeme tak, že pro každý příznak nalezneme jeho minimální (min_x) a maximální (max_x) hodnotu, a poté každou hodnotu x_i tohoto příznaku nahradíme pomocí vzorce

$$x_i \leftarrow \frac{x_i - min_x}{max_x - min_x}.$$

2.2 Vzdálenost a její příklady

Dále si zavedeme pojem, který je u metody k-nejbližších sousedů zásadní. Musíme být každému datovému bodu určit, které z ostatních datových bodů pro mu jsou více nebo méně vzdáleny. S tím nám pomůže *vzdálenost*, která nám říká, jak daleko se od sebe dva datové body nacházejí. [11]

▶ **Definice 2.2.** *Vzdálenost (nebo také metrika vzdálenosti) na množině $\mathcal{X}$ je funkce $d : \mathcal{X} \times \mathcal{X} \to [0, +\infty)$ taková, že pro každé $x, y, z \in \mathcal{X}$ platí*

i) pozitivní definitnost: $d(x, y) \geqq 0$ a $d(x, y) = 0$ právě tehdy, když $x = y$,

ii) symetrie: $d(x, y) = d(y, x)$,

iii) trojúhelníková nerovnost: $d(x, y) \leqq d(x, z) + d(z, y)$.

V další části si zadefinujeme metriky vzdálenosti, které je možné v implementaci metody k-nejbližších sousedů v knihovně *scikit-learn* využít. Mezi ně patří manhattanská vzdálenost, euklidovská vzdálenost a cosinová vzdálenost. [12]

Metriky vzdálenosti ale mohou být navrženy přímo na doménu, které se mají věnovat, aby dosáhly lepších výsledků. V knihovně je například ještě implementována haversinská vzdálenost, která se využívá při výpočtech vzdálenosti, kde chceme brát v potaz zeměpisnou šířku a délku. [13]

V našem případě si v kapitole o návrhu systému 7.2.2 doménově specifickou metriku také vyzkoušíme navrhnout, implementovat a porovnat s běžně využívanými metrikami.

2.2.1 Minkovského norma

Jedná se o obecný předpis výpočtu vzdálenosti dvou datových bodů. [11]

▶ **Definice 2.3** (Minkowského norma)**.** *Mějme dva datové body $\boldsymbol{x} = (x_1, \ldots, x_n)$ a $\boldsymbol{y} = (y_1, \ldots, y_n)$. Oba mají n příznaků. Potom vzdálenost těchto dvou bodů můžeme spočítat pomocí vzorce:*

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}.$$

Parametrem p určujeme charakteristiku výpočtu vzdálenosti.

2.2.1.1 Speciální případy Minkowského normy

V rámci knihovny *scikit-learn* jsou implementovány dva speciální případy Minkowského normy.

Manhattanská vzdálenost (*jinak také city block*) odpovídá hodnotě parametru $p = 1$. Zároveň jde o absolutní rozdíl vzdálenosti v kartézském systému souřadnic. Konkrétně tedy:

$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i|.$$

Euklidova vzdálenost odpovídá hodnotě parametru $p = 2$. Tato metrika se dá připodobnit ke vzdušné vzdálenosti na mapě. Vzorec pro její výpočet po dosazení do normy je následující:

$$d_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}.$$

Obrázek 2.3 Grafické srovnání výpočtu manhattanské a Euklidovy vzdálenosti ve 2D prostoru [14]

2.2.2 Cosinová vzdálenost

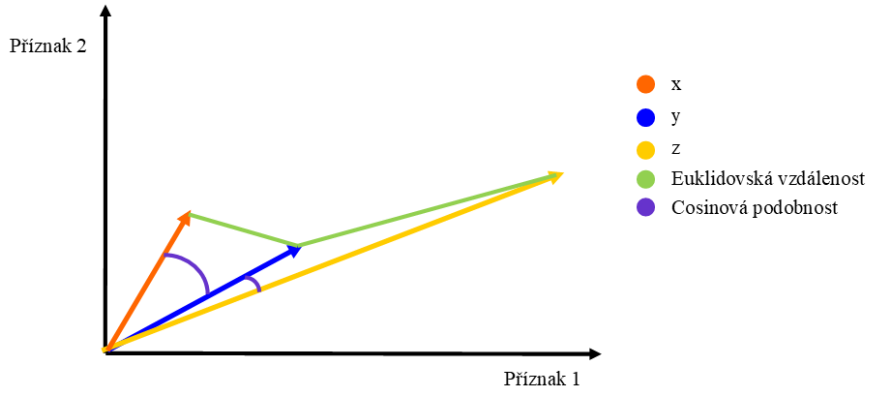
Cosinová vzdálenost nám dává úplně jiný pohled na měření vzdálenosti než metriky podle Minkowského normy. Snaží se totiž místo vzdálenosti modelovat vzdálenost dvou bodů pouze v kontextu úhlu, který svírají dva vektory. [15]

Představme si data jako vektory v prostoru. Cosinová vzdálenost nám potom měří vzdálenost dvou datových bodů na základě velikosti úhlu, který spolu dva vektory reprezentující tyto datové body svírají. Vzorec pro výpočet tedy vypadá následovně:

$$d_{cos}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \cdot ||\vec{y}||} = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sum_{i=1}^{n} y_i^2}$$

Na obrázku 2.4 se snažíme spočítat metriku pro tři datové body. Je vidět, že v případě volby Euklidovy vzdálenosti si jsou blíže body x a y, zatímco pokud bychom volili cosinovou vzdálenost, jsou si bližší body y a z.

Obvykle se tato metrika využívá při doporučování dokumentů, kde dokument nejdříve rozdělíme na jednotlivá slova a provedeme jejich předzpracování a poté zkoumáme jejich vzájemnou polohu v mnohadimenzionálním prostoru.



Obrázek 2.4 Grafické srovnání výpočtu Euklidovy vzdálenosti a cosinové vzdálenosti ve 2D prostoru

2.2.3 Levenshteinova vzdálenost

Tato metrika pracuje nad řetězci. Můžeme ji využít při hledání podobných slov v příznacích. Nebere však v potaz jejich význam, nýbrž editační vzdálenost. [16]

▶ **Definice 2.4** (Levenshteinova vzdálenost)**.** *Uvažujme řetězce nad abecedou* Σ*. Nad každým řetězcem zadefinujeme operace vložení znaku, smazání znaku nebo záměnu jednoho znaku za jiný. Levenshteinova (editační) vzdálenost řetězců* $\boldsymbol{x} = x_1, \ldots, x_m$ *a* $\boldsymbol{y} = y_1, \ldots, y_n$ *poté označuje nejmenší počet operací potřebných k tomu, abychom z řetězce* $\boldsymbol{x}$ *vytvořili řetězec* $\boldsymbol{y}$*.*

Levenshteinova vzdálenost není v knihovně *scikit-learn* implementovaná, pokud bychom ji chtěli využít pro nějaký náš algoritmus, musíme si funkci pro její implementaci vytvořit sami a poté algoritmu, který má vzdálenost využít, vložit jako hyperparametr.

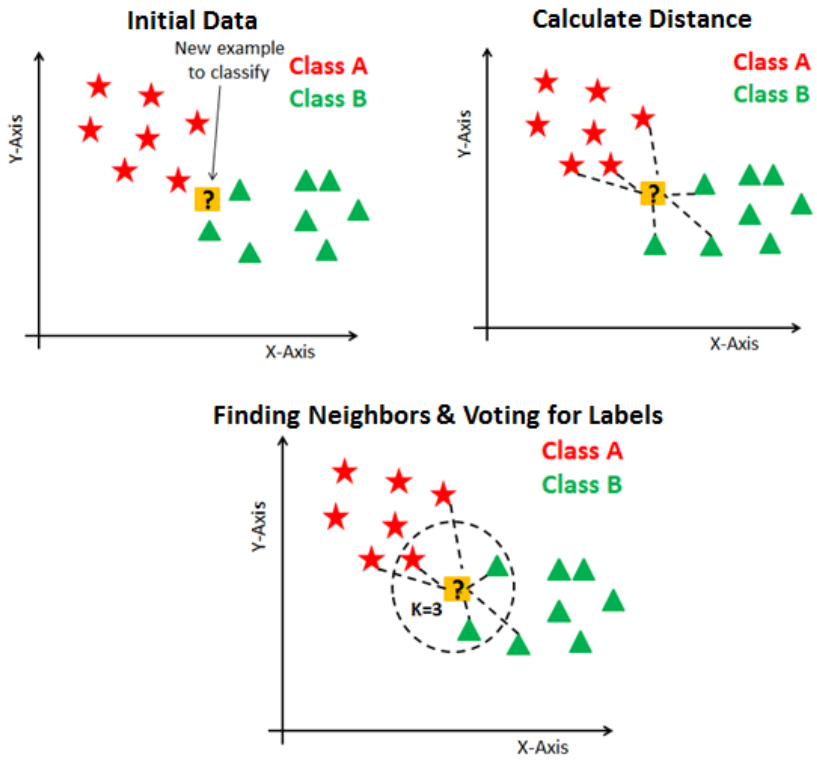
2.3 Metoda k-nejbližších sousedů (k-nearest neighbours)

Jedná se o jeden ze základních algoritmů strojového učení. Je velmi jednoduchý, a i přesto dokáže konkurovat složitějším modelům jako jsou například neuronové sítě. Je zároveň všestraný, protože se dá využít k řešení problémů supervizovaného i nesupervizovaného učení. Pro účely této práce bude stačit, když si vysvětlíme, jak se metoda pracuje v rámci učení s učitelem.

Základní myšlenka algoritmu spočívá v tom, že máme trénovací data $X \in \mathbb{R}^{N,p}$ se známými hodnotami vysvětlované proměnné $Y \in \mathbb{R}^N$ a dále datový bod $x \in \mathbb{R}^p$. Pro bod x nalezneme několik nejbližších datových bodů (*sousedů*) z trénovací množiny podle zadané metriky vzdálenosti. Podle hodnot vysvětlované proměnné těchto nejbližších bodů předpovídáme hodnotu vysvětlované proměnné datového bodu x.

Na obrázku 2.5 jde o binární klasifikaci[4] bodu x označeného otazníkem ve žlutém obdélníku. Spočítáme vzdálenost ke všem ostatním datovým bodům. Poté zvolíme hyperparametr K=3. Algoritmus podle tří nejbližších sousedů určí příslušnost bodu x. Protože ze tří nejbližších bodů dva náleží třídě B a pouze jeden třídě A, bodu x algoritmus přiřadí jako hodnotu vysvětlované proměnné třídu B.

[4] Vysvětlovanou proměnnou je zde příslušnost k třídě A nebo B.



Obrázek 2.5 Grafické znázornění metody k-nejbližších sousedů [17]

2.3.1 Vlastnosti algoritmu

Říkáme, že jde o neparametrický (*angl. non-parametric*) a líně se učící (*angl. lazy learning*) algoritmus. [18]

Neparametrický algoritmus znamená, že model nemá žádné parametry, to jsou hodnoty, které jsou naučeny z trénovacích dat a jsou specifické pro konkrétní model. Metoda k-nejbližších sousedů si během učení nevytváří žádnou vlastní znalost o struktuře a distribuci dat, neukládá si žádné parametry, na základě kterých by se rozhodoval. Jedná se tedy o neparametrický algoritmus.[5]

Lazy learning algoritmus znamená, že nepoužívá trénovací data pro naučení se vzorců, které trénovací data obsahují. Při vytváření predikcí tak algoritmus spočítá všechny vzájemné vzdálenosti všech bodů, vybere nejbližší a na základě nich rozhoduje o výsledku.

Absence trénovací části vytváření modelu je příčinou krátkého času, který by bylo třeba trénování věnovat. Výměnou za to ale predikce hodnot nějaký čas trvá kvůli výpočtu vzdáleností všech bodů.

Metoda k-nejbližších sousedů je tedy velmi závislá na vstupních datech. Model se díky tomu však dobře adaptuje na změny v datech. Této vlastnosti se dá dobře využít u doporučovacích systémů, kde se nám data neustále mění (resp. přibývají) a model je tak schopen snadno predikovat hodnoty na nových datech.

[5] Neplést si parametry a hyperparametry. Parametry se model učí sám, zatímco hyperparametry jsou zadané uživatelem.

Zároveň jsme ale u doporučování omezeni časovou složitostí, která roste s množstvím zpracovávaných dat. Použití této metody pak kvůli tomu není vhodné pro zpracování velkého množství dat.

Klasifikace a regrese pomocí knn se liší pouze ve výsledném výběru předpovídané hodnoty vysvětlované proměnné. Při klasifikaci mezi nejbližšími sousedy probíhá hlasování o příslušnosti bodu k nějaké třídě, zatímco při regresi pro předpověď výsledné hodnoty na základě nejbližších sousedů počítáme průměr. V obou případech lze brát v potaz váhy sousedů, kterými může být reprezentována například vzdálenost sousedů od datového bodu, pro který hodnotu vysvětlované proměnné určujeme.

2.3.2 Hyperparametry metody k-nejbližších sousedů

V rámci implementace metody k-nejbližších sousedů si před predikováním vysvětlované proměnné můžeme zvolit hodnoty několika hyperparametrů. [11]

Nejdůležitějšími z nich pro chování algoritmu jsou:

- *n_neighbors* – číslo k, které udává, podle kolika nejbližších sousedů budeme předpovídat hodnotu vysvětlované proměnné,
- *metric* – funkce, která má být využita pro výpočet vzdálenosti dvou bodů,
- *weights* – v případě, že chceme při výpočtu vysvětlované proměnné brát v potaz váhu souseda – nejčastěji za váhu volíme vzdálenost[6].

2.3.2.1 Počet sousedů

U metody k-nejbližších sousedů se jedná o stěžejní hyperparametr. Volíme číslo $1 \leqq k \leqq n$, kde n odpovídá celkovému počtu datových bodů.

Při volbě $k = 1$ mluvíme o metodě *1 Nearest Neighbor*, kdy najdeme vždy nejbližší datový bod a podle něj určíme hodnotu vysvětlované proměnné. Model je pak přeučený, protože nám vrací vedle správných výsledků i výsledky založené na šumu v datech nebo náhodných chybách, které se v datech mohou vyskytovat. [18]

Při volbě $k = n$ pak narážíme na problém, že každá hodnota, kterou se snažíme predikovat je rovna hodnotě, kterou nese třída, která je v datech zastoupena nejvíce.

Nižší hodnoty hyperparametru k vedou k velkému rozptylu (*angl. variance*), ale k nízké odchylce (*angl. bias*) modelu, vyšší hodnoty hyperparametru naopak mají malý rozptyl, ale vysokou odchylku. Proto je třeba iterativně nalézt nejlepší hodnotu hyperparametru, abychom vyhověli *bias-variance tradeoff*.

Nejlepší hodnotu hyperparametru k nalezneme tak, že postupně zkoušíme predikovat hodnoty vysvětlované proměnné v našich trénovacích datech a zjistíme, která hodnota k se k našim datům hodí nejvíce[7].

2.3.2.2 Metrika vzdálenosti

Pro správné fungování algoritmus k-nejbližších sousedů vyžaduje, aby mohl počítat metriku vzdálenosti. Tu můžeme vybírat z několika možností nebo si můžeme naimplementovat vlastní.

Volba metriky může vnést do algoritmu obrovskou míru složitosti, pokud bude její výpočet náročný. Její volba tak dokáže ze zdánlivě jednoduchého modelu udělat velmi složitý model, je proto třeba vybírat opatrně. [11]

[6] Čím menší hodnota vzdálenosti, tím vyšší váha při určování hodnoty souseda.

[7] Při problému binární klasifikace obvykle volíme liché číslo, aby existovala hlasovací většina.

2.3.2.3 Váhy

Na uvedeném příkladu ukážeme, jaký je rozdíl mezi tím, jestli váhy jako hyperparametr nastavíme nebo ne.

▶ **Příklad 2.5.** Řešíme problém regrese pomocí metody k-nejbližších sousedů. Pro datový bod $\boldsymbol{x}$ jsme nalezli k-nejbližších sousedů $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ s vyplněnými hodnotami vysvětlované proměnné $y_1, \ldots, y_k$.

Předpověď vysvětlované proměnné datového bodu $\boldsymbol{x}$ můžeme spočítat několika způsoby. Nejjednodušší z nich, který nebere v potaz žádné váhy, je *průměr*:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i.$$

Často ale chceme brát v potaz rozdíl mezi nejbližším a k-tým nejbližším sousedem. K tomu nám dobře poslouží *vážený průměr*:

$$\hat{y} = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i},$$

kde w_i jsou váhy jednotlivých bodů vůči bodu $\boldsymbol{x}$.

Za váhy $w_1, \ldots, w_k$ poté můžeme zvolit:

$$w_i = \frac{1}{d(\boldsymbol{x}, \boldsymbol{x}_i)}.$$

Tato volba odpovídá vahám na základě vypočtené vzdálenosti bodu $\boldsymbol{x}_i$ od bodu $\boldsymbol{x}$ a dává větší rozhodovací schopnost do rukou bližších sousedů před vzdálenějšími.[11]

Stejně jako u metriky vzdálenosti si také můžeme vytvořit vlastní způsob výpočtu váhy souseda při rozhodování, která ovlivňuje predikce vysvětlované proměnné. Váha také může být velmi komplexní výpočet, který zpomalí celý algoritmus, zároveň ale dokáže přinést lepší výsledky.

2.3.3 Výhody a nevýhody metody k-nejbližších sousedů

Zde na jednom místě shrneme hlavní výhody a nevýhody, kterými algoritmus k-nejbližších sousedů disponuje. [11] [17] [19]

Mezi hlavní výhody patří jednoduchost, krátký čas trénování, schopnost se adaptovat a nízký počet hyperparametrů, které musí uživatel nastavovat. Algoritmus však trpí tím, že ke své funkčnosti potřebuje mít přístup k datům, která musí být dobře předzpracovaná. Dále mu vadí práce s vyšším počtem příznaků a v neposlední řadě, když mu nastavíme špatné hyperparametry, můžeme dostávat zavádějící výsledky.

Jednoduchost Algoritmus je velmi jednoduchý jak na pochopení, tak na implementaci.

Krátká fáze trénování modelu Protože samotná data jsou v případě k-nejbližších sousedů natrénovaným modelem, vyžaduje se po nás ve fázi trénování pouze správné nastavení hyperparametrů. Trénovací fáze tak kvůli charakteru algoritmu vůbec neprobíhá.

Schopnost adaptace Model tvoří predikce z dat, a tak jej není třeba znovu trénovat, když se změní množství dat, které musí model zpracovávat.

Málo hyperparametrů Metoda k-nejbližších sousedů vyžaduje nastavit pouze počet sousedů, které má brát v potaz a metriku, kterou bude hodnotit jejich vzdálenost. Samozřejmě je možné nastavit ještě mnoho dalších hyperparametrů – například váhu nebo jestli má metoda počítat vždy vzdálenost ke všem datovým bodů – ale v porovnání se složitějšími algoritmy strojového učení se jedná o malý počet hyperparametrů, které jsou na vstupu vyžadovány.

Špatná škálovatelnost Protože je k-nejbližších sousedů lazy learning algoritmus, musí mít všechna data, podle kterých predikuje, uložená v paměti. Kvůli tomu jsou modely náročné na výpočetní zdroje, což je v případě zpracovávání velkého množství dat limituje.

Citlivost na data Algoritmus je velmi citlivý, co se týče:

- zašuměných dat – může tento šum dosadit za vysvětlovanou proměnnou,
- odlehlých hodnot – při vyhodnocení některé odlehlé hodnoty jako jednoho z nejbližších sousedů dochází ke špatné predikci vysvětlované proměnné,
- chybějících hodnot – na základě chybějících hodnot algoritmus není schopný nic předpovědět.

Potřeba normalizace dat Pro správný výpočet vzdálenosti datových bodů je potřeba mít data naškálovaná do rozmezí, které je pro všechny příznaky stejné.

Práce s nominálními příznaky Neexistuje obecný postup, jak v algoritmu naložit s daty, které přirozeně nemají žádné pořadí, protože to znemožňuje výpočet vzdálenosti. Je třeba vymyslet metriku, která se s těmito speciálními případy vypořádá, nebo v rámci předzpracování upravit data tak, abychom mohli využít nějakou ze standardních metrik[8].

Prokletí dimenzionality Pokud data, se kterými pracujeme, mají mnoho příznaků, model musí při výpočtu vzdálenosti dvou bodů zohlednit každý z nich. To vede na vysokou výpočetní náročnost.

V rámci předzpracování dat se tak snažíme pomocí různých metod vybírat pouze příznaky, které nesou nějakou informační hodnotu pro vysvětlovanou proměnnou.

Náchylnost k přeučení Model může být podučený nebo předučený při špatném výběru hyperparametru počtu sousedů. Tyto jevy zároveň mohou nastat při snaze o redukci počtu dimenzí při předzpracování dat nesprávným výběrem příznaků.

[8] Příklad pro lepší představu: Máme příznak, který určuje část Prahy (při zpracovávání prodeje domů v Praze). V takovém případě není vhodné měřit rozdíl čísel.

Kapitola 3

Vyhodnocení přesnosti doporučování

Systém považujeme za funkční, když vrací správné výsledky. Pokud implementujeme několik různých systémů, které se zabývají stejným problémem, musíme je umět srovnat, abychom z nich byli schopni vybrat ten nejlepší. K výběru nám slouží metriky vyhodnocování přesnosti nebo chybovosti při doporučování. V následující kapitole si několik metrik představíme. Dále také zmíníme, s jakými hodnotami budeme porovnávat systém vyvinutý v této práci se systémem, který byl vyvinut v rámci jiné práce pro řešení stejné problematiky.

3.1 Metrika měření systémů

Celá problematika ohledně správnosti doporučování a vyhledávání na webu a v multimediálních databázích se nazývá informační vyhledávání (*angl. information retrieval*). V rámci ní se snažíme vyvážit rychlost, přesnost a množství vyhledaných zdrojů informací při zadání dotazu[1]. Všechny tyto vlastnosti se dají měřit.

Co se týče přesnosti, nejdříve si ukážeme základní statistické míry měření v rámci strojového učení, navážeme metrikami, které se využívají v rámci information retrieval, představíme si i složitější míru, která bere v potaz nejen, zdali byla položka doporučena správně, ale zároveň počítá i s pořadím, ve kterém byla položka doporučovacím systémem vrácena.

Nakonec zadefinujeme veličinu ohledně množství prokrytých doporučitelných položek, kterou nazveme *coverage*.

3.1.1 Statistické měření chybovosti

Používáme při porovnávání odchylky předpokládaného doporučení od doporučení, které provedl systém. Hojně využívané jsou ve strojovém učení i mimo oblast doporučovacích systémů. [20]

Mean absolute error (MAE) označuje průměrný rozdíl mezi předpokládanou vrácenou hodnotou y a hodnotou vrácenou systémem $\hat{y}$, celý rozdíl je v absolutní hodnotě[2]. Tyto absolutní chyby následně sečteme a vydělíme počtem měření.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

[1] Dotazem se myslí například i doporučení jednomu uživateli.

[2] To abychom splnili požadavek definice na pozitivní definitnost.

Hlavní výhodou této metriky je jednoduchá interpretace. Při jejím použití se musíme zamyslet, jakým způsobem chceme naložit s odlehlými hodnotami, protože na ně metrika není citlivá.

Mean squared error (MSE) udává opět míru chyby, nyní je však rozdíl mezi správnou a predikovanou hodnotou umocněn. Díky tomu jsou více trestány predikce modelu, které jsou daleko od očekávané hodnoty. Jinými slovy je tato metrika citlivá na odlehlé hodnoty.

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

Kvůli přítomnosti druhé mocniny v sumě ztrácíme schopnost snadno interpretovat, co znamená hodnota celé metriky.

Root mean squared error (RMSE) je na první pohled velmi podobná MSE, jedná se totiž pouze o její odmocninu.

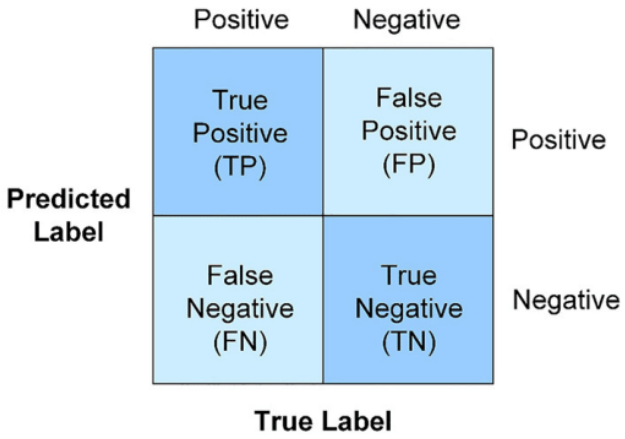
$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$$

Díky odmocnině se však dostáváme zpátky ke snažší interpretovatelnosti, ale stále si zachováváme penalizaci odhadu odlehlých hodnot. Proto také bývá častější volbou než MSE při statistické analýze přesnosti modelu.

Ve výsledku na nás je rozhodnutí, jak se chceme zachovat k odlehlým hodnotám, pokud se rozhodneme je neřešit, volíme MAE, pokud chceme jejich špatnou predikci penalizovat, volíme RMSE. [21]

3.1.2 Měření z hlediska kvality doporučování

V doméně vyhledání na webu a doporučování můžeme řešit další metriky našich systémů. Při predikci mohou nastat čtyři různé případy, které jsou pro lepší představu vyobrazeny na obrázku 3.1.



Obrázek 3.1 Obecná matice záměn (*angl. confusion matrix*) [22]

- True positive (TP) – v případě, že se predikce shoduje s očekávanou hodnotou a výsledek má být pozitivní,

- False positive (FP) – v případě, že predikujeme pozitivně, ale výsledek má být negativní,
- False negative (FN) – v případě, že predikujeme negativně, ale výsledek má být pozitivní,
- True negative (TN) – v případě, že se predikce shoduje s očekávanou hodnotou a výsledek má být negativní.

Na těchto případech pak závisí hodnota veličin *accuracy*, *precision* nebo *recall*. [23]

Accuracy Nejjednodušší a nejintuitivnější metrikou je *accuracy*. Vyjadřuje poměr položek, které systém správně klasifikoval, proti všem položkám:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}.$$

Udává nám, kolik z provedených doporučení bylo provedeno správně. Velmi často je ale zavádějící. Protože počítá přesnost nejen správně doporučených položek, ale zároveň i přesnost správně nedoporučených položek.

To znamená, že pokud například máme celkově 1000 položek na doporučení a doporučujeme pouze prvních 10, zbylých 990 doporučení je pouze negativních ($TN + FN$). Z těchto 990 doporučení je pak mnoho identifikováno jako správná predikce, protože jsme je nedoporučili.

S tímto problémem se snaží vyrovnat metriky *precision* a *recall*, protože se dívají pouze na poměr doporučování se správně pozitivně doporučenými položkami TP.

Precision vyjadřuje poměr položek, které byly správně doporučeny a položek, které byly celkově doporučeny:

$$precision = \frac{TP}{TP + FP}.$$

Udává nám, jak velká část z námi doporučených položek, byla doporučená správně.

Recall je poslední z jednodušších metrik pro určení kvality doporučování, kterou si v této sekci představíme. Vyjadřuje poměr položek, které byly správně doporučeny a položek, které měly být doporučeny.

$$recall = \frac{TP}{TP + FN}.$$

Udává nám, jak velkou část z položek, které jsme měli doporučit jsme systémem doporučili.

Normalized discounted cumulative gain (NGCD) je o něco složitější metrikou a musíme nejdříve projít vysvětlením několika pojmů než se dáme do definice vzorce.

▶ **Definice 3.1** (Kumulativní zisk [24]). *Mějme kolekci k dokumentů $D = (d_1, \dots d_k)$. Pro dotaz q mějme vektor $R = (r_1, \dots, r_k)$, který obsahuje hodnoty relevance k-tého dokumentu pro dotaz q v pořadí, jak dokumenty vrátil náš doporučovací systém.*

Kumulativní zisk definujeme:

$$CG[i] = \begin{cases} G[1] & i = 1 \\ CG[i-1] + G[i] & jinak \end{cases}$$

Výstupem je upravený vektor R, kde je na každé pozici součet relevancí všech předchozích a aktuální hodnoty.

Kumulativní zisk nám tedy říká, že čím vyšší hodnota je uložena ve vektoru, tím menší je šance, že se uživatel k dokumentu dostane, protože systém ho doporučil až po všech dokumentech, co se ve vektoru podle indexu nachází před ním. Hodnoty ve vektoru mají lineárně rostoucí tendenci, tomu se ale pokusíme zabránit, protože pravděpodobnost, že si uživatel zobrazí dokument, který mu je doporučený jako 100. nebo 300., je v obou případech téměř nulová. Toho docílíme tak, že každou hodnotu ve vektoru vydělíme logaritmem indexu. Zároveň funkce změní chování tak, že začne vracet vyšší hodnoty pro lepší doporučení a naopak pro horší doporučení nižší hodnoty. Získáváme tedy *discounted cumulative gain*: [24]

$$DCG[i] = \begin{cases} G[1] & i = 1 \\ DCG[i-1] + \frac{G[i]}{\log i} & jinak \end{cases}$$

Při normalizaci tohoto vztahu jde už jen o přidání vztahu, se kterým budeme DCG poměřovat. Tuto roli zastoupí *orákulum*[3], které pro zadaný dotaz vrací dokumenty seřazené podle relevance.

Nyní jsme schopni zadefinovat metriku NDCG, která se pro zjednodušení rovná:

$$NDCG = \frac{DCG}{IDCG},$$

kde DCG odpovídá seřezení, které vygeneroval náš doporučovací systém, a $IDCG$ odpovídá ideálnímu seřazení dokumentů. [25]

▶ **Definice 3.2** (Normalized discounted cumulative gain [26])**.** *Mějme množinu dotazů* $\mathcal{Q} = \{q^1, \ldots, q^n\}$*. Pro každý dotaz* q^k *mějme kolekci dokumentů* $\mathcal{D}^k = \{d_i^k, i = 1, \ldots, m_k\}$*, jejichž relevance k dotazu* q^k *jsou dány vektorem* $r^k = (r_1^k, \ldots, r_{m_k}^k) \in \mathbb{Z}^{m_k}$*.*

Dále mějme hodnotící funkci $F(d, q)$*, která na vstupu dostává dvojici dokument-dotaz a vrací číslo odpovídající relevanci dokumentu k tomuto dotazu*[4]*. Dále mějme číslo* j_i^k *udávající pozici dokumentu* d_i^k *v kolekci* $\mathcal{D}^k$ *pro dotaz* q^k*.*

Hodnota NDCG pro hodnotící funkci $F(d, q)$ *je rovna:*

$$\mathcal{L}(Q, F) = \frac{1}{n} \sum_{k=1}^{n} \frac{1}{Z_k} \sum_{i=1}^{m_k} \frac{2^{r_i^k} - 1}{\log(1 + j_i^k)}$$

Hodnota Z^k *hraje roli dalšího normalizačního faktoru.*

Tato metrika se používá v notaci NDCG@k, kde za k dosazujeme, na prvních kolik relevantních odpovědí na dotazy se má metrika zaměřit.

3.1.3 Pokrytí dat

V neposlední řadě nás také zajímá, jak velký prostor možných doporučení jsme prozkoumali, jestli jsme prošli všechny možnosti, co pro nás byly relevantní. Této metrice se říká *coverage* a počítá poměr položek, které byly systémem doporučeny, proti všem položkám, které systém mohl doporučit. [27]

▶ **Definice 3.3** (Míra pokrytí)**.** *Označme* I *množinu všech položek, které systém může doporučit a* I_p *množinu všech položek, které systém alespoň jednou doporučil. Potom je míra pokrytí vyjádřena vztahem:*

$$coverage = \frac{|I_p|}{|I|}$$

Při úpravách této míry se můžeme zabývat tím, kolikrát byly předměty z množiny I_p doporučeny, ale v rámci této práce to dělat nebudeme.

[3] Stroj „podivuhodných vlastností“, který na základě vstupu odpovídá nějakým výstupem.

[4] Tato funkce reprezentuje chování ideálního doporučovacího systému, který vrací dokumenty seřazené od nejrelevantnějšího po nejméně relevantní.

Tabulka 3.1 Nejlepších 5 modelů podle naměřené hodnoty MSE na nizozemských středních školách

Pořadí	User/Item based	Míra vzdálenosti	Počet sousedů	Hodnota metriky
1.	User	Euclid	200	0,36
2.	User	Euclid	25	0,38
3.	User	Euclid	200	0,39
4.	User	Euclid	1500	0,39
5.	User	UserThresh	neuvedeno	0,39

3.1.4 Časová náročnost

Vzhledem k velikosti zkoumané domény – předměty na jedné fakultě – jsme se časové složitosti rozhodli nevěnovat, protože na takto malé instanci to postrádá smysl.

3.2 Aktuální stav řešení problému

Samozřejmě tato práce není první, která se problematice doporučování předmětů věnuje. Jedna, která se zabývala předměty přímo na FIT ČVUT, byla publikována před šesti lety Ondřejem Novým. [1]

Další práce, která ale nemá s FIT ČVUT nic společného, je z Nizozemska. Autorka se snažila zmapovat místní střední školství a doporučovat předměty, které by si studenti měli zapsat. [2]

3.2.1 Doporučování nad předměty FIT ČVUT

Po bližším zkoumání se ukázalo, že práce sice pracovala s předměty na FIT ČVUT, ale k doporučování využívala jiné metody než ty, které byly vybrány pro tuto práci. Zároveň v sekci, která se zabývala modelováním doporučovacího systému, není uvedené vyhodnocení modelu podle žádných metrik typu *precision* nebo *recall*, ale pouze jako bodový graf přesnosti doporučení předmětu v prvních pěti doporučovaných předmětech. Z těchto důvodů není možné snadno srovnávat výsledky této práce s prací předchozí.

3.2.2 Doporučování pro středoškolské studenty v Nizozemsku

Tato práce je pro naši zajímavá tím, že se snaží využít k doporučování metodu k-nejbližších sousedů. Zároveň je provedena analýza doporučování pomocí metrik *average error*[5], *MSE*, *precision*, *recall*, *coverage* a dokonce byl měřen i čas, jak dlouho systém dotaz průměrně zpracovával.

Výsledné hodnoty, které byly systémem naměřeny, porovnáme s hodnotami měření, kterých dosáhne náš systém. Na základě tohoto porovnání nemůžeme říci, který systém je lepší, protože každý zpracovává jiná data, která mohou obsahovat různé množství šumu a chyb. Zároveň ale nebyly nalezeny žádné odkazy na zdrojový kód systému, tudíž jej na náš problém nelze aplikovat[6].

Hodnoty, se kterými budeme posléze porovnávat náš systém, nalezneme v tabulkách 3.1-3.4. Hodnoty jsou z práce [2].

V tabulce 3.1 vidíme míru vzdálenosti *UserThresh*. Podle zdrojů by mělo jít o metriku, která nebere v potaz k nejbližších sousedů, nýbrž všechny sousedy, kteří překročili zadaný práh podle metriky vzdálenosti. Konkrétně zde byla pro výpočet vzdáleností využita Euklidova vzdálenost.

[5] Tuto metriku v rámci práce měřit nebudeme.

[6] Kdybychom kód našli, není jisté, že by systém šel vyzkoušet, protože zkoumaná data jsou velmi odlišná.

Tabulka 3.2 Nejlepších 5 modelů podle naměřené hodnoty precision na nizozemských středních školách

Pořadí	User/Item based	Míra vzdálenosti	Počet sousedů	Hodnota metriky
1.	User	Euclid	25	0,52
2.	User	Cosine	25	0,52
3.	User	Cosine	25	0,52
4.	User	Euclid	200	0,51
5.	User	Pearson	1500	0,51

Tabulka 3.3 Nejlepších 5 modelů podle naměřené hodnoty recall na nizozemských středních školách

Pořadí	User/Item based	Míra vzdálenosti	Počet sousedů	Hodnota metriky
1.	User	Euclid	25	0,60
2.	Item	Manhattan	neuvedeno	0,56
3.	Item	Log likelyhood	neuvedeno	0,55
4.	Item	Cosine	neuvedeno	0,55
5.	User	Pearson	25	0,55

V tabulkách 3.3 a 3.4 si můžeme všimnout, že pokud šlo o item-based doporučování, není uveden počet sousedů, protože při tomto typu doporučování neřešíme, kolik podobných položek je v seznamu, nýbrž hledáme nejpodobnější předměty. Otázkou pak není *kolik*, ale *které*.

Zajímavostí na tabulce 3.4, která stojí za zmínění, je, že od 6. pozice dál mají největší míru pokrytí doporučovací systémy, které jsou založené na náhodném doporučování.

Tabulka 3.4 Nejlepších 5 modelů podle naměřené hodnoty coverage na nizozemských středních školách

Pořadí	User/Item based	Míra vzdálenosti	Počet sousedů	Hodnota metriky
1.	Item	Manhattan	neuvedeno	0,95
2.	Item	Log likelyhood	neuvedeno	0,95
3.	User	Euclid	25	0,95
4.	User	Pearson	25	0,93
5.	Item	Cosine	neuvedeno	0,92

Kapitola 4
Využité technologie

Před vlastní implementací a praktickou částí práce si krátce povíme o tom, jaké jsme zvolili technologie pro průzkum a předzpracování dat, implementaci systému a vyhodnocení jeho správnosti.

4.1 Python

Python[1] je velmi rozšířeným programovacím jazykem, který se hlavně v oblasti datové analýzy a strojového učení velmi hojně využívá. Jedná se o dynamicky typovaný programovací jazyk z třídy vysokoúrovňových jazyků. Zároveň se jedná o otevřený a volně šiřitelný software s mnoha knihovnami, které nám usnadňují práci. V rámci práce využíváme verzi 3.10.

4.1.1 NumPy

NumPy[2] je velmi hojně využívanou knihovnou. Knihovna nám poskytuje infrastrukturu pro matematické operace s vektory nad vícerozměrnými poli. Její největší výhodou je, že je částečně psaná v jazyku C, což vede k vyšší rychlosti výpočtů než kdyby byla implementována pomocí Pythonu. V rámci práce budeme pracovat s verzí 1.24.2. [28] [29]

4.1.2 Pandas

Další knihovnou, kterou si představíme je Pandas[3]. Knihovna se využívá při předzpracování a analyzování dat. V rámci ní často pracujeme s datovou strukturou zvanou *dataframe*. Knihovna provádí matematické operace pomocí NumPy. V práci budeme používat verzi 1.5.3. [30]

Dataframe si můžeme představit jako tabulku, která má ve sloupcích příznaky a v řádcích jednotlivé záznamy, které nabývají pro každý sloupec nějaké hodnoty. Dataframy se dají načítat a ukládat v různých formátech jako například csv nebo jako excel tabulku.

[1] `https://www.python.org/`
[2] `https://numpy.org/`
[3] `https://pandas.pydata.org/`

4.1.3 Scikit-learn

Knihovna scikit-learn[4] obsahuje obrovské množství funkcí, pro předzpracování dat, strojové učení, měření kvality modelů a mnoho dalšího. Je postavená nad NumPy a SciPy pro rychlé výpočty. Budeme pracovat s verzí 1.2.2. [12]

4.2 Repsys

Poslední technologií, kterou zmíníme, je framework RepSys. Jeho implementací se inspiruje část naší práce, která se zabývá doporučováním. Původně jsme jej chtěli využít pro celé měření doporučování. Framework se však ukázal být pro naše až zbytečně obsáhlý. Proto jsme se rozhodli se pouze inspirovat některými částmi kódu pro naší vlastní implementaci. [31]

Konkrétně nám kód ukázal, jak přistupovat k doporučování pomocí různých modelů, a jakým způsobem se dá měřit kvalita těchto modelů.

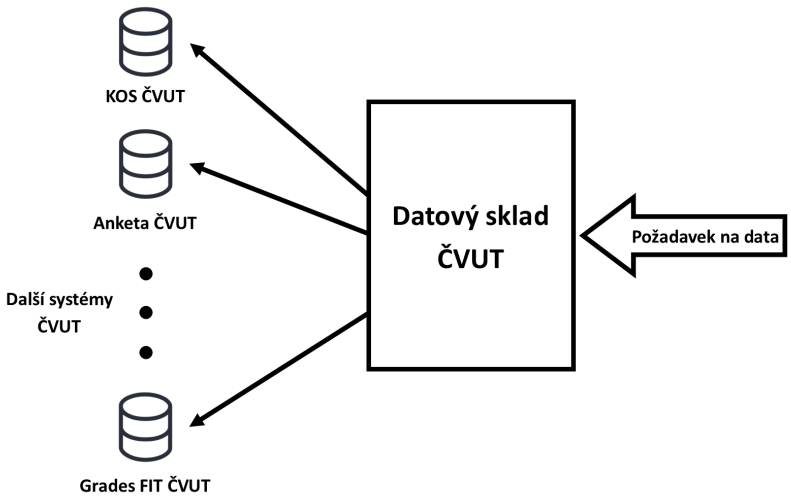
[4] `https://scikit-learn.org/`

Kapitola 5
Dostupná data

Hlavní motivací této práce je usnadnit studentům výběr filtrováním dat ze školních systémů, které si postupně představíme. Půjde o univerzitní systémy, jejichž data shromažďuje Datový sklad ČVUT. Konkrétně se podíváme na KOS ČVUT a Anketu ČVUT.
Následně si popíšeme i samotná data, která byla pro práci k dispozici, a zdůrazníme několik problémů, které data na první pohled obsahují a které se budeme snažit odstranit v rámci kapitoly Předzpracování dat 6.

5.1 Zdroje dat

Systémů, které uchovávají data studentů, vyučujících a předmětů je na ČVUT mnoho. Bylo potřeba zanalyzovat, jaká data nám může který školní systém poskytnout, a vyhodnotit jejich využitelnost pro účely práce.



Obrázek 5.1 Nákres procesu získání dat z informačních systémů ČVUT

5.1.1 Datový sklad ČVUT

Jedná se o oddělení na ČVUT, které má za úkol shromažďovat data univerzitních systémů na jednom místě. Velké využití sklad nalézá v době psaní zpráv o dění na univerzitě, kdy garanti událostí potřebují data, ze kterých v těchto reportech čerpají. Pracovníci datového skladu tak mají přístup do databází většiny školních systémů, ze kterých dostávají a čistí potřebná data tak, aby byla garantům snadno srozumitelná a ušetřilo jim to práci. Tento proces je vyobrazen na obrázku 5.1.

Hlavní systém, ze kterého datový sklad těží, je KOS ČVUT. Pracovníci mají přístup i do dalších systémů jako je například Anketa ČVUT, Grades a mnoho dalších. Pro účely této práce byla využita data z *KOS ČVUT* a z *Ankety ČVUT*.

5.1.2 KOS ČVUT

KOS ČVUT je celouniverzitní systém, ve kterém probíhá téměř veškerá aktivita studentů ohledně studia. Probíhá zde zápis nebo termínů zkoušek či zápočtových testů, uchovávají se zde studijní výsledky, kontrolují se požadavky pro postup ve studiu a mnoho dalšího.

Studenti tak mohou zjistit, jaké předměty jsou v dalším semestru vypsány, jaká je jejich kapacita a obsazenost, co se v předmětech probírá. Tyto veličiny jsou však mimo rozsah této práce. Pro její účely nás bude zajímat, jaké předměty si studenti zapisují, jak se jim daří je dokončovat, případně, jaké vyučující si při absolvování předmětu vybírali. Dále máme k dispozici informace o jednotlivých předmětech, které se vypisují.

5.1.3 Anketa ČVUT

Anketa ČVUT je dalším informačním systémem ČVUT. Skrz ni po semestru probíhá komunikace směrem od studentů k vyučujícím. Studenti hodnotí absolvované předměty známkami jako ve škole a pokud chtějí, mohou přiložit textovou odpověď. Vyučující si ji po jejím vyplnění přečtou a na některé komentáře odpoví.

Studenti se tak mohou dočíst, na co si dát v předmětu pozor, pokud si jej zapíší, jak je vyučovaný nebo náročný a dokonce si podle ní mohou vybrat vhodného vyučujícího.

5.2 Datové soubory

Z datového skladu ČVUT byly pro práci k dispozici 4 datové soubory, jejichž atributy budou v následující sekci postupně popsány. Data z KOSu, kterým se budeme věnovat, jsou od zimního semestru 2017/2018 do letního semestru 2021/2022.

Dále byla k dispozici data z Ankety ČVUT, jejichž bližší popis je také k nalezení níže. Tato data byla dodána vždy po dvou souborech za semestr – jeden pro předměty, druhý pro vyučující. Protože Anketa ČVUT je jeden z novějších systémů, obsahuje data pouze od letního semestru 2017/2018, a tak nám oproti datům KOSu chybí zimní semestr tohoto akademického roku. Vzhledem k charakteru dat z KOSu ale proběhne nad daty z Ankety agregace, díky které tento chybějící semestr ničemu nevadí.

5.2.1 Klasifikace studentů

Obsahuje informace o tom, který student v kterém semestru měl zapsaný který předmět. Zároveň obsahuje informace o zakončení předmětu tímto studentem.

studium_id – identifikátor studia[1],

predmet_id – identifikátor předmětu,

semestr_id – identifikátor semestru,

zakonceno – atribut nabývá dvou hodnot:

- *1* – pokud student předmět úspěšně dokončil,
- *NaN* – pokud student předmět nedokončil nebo má známku F,

zapocteno – atribut nabývá tří hodnot:

- *Z* – pokud se studentovi podařilo získat zápočet,
- *N* – pokud student zápočet nezískal,
- *NaN* – v případě, že rámci předmětu zápočet zápočet získat nelze,

znamka – jakou student z předmětu dostal známku. Pokud student není klasifikován, hodnota je nevyplněná,

cislo_pokusu – na kolikátý pokus u zkoušky student dostal známku.

5.2.2 Zapsaní vyučující

Tabulka pro každý zápis předmětu každého studenta udržuje, který vyučující vyučoval paralelku, kterou měl student zapsanou.

studium_id – identifikátor studia,

predmet_id – identifikátor předmětu,

semestr_id – identifikátor semestru,

typ_paralelky – atribut nabývá tří hodnot podle toho, o jakou hodinu se jednalo:

- *P* – značí přednášku,
- *C* – značí cvičení,
- *L* – značí laboratoř[2],

cislo_paralelky – jaké číslo měla paralelka,

ucit1_cele_jmeno – jméno prvního vyučujícího, který paralelku vyučoval,

ucit2_cele_jmeno – jméno druhého vyučujícího, který paralelku vyučoval. Pokud paralelka neměla druhého vyučujícího, sloupec zůstává prázdný.

[1] Obvykle má v jednu chvíli student právě jedno studium_id, pokud je na škole po několikáté, je mu přiřazeno jiné studium_id než v předchozím působení na fakultě.

[2] U některých předmětů se toto názvosloví tranformuje, aby vyhovovalo stylu výuky – např. PA1 (*Programování a algoritmizace 1*) měl přednášky (označeno *P*), proseminář (označeno *C*) a cvičení (označeno *L*).

5.2.3 Předměty FIT ČVUT

V tabulce se vyskytují všechny předměty, které byly na FIT ČVUT od zimního semestru 2017 vyučovány. Nalezneme zde předměty jak ze staré, tak i z nové akreditace. Pro každý předmět tabulka ještě obsahuje informace o tom, v jakém semestru je vyučován a mnoho dalšího.

predmet_id – identifikátor předmětu,

kod_predmetu – kód předmětu, který zároveň může také sloužit jako identifikátor,

nazev_predmetu – český název předmětu

nazev_predmetu_en – anglický název předmětu,

pocet_kreditu – počet ECTS kreditů, které student dostane za úspěšné dokončení předmětu,

jazyk_vyuky – jazyk, ve kterém probíhá výuka předmětu, atribut nabývá tří hodnot:

- *CS* – předmět je vyučován česky,
- *EN* – předmět je vyučován anglicky,
- *OO* – tento jev nastává pouze u předmětu *Magisterská práce* – nejspíš jde pouze o chybu v tomto datovém záznamu,
- *NaN* – u předmětů jako je tělocvik, uznávání kreditů za výjezd nebo dalších kurzů, kde na vyplnění jazyka zadavatel nejspíš zapomněl,

zpusob_zakonceni – jakým způsobem se dá předmět zdárně dokončit, nabývá hodnot:

- *Z,ZK* – předmět je zakončený zkouškou, které předchází zápočet,
- *KZ* – předmět je zakončený klasifikovaným zápočtem,
- *ZK* – pro úspěšné dokončení stačí absolvovat pouze zkoušku,
- *Z* – předmět vyžaduje pouze zisk zápočtu pro absolvování,
- *NIC* – zpravidla jde o zvláštní předměty, které musí uznávat studijní oddělení – např. výjezdy, fiktivní předmět…

typ_predmetu – jakého typu je předmět, konkrétně jde o hodnoty nastíněné v úvodu práce,

typ_semestru – ve kterém semestru je předmět vyučován, nabývá 6 hodnot:

- *Z* – předmět je vypisován pouze v zimním semestru,
- *L* – předmět je vypisován pouze v letním semestru,
- *Z,L, L,Z, *, NaN* – všechny tyto hodnoty popisují, že předmět je vypisován v letním i zimním semestru.

5.2.4 Studenti

V tabulce se nachází údaje o aktuálních studentech i o absolventech. Pro každého studenta můžeme najít, jakou studoval specializaci.

studium_id – identifikátor studia,

rocnik – ve kterém ročníku se student teď nachází nebo ve kterém ročníku byl v době ukončení studia,

studuje – atribut, který nabývá dvou hodnot:

- *K* – u studentů, kteří už ukončili studium[3],
- *S* – pokud studenti stále studují,

ukonceni_zpusob – jde o číselné označení způsobu ukončení studia:

- pokud *studuje* je rovno K, tento sloupec drží hodnotu 1,
- v případě, že *studuje* je S, tento sloupec má NaN,

jazyk_vyuky – jazyk, ve kterém student studuje, atribut nabývá 3 hodnot:

- *CS* – když student studuje v češtině,
- *EN* – když student studuje v angličtině,
- *NaN* – pokud u studenta není vyplněno, v jakém jazyce studuje,

forma_studia – nabývá přesně dvou hodnot:

- *P* – pro studenty, kteří studují prezenčně,
- *K* – pro studenty, kteří mají kombinované studium,

nazev_programu – název studijního programu – obvykle odpovídá akreditaci, kterou student studuje/studoval,

zkratka_oboru – zkrácený zápis specializace[4], kterou si student zapsal,

nazev_oboru – celý název specializace, kterou si student zapsal,

zamereni – tento atribut upřesňuje atribut *zkratka_oboru* – zpravidla jde o doplnění přesného programu podle akreditace, v níž student studuje (nebo v případě specializace *Webové a softwarové inženýrství* v minulé akreditaci jde o rozdělování specializace na zaměření počítačová grafika, webové inženýrství a softwarové inženýrství).

5.2.5 Anketa ČVUT – Předměty

Soubory obsahují informace, které jsou velmi snadno dostupné i po přihlášení do Ankety, ve které jsou pomocí GUI uživatelsky přívětivě zobrazena. Pro každý předmět existuje mnoho atributů, které přidáme do tabulky předmětů, kterou jsme v rámci práce dostali z Datového skladu ČVUT.

department_code – identifikátor katedry, pod kterou předmět spadá,

id_course – identifikátor předmětu, kterého se záznam týká,

course_code – kód předmětu, který zároveň může také sloužit jako identifikátor,

course_name – název předmětu (česky nebo anglicky podle toho, v jakém jazyce je vyučován),

completion – jakým způsobem se dá předmět zdárně dokončit (viz. 5.2.3),

range – rozsah předmětu, neboli kolikrát týdně je předmět vyučován,

credits – počet ECTS kreditů, které student dostane za úspěšné dokončení předmětu,

RA, RB, RC, RD, RE, RF – každý z těchto sloupců říká, kolik studentů v příslušném semestru dostalo kterou známku,

num_stud_subscribed – udává počet studentů, kteří měli předmět zapsaný,

[3] Ne nutně úspěšně.
[4] Obor je pozůstatkem z minulosti, v dnešní době používáme slovo specializace.

num_stud_finished – počet studentů, kteří předmět úspěšně dokončili[5],

num_survey_filled – počet studentů, kteří vyplnili Anketu,

is_evaluated – binární příznak, který nám říká, jestli tento předmět v Anketě má alespoň jedno hodnocení, v případě, že nabývá hodnoty *N*, všechny následující sloupce obsahují NaN,

num_diff_qest_answered – počet odpovědí na různé otázky – pokud bylo odpovězeno na všechny známkové i textové odpovědi, nabývá hodnoty 3, pokud na některou z otázek v rámci předmětu nebylo odpovězeno, nabývá hodnoty 2,

num_answers – počet různých odpovědí, neodpovídá součtu *num_value_answers* a *num_text_answers*, protože existují studenti, kteří vyplnili jak známky tak textové odpovědi, a ti se započítávají jako jeden,

num_value_answers – počet známek, které předmět v Anketě od studentů získal[6],

avg_value – průměrná známka, kterou předmět dostal od studentů,

num_text_answers – počet textových odpovědí, které studenti k předmětu vyplnili,

VA, VB, VC, VD, VE – rozložení jednotlivých známek, kterými studenti ohodnotili předmět.

Ve sloupcích *num_survey_filled, num_diff_qest_answered, num_answers, num_value_answers* a *num_text_answers* existují nekonzistence podle toho, kolik studenti vyplnili kterých otázek, ale pro účely této práce budeme z těchto příznaků počítat pouze s počtem vyplněných dotazníků, tudíž nám to nevadí.

5.2.6 Anketa ČVUT – Vyučující

Stejně jako u informací z Ankety o předmětech tyto informace nejsou studentům skryté a je třeba tato data vhodně zagregovat a využít. Podobně jako u předmětů budou vybrány užitečné atributy, které budou přidány k datům, která byla k dispozici z Datového skladu ČVUT.

id_teacher – číselný identifikátor vyučujícího,

username – školní uživatelské jméno vyučujícího,

full_name – celé jméno vyučujícího i s tituly

teacher_department_code – identifikátor katedry, pod kterou je vyučující zaměstnán,

num_courses_evaluations – počet předmětů, který vyučující vyučoval,

num_value_answers – počet známek, které vyučující od studentů dostal (pokud učil více předmětů, jedná se o agregovaný počet všech známek, co vyučující dostal),

avg_value – průměrná známka, kterou vyučující dostal od studentů,

num_text_answers – počet textových odpovědí, které vyučující obdržel,

num_answers – stejně jako u tohoto atributu v předmětech se jedná o počet různých odpovědí, neodpovídá součtu *num_value_answers* a *num_text_answers*, protože existují studenti, kteří vyplnili jak známky, tak textové odpovědi, a ti se započítávají jako jeden,

A, B, C, D, E – rozložení jednotlivých známek, kterými studenti ohodnotili vyučujícího.

[5] Dostali známku A–E.

[6] Každý student může předmětu dát až dvě známky.

5.3 Problémy v datech

Problémy a chyby se v datech nachází kvůli jejich špatnému nebo nedbalému zadávání. Někdy jde pouze o změnu datových typů, doplnění chybějících hodnot nebo správné zakódování do reprezentace, která je strojově snadno čitelná. Těmto problémům se budeme více věnovat v kapitole 6.

Na první pohled ale byly v předešlé kapitole objeveny problémy, jejichž řešení vyžaduje rozsáhlejší analýzu a složitější řešení.

5.3.1 Nové předměty

Mnoho předmětů po změně akreditace, která proběhla v roce 2021, zůstalo obsahově stejných, pouze se změnil jejich kód nebo název. Tabulka s předměty bude předělána tak, aby o těchto předmětech byl veden pouze jeden záznam.

Další předměty si prošly změnou, přeskládáním sylabu, nebo roztržením na více menších předmětů. Jedná se hlavně o povinné matematické předměty. Sice nechceme, aby systém tyto předměty doporučoval, protože je musí každý student odstudovat povinně, ale jejich data budeme chtít využít při určování sobě podobných studentů.

Řešení může být více. Kdybychom se rozhodli se tímto problémem nezabývat, můžeme ztratit část informace, na základě které se dají studenti spojovat a algoritmus by bylo možné aplikovat napříč stejnou akreditací. Dále můžeme vyzkoušet předměty s překrývajícími se sylaby sloučit do jednoho velkého předmětu a známku z něj určit průměrem.

Další problém způsobuje vytváření nových předmětů. Tyto předměty nejspíš nebude algoritmus schopný doporučovat, protože si ho zatím žádný student nikdy nezapsal. S tím v rámci předzpracování dat nejsme schopni nic udělat.

Kdybychom se chtěli pokusit je zapsat některým studentům „jako že je měli“, nikdy to nebude správně, protože nevíme, jestli by tyto předměty studenty oslovily. To by do dat uměle zaneslo odchylku, což by nikdy nemělo být cílem.

Nejlepší možnost, jak naložit s novými předměty, bude vyčkat, až si je nějací studenti zapíší a poté při aktualizaci systému pro další ročníky už bude algoritmus schopný doporučovat i tyto předměty, protože jejich zápis bude promítnut v datech.

5.3.2 Využití informací o vyučujících

V rámci předzpracování dat bude třeba spojit vyučující z tabulky z KOS s vyučujícími z Ankety. V KOSu však nemáme k dispozici identifikátor, na základě kterého bychom tyto dvě tabulky mohli snadno propojit, a vytvořit tak tabulku, kde bude výpis všech vyučujících spolu s jejich průměrným hodnocením v anketě.

Jediným identifikátorem v datech z ankety o vyučujících (5.2.2) je jméno s tituly, které nám v případě duplicity jmen některých vyučujících bude dělat problém.

Kapitola 6
Příprava dat

Při strojovém učení se běžně setkáváme s tím, že data vypadají jinak než potřebujeme pro následné zpracování, a proto je třeba je předzpracovat. Předzpracování dat by nijak zásadně nemělo měnit informaci, kterou data nesou. Často jde pouze o změnu datových typů, doplnění chybějících hodnot, normalizaci, výběr z dat nebo tvorbu nových sloupců kombinací jiných... V následující kapitole si popíšeme, jak jsme se s předzpracováním dat z kapitoly 5 vypořádali, proč jsme se tak rozhodli a jak vypadají data, která jsou výstupem našeho předzpracování. Vše, co budeme popisovat v této kapitole, je naimplementováno v přiložených souborech preprocessing.ipynb a prepare_test_data.ipynb.

6.1 Výběr datových souborů

Naším cílem je z dostupných dat vytvořit jeden datový soubor, ve kterém budou informace a studentech, předmětech a klasifikaci každého studenta z jím zapsaných předmětů.

6.1.1 Klasifikace

Základem celého řešení bude soubor s klasifikacemi známý z části 5.2.1. Ten obsahuje číselné identifikátory studentů a předmětů, podle kterých proběhne spojení všechno těchto tří datasetů do jednoho.

V tomto souboru jsme objevili, že existují záznamy o tom, že student měl zapsané předměty, které měly označený identifikátor jako NaN. To si můžeme intepretovat tak, že student studoval předmět, který neexistuje. Takové záznamy z tabulky odstraníme.

Tento soubor sám o sobě neříká více informací, ale jeho dalším předzpracováním se budeme zabývat po spojení se studenty a předměty, na základě kterých můžeme vyvozovat další závěry, podle kterých budeme rozhodovat, co se má stát dál.

6.1.2 Studenti

Datový soubor se studenty z části 5.2.4 nejdříve zpracujeme a následně jej spojíme s daty o klasifikaci. Data o studentech obsahují mnoho informací o každém studentovi, kterým je vhodné se před dalším zpracováním věnovat pozornost.

Údaje o aktuálním stavu studia V souboru se nachází příznaky *studuje* a *ukonceni_zpusob*, které vyjadřují stejnou skutečnost ve dvou různých sloupcích. Příznak *studuje* zakódujeme na hodnoty:

- 0 – pokud je studium ukončené,
- 1 – pokud student studuje.

Příznak *ukonceni_zpusob* z dat odstraníme.

Příznak *prezencni_studium* také zakódujeme pomocí nul a jedniček tak, abychom byli hodnotou schopni odpovědět na otázku, jestli je student na prezenčním nebo kombinovaném studijním programu.

Tento příznak využijeme k výběru dat na měření metrik doporučovacího systému. Pro měření bude třeba mít k dispozici pouze studenty s velkým počtem odstudovaných předmětů, protože těmto studentům v rámci měření předměty skryjeme a poté je budeme porovnávat s výstupem systému, abychom tak mohli určit, jestli systém určil předmět správně.

Specializace a zaměření studenta Prvně ve sloupcích *zkratka_oboru*, *nazev_oboru* a *zamereni* nastavíme chybějící hodnoty na *not chosen*. Každý, kdo úspěšně dokončí studium, musí mít zapsanou specializaci. Proto předpokládáme, že studenti, kteří specializaci nemají zapsanou, ještě nemají vybráno.

Následně pomocí zkratky oboru vybereme pouze studenty, kteří jsou vedeni v prezenčním studiu českého bakalářského stupně nebo nemají vybraný obor.

Nad těmito vybranými studenty pro každý název oboru vypíšeme všechna zaměření, která se v datech nachází. Tato zaměření ukládáme do slovníku, který je indexován podle názvů specializací z nové akreditaci. Po této operaci máme dostupný překlad všech různých zkratek zaměření na 12 specializací[1].

Nový příznak *specializace* využijeme při hledání předmětů, které jsou pro studenta povinné díky zapsanému oboru. Tyto předměty totiž budou využity k doporučování, pokud je student už absolvoval. Zároveň je ale chceme odfiltrovat z výsledného doporučení, protože student si je stejně bude muset během studia zapsat.

Tabulka studentů aktuálně Nyní máme tabulku obsahující pouze studenty, kteří studují prezenčně na bakalářském stupni v českém jazyce. Zároveň tabulka obsahuje údaj o specializaci, kterou má student zapsanou. V následujících fázích předzpracování se ji pokusíme pomocí číselných identifikátorů studentů spojit s tabulkou klasifikací.

6.1.3 Předměty

Data týkající se předmětů se skládají z několika souborů. Prvním souborem, který budeme zpracovávat, jsou údaje ze systému KOS ČVUT, který známe z části 5.2.3. Tato data poté spojíme s daty z Ankety ČVUT, která byla popsána v části 5.2.5. Nakonec předměty rozdělíme podle úrovně studia, pro kterou jsem vypisovány.

Spojení dat předmětů z KOS ČVUT a Ankety ČVUT Soubory s daty z Ankety ČVUT jsou rozdělené podle semestru, kterého se anketa týká. Z těchto souborů nejdříve vybereme pouze potenciálně relevantní příznaky, poté je všechny složíme dohromady do jednoho velkého souboru a ten nakonec spojíme se souborem předmětů z KOS ČVUT.

V každém souboru jsme si ponechali pouze identifikátor předmětu, kód předmětu, statistické údaje o známkách, které z předmětů v daném semestru byly uděleny, a průměrné číselné hodnocení předmětu v anketě spolu s počtem hlasujících studentů.

Na základě identifikátoru jsme pak spojili všechny soubory do jednoho, kde jsme přepočítali údaje o známkách studentů a o průměrném hodnocení předmětu v anketě.

[1] Na FITu je 10 specializací, další byla v datech vytvořena ze staré akreditace – Informační technologie – a poslední specializaci máme pro studenty, kteří ji ještě v systému zapsanou.

Takový soubor jsme opět na základě číselného identifikátoru předmětu spojili se souborem dat z KOS ČVUT. Po propojení se ukázalo, že ne každý předmět, který je v KOS ČVUT, je zároveň v Anketě ČVUT. Zpravidla šlo o předměty z nové akreditace, které ještě nebyly vyučovány. U záznamů, kde toto nastalo, jsme nastavili chybějící číselné hodnoty na 0. Zároveň se ukázalo, že existuje i předmět, který je v Anketě ČVUT, ale data z KOS ČVUT jej neobsahují. Protože ho měli zapsaní za celou dobu pouze tři studenti, nemá smysl se jím v doporučování zabývat, a tak ho odstraníme.

Data z ankety se dají využít při doporučování tak, že nám říkají globální oblíbenost předmětů. Pokud na základě známek doporučíme studentovi, můžeme využít informaci o průměrném hodnocení v anketě pro lepší filtraci dat po predikci.

Předzpracování předmětů a odstraňování zbytečných údajů Předmětům, kterým chybí údaj o počtu kreditů, tento příznak doplníme nulou.

Na základě kódu předmětu správně určíme jazyk, ve kterém je předmět vyučovaný. Zjistili jsme, že pokud na základě kódu nejsme schopni určit vyučovaný jazyk, získáme předměty, nemá smysl doporučovat, a tak je z dat odstraníme. Konkrétně jde o fiktivní předměty, které zapisuje pouze studijní oddělení, nebo o češtinu pro studenty ze zahraničí.

Následně odstraníme předměty, které mají *zpusob_ukonceni* nastavený na *NIC*. Jde o předměty, které navyšují počty kreditů.

Dále se ukázalo, že dobrým filtrem pro odstranění předmětů z dat je i počet kreditů. Pokud předmět má hodnotu více než 10 kreditů, jedná se buď o uznání zahraničních předmětů nebo závěrečné práce. Ani jeden z těchto předmětů při doporučování nebudeme využívat.

Poslední příznak, který budeme zkoumat pro všechny předměty najednou, je typ předmětu. Při doporučování po předmětech vyžadujeme rozdělení do tří kategorií:

- P – povinné předměty, které musí splnit každý student,
- PO – oborové předměty, které musí splnit každý student, který chce absolvovat studium některého oboru,
- V – volitelné předměty, které si student zapisuje, jak chce.

Počet různých kategorií nejdříve snížíme odstraněním středníku u hodnot, které jím končí. Naším cílem v této části je přesně oddělit povinné předměty, které z doporučování budeme odstraňovat při filtraci, od ostatních předmětů. Ostatní označíme za oborové nebo volitelné podle toho, jestli se v hodnotě příznaku *typ_predmetu* nachází *PO* (případně *PZ*) nebo *V*.

Nakonec jsme z dat odstranili předměty, kterým po této proceduře zůstal příznak *typ_predmetu* nevyplněný. Konkrétně se jednalo o některé předměty pro tělesnou výchovu a navýšení počtu kreditů za náročnost předmětu.

Tato opatření vedou k čistějšímu datasetu z pohledu čitelnosti příznaků a zároveň jsme díky nim z odstranili 38 předmětů, které by byly pro doporučovací systém přítěží.

Rozdělení podle programu Nyní předměty rozdělíme podle kódu na:

- prezenční bakalářské učené česky,
- prezenční magisterské učené česky,
- prezenční bakalářské učené anglicky,
- prezenční magisterské učené anglicky,
- kombinované bakalářské učené česky,
- ostatní.

V kategorii *ostatní* máme humanitní předměty z dob, kdy jejich kód začínal *FI-*, a dále různé kurzy tělesné výchovy. Humanitní předměty podle kódu oddělíme a přidáme k předmětům do programů, které jsou učené v češtině.

Tento způsob předzpracování volíme, protože by se mohlo stát, že bychom studentům v bakalářském programu doporučovali magisterské předměty nebo obráceně[2]. Dále se může stát, že pro větší datasety (typu bakalářský prezenční učený česky) jsou úspěšnější jiné hodnoty hyperparametrů modelu než pro menší datasety. Nakonec složitost výpočtu nejbližších sousedů roste s množství zpracovávaných dat, proto se hodí mít více menších datasetů[3].

Dále se budeme zabývat pouze předměty prezenčního bakalářského programu učeného v českém jazyce.

Stejné předměty vs. jiná akreditace Nyní v předmětech máme více záznamů týkajících se stejných předmětů, které přechodem na novou akreditaci pouze změnily název nebo kód. Pokusíme se tyto předměty identifikovat a více záznamů spojit do jednoho. Zároveň výstupem tohoto kroku musí být seznam překladů různých identifikátorů na právě jeden, který zvolíme, abychom mohli snadno provést spojení tabulky předmětů s tabulkou klasifikací.

Náhrada identifikátorů proběhne ve dvou fázích – ruční a automatické. Nejdříve ručně vybereme předměty, které přechodem na novou akreditaci změnily název, takovým způsobem zpracováváme tři vzory předmětů:

- předměty úplně změnily název i kód (např. *BI-CAO* → *BI-TZP*),
- předměty změnily název, ale kód zůstal stejný (např. *BI-GIT*),
- předměty se vypisují vícekrát, aby si je student mohl zapsat vícekrát, tudíž mají různé názvy (např. *BI-ACM*).

Poté spustíme automatický mechanismus, který seskupí předměty podle názvu[4]. Z těchto skupin vždy vytvoříme jeden předmět. Tomuto předmětu dáme součet všech známek, které v něm byly uděleny, přepočítáme vážené průměrné hodnocení, které si s sebou předmět nese z dat z Ankety ČVUT a přiřadíme kategorické hodnoty prvního z těchto předmětů.

Tímto krokem odstraníme 67 předmětů a zároveň tak předejdeme tomu, že byly předměty z různých akreditací systémem hodnoceny různě, i když jsou ve skutečnosti stejné.

Tabulka předmětů aktuálně Výstupem této části je tabulka předmětů, které jsou dostupné na prezenčním bakalářském stupni v českém jazyce. Tabulka je zredukovaná o mnoho předmětů, které byly vyhodnoceny jako zbytečné pro doporučování. Mezi tyto záznamy spadají uznání kreditů za výjezd, český jazyk pro ukrajinské uprchlíky nebo předměty pro odevzdání závěrečné práce. Zároveň je zredukovaná o předměty, které při změně akreditace změnily identifikátor, ale jinak zůstaly stejné.

Další důležitá struktura z této části je tabulka překladů, která obsahuje mnoho číselných identifikátorů předmětů a jejich ekvivalent v novém systému identifikátorů.

6.2 Spojení předzpracovaných datových souborů

Na konci této části budeme mít jeden velký dataset, který bude obsahovat záznamy o tom, který student si zapsal který předmět a jakou si z něj odnesl známku, případně jestli se mu předmět nepodařilo splnit.

[2] I když tento problém by se nejspíš řešil sám malou podobností studentů různých programů.

[3] Vzhledem k velikosti dat se tento problém ale také dá zanedbat.

[4] Seskupení podle kódu by vyžadovalo odstranění přípony *.21* a nepokrylo by předměty, které mají kód končící *.1*.

Spojení klasifikace s předměty Nejdříve předzpracujeme dataset klasifikací tak, že z něj vybereme záznamy o předmětech, pro které jsme schopni najít identifikátor v tabulce překladů identifikátorů. Tím vybereme pouze předměty českého bakalářského prezenčního studia.

Tento dataset poté spojíme s předměty na základě číselného identifikátoru předmětu. Zjistili jsme, že mezi klasifikacemi neexistují záznamy o 28 předmětech, tím pádem množství zpracovávaných předmětů opět klesá.

Další filtrací, kterou nyní můžeme provést, je filtrace předmětů, které mělo za celou dobu zapsáno málo studentů a jejich doporučení tak také postrádá smysl. Tuto hranici jsme určili na 20 studentů. Odstraníme tak 46 předmětů, které v celém datasetu mělo zapsáno méně než 20 studentů.

Spojení datasetu se studenty Opět provedeme spojení na základě číselného identifikátoru. Ze všech studentů, které máme k dispozici, má záznam o klasifikaci pouze 2 910 z nich.

Při doporučování chceme zahrnout i studenty, kteří jsou už absolventy, protože z jejich průchodu studiem můžeme vyčíst, jaké volitelné předměty si při studiu volili. Narazili jsme na 345 studentů, kteří už nestudují, ale zároveň měli zapsáno méně než 20 předmětů.

6.3 Finální předzpracování dat

Po všem uplynulém předzpracování máme nyní k dispozici dataset, který má 58 627 záznamů, které obsahují informace o 2565 studentech a 139 předmětech. Protože už máme všechna potřebná data u sebe v jedné tabulce, můžeme se pustit do pokročilejšího předzpracování.

Zakódování známek Známka je jediná hodnota, která mění svou hodnotu při zápisu stejného předmětu více studenty. V doporučovacím systému se nám hodí reprezentovat ji jinými hodnotami než A–F. Příznak má výhodu, že se jedná o ordinální proměnnou[5], a tak jej můžeme snadno zakódovat číslem. Při kódování známku normalizujeme na interval [0,1], tudíž zakódování vypadá následovně:

- A = 1,0,
- B = 0,83333333,
- C = 0,66666667,
- D = 0,5,
- E = 0,33333333,
- F = 0,16666667,
- nezapsáno = 0.

Hodí se nám znát rozdíl mezi známkou F a tím, že si student předmět nezapsal, protože můžeme usuzovat, že se studentovi předmět nelíbil. Bohužel nemáme jinou metriku, jak předměty doporučovat, a kvůli anonymizaci Ankety ČVUT nejsme schopni ověřit, jestli je to pravda.

V datasetu se ale nachází i předměty, jejichž zakončením nelze získat známka. U takových předmětů ohodnotíme studenty známkou *A* v případě jej splnili[6], v opačném případě do známky dosadíme *F*.

Po těchto úpravách se v datech stále nachází známky bez vyplněné hodnoty. Po výpisu hodnot zjistíme, že jde o velké povinné předměty. Na základě toho usuzujeme, že jde o studenty, kteří předmět nedokončili, proto jim doplníme ekvivalent známky *F*.

[5] Hodnoty mezi sebou mají dané pořadí.
[6] Ve sloupci *zakonceno* je hodnota 1.

Nakonec se ještě ukázalo, že v datech máme některé záznamy duplikované. Některé předměty student absolvoval vícekrát, to se může stát buď při nedokončení předmětu a následném zápisu v dalším semestru nebo u tělesné výchovy, kdy si studenti mohou zapsat kurz vícekrát.

Touto úpravou do datasetu přidáváme možnost vyhodnocovat podobnost studentů jiným způsobem než tak, že student měl zapsaný předmět – a to podle známky.

Povinnost oborových předmětů Posledním procesem, kterým data před rozdělením na trénovací a testovací množinu projdou, je přidání sloupce *droppable*, který nám říká, jestli záznam můžeme z dat odstranit při vytváření trénovací množiny, zároveň nabývá těchto hodnot:

- 0 – pokud je předmět pro studenta povinný nebo oborový,
- 1 – pokud je předmět volitelný nebo není povinný v rámci studentova oboru.

Nejdříve musíme správně rozdělit oborové předměty, opět se vracíme k příznaku *typ_predmetu*, ve kterém opravíme všechny hodnoty v datasetu podle toho, jestli je předmět pro některý obor povinný nebo ne. Tyto informace se dají najít v [32] [33].

Vytvoříme slovník překladů mezi předmětem a specializací. A pro každý záznam tak můžeme zjistit, jestli ho můžeme odstranit, abychom se mohli pokusit jej následně doporučit nebo ne.

Nyní máme hotový dataset, pomocí kterého můžeme měřit, jak se různým přístupům k implementaci systému daří doporučovat. Zároveň, až zjistíme, které hyperparametry algoritmu jsou nejlepší, využijeme tento dataset pro ostré doporučování předmětů.

Dále nám v rámci předzpracování dat vznikl dataset, který obsahuje informace o tom, které předměty spadají pod kterou specializaci a jaké jsou jejich identifikátory.

Tento dataset se skládá ze tří sloupců – *specializace*, *nazev_predmetu* a *predmet_id*. Uložíme si jej ve formátu csv pro budoucí zpracování. Poslouží nám při filtraci relevantních předmětů, abychom při doporučení skryli předměty, které si student musí zapsat povinně.

6.4 Příprava trénovacích a testovacích dat

Z celého datasetu interakcí studentů s předměty vybereme pouze ty, kteří už mají ukončené studium. Tito studenti s velkou pravděpodobností už prošli celým bakalářským programem a mají proto mnoho záznamů s volitelnými předměty, podle kterých můžeme formovat doporučení.

Množinu všech záznamů se studenty, kteří už mají ukončené studium, nazveme testovací. Na těchto datech budeme měřit přesnost doporučení a další metriky.

Trénovací množinu vytvoříme odstraněním některých záznamů z testovací množiny. Záznamy, které odstraníme, vybíráme tak, že data nejdříve pro každého studenta seskupíme podle příznaku *droppable* = 1. Poté 30 % studentů, kteří mají více než pět takových záznamů, skryjeme čtyři volitelné předměty, ze kterých dostali lepší známku než F.

Od tohoto přístupu si slibujeme, že pokud systém předmět doporučí, můžeme říci, že doporučuje předměty, ze kterých nejpodobnější studenti měli lepší známky. Zároveň někteří studenti mají v datech zapsané všechny předměty – tím se snažíme simulovat ostrý běh algoritmu.

Kapitola 7
Návrh řešení

Následující kapitola pojednává o vlastní implementaci celého doporučovacího systému a využití metody k-nejbližších sousedů pro účely doporučování. Z datasetu vybereme potřebné příznaky, vytvoříme vhodnou reprezentaci dat, navrhneme algoritmus pro doporučování, provedeme filtraci povinných předmětů a každému studentovi doporučíme předměty seřazené podle relevance.

7.1 Popis problému

Na základě vybraných údajů o studiu se studentům snažíme doporučovat předměty, které by si mohli zapsat do dalších semestrů. Jedná se o problém strojového učení s učitelem, které jsme si představili v části 2.1.2, protože díky předzpracování známe hodnoty, které mají být na výstupu. Díky konečnému počtu možných doporučení jde zároveň o problém klasifikace, který byl vysvětlen ve stejné části.

Pro doporučování použijeme metodu nejbližších sousedů s různými hodnotami hyperparametrů *počet sousedů* a *metrika vzdálenosti*. Pro různé výstupy doporučování změříme hodnoty metrik z kapitoly 3 a na základě jejich hodnot provedeme v kapitole 8 analýzu, po které budeme schopni určit hodnoty hyperparametrů, které vedou k nejlepším výsledkům.

Z pohledu doporučování se jedná o implementaci kolaborativního filtrování vysvětleného v části 1.2.2.1. Uživatele v matici reprezentují studenti, položky odpovídají předmětům a interakce studenta s předmětem je určena známkou, pokud student nějakou získal, a tím interakce zároveň vyjadřuje, že student měl předmět zapsaný.

7.2 Řešení

Postupně si představíme všechny úkony, které bylo třeba provést na cestě od předzpracovaného datasetu k doporučení předmětů.

7.2.1 Výběr příznaků a jejich reprezentace

Z předzpracovaných dat vybereme příznaky *studium_id*, *predmet_id* a *znamka*. Matici pro kolaborativní filtrování reprezentujeme pomocí 2D pole, které má 812 řádků (každý odpovídá jednomu studentovi) a 130 sloupců (každý odpovídá jednomu předmětu).

Pro každý obor si připravíme speciální vektor, který má 130 prvků, kde budeme na každé pozici indikovat, zdali se jedná o povinný předmět (případně předmět specializace). Tyto vektory

využijeme při filtraci doporučených předmětů, abychom studentům doporučovali pouze předměty, které pro ně jsou volitelné.

7.2.2 Implementace vlastních metrik vzdálenosti

Metrika vzdálenosti musí umět spočítat vzdálenost dvou bodů napříč všemi dimenzemi a zároveň splňovat definici 2.2. Požadujeme, aby výpočet metriky byl rychlý, protože ji musíme vypočítat pro každého studenta s každým studentem. K tomu nám poslouží správné využití funkcí z knihovny NumPy.

Tradiční metriky z knihovny *scikit-learn* [12] počítají se všemi hodnotami v matici pro kolaborativní filtrování. Při jejich bližším zkoumání se ukázalo, že do hledání nejbližších sousedů vnáší element toho, že nepočítáme vzdálenost studentů pouze na základě předmětů, které oba měli zapsané, ale do výpočtu nám zasahují i studenti, kteří předmět zapsaný ještě neměli, což je efekt, který bychom chtěli odstranit.

Ve vlastních metrikách tento problém řešíme vytvořením masky předmětů, kde jedničkou označíme předměty, které měli oba studenti a nulou všechny ostatní. Následně počítáme metriky pouze nad předměty, které nám v seznamu zbydou po vymaskování. Takto upravíme manhattanskou, euklidovskou a cosinovou vzdálenost. Všechny společně s metrikami ze *scikit-learn* zahrneme do testování a hledání nejlepších hyperparametrů v kapitole 8.

7.2.3 Doporučování pomocí knn

Pro implementaci metody nejbližších sousedů využijeme knihovnu *scikit-learn*. Model propojíme s daty a určíme k nejbližších sousedů. Výstupem je seznam nejbližších sousedů a hodnoty jejich vzdáleností.

Při výpočtu nejrelevantnějších předmětů bychom museli hledat nejmenší vzdálenosti, které jsou ale větší než nula. Proto je výhodné vzdálenosti převést na podobnosti, abychom při výpočtu hledali největší hodnoty. Toho docílíme tak, že všechny hodnoty vzdáleností dosadíme do funkce převrácené exponenciály:

$$y = \frac{1}{e^x},$$

kde x odpovídá vzdálenosti a y podobnosti.

Dalším krokem je výpočet relevance každého předmětu pro každého studenta. Každému studentovi vytvoříme matici $\mathbb{D}^{n,n}$, kde na diagonálu umístíme hodnoty podobností s n nejbližšími sousedy a matici $\mathbb{A}^{n,p}$[1]. Provedeme maticové násobení $\mathbb{D} \cdot \mathbb{A}$ a následně pro každý sloupec spočítáme průměr. Tím nám pro každého studenta vznikne vektor relevancí, který je indexovaný podle předmětu, o který se jedná.

Poté každému studentovi vyfiltrujeme vektor relevancí předmětů. Nejdříve označíme za irelevantní předměty, které už absolvoval, a následně všechny předměty, které jsou povinné pro jeho specializaci a předměty, které jsou povinné pro všechny studenty.

Nakonec vrátíme seznam předmětů seřazený podle relevance předmětu pro daného studenta.

Pro tento seznam doporučených předmětů v následující části spočítáme metriky pro vyhodnocování doporučovacích systémů.

[1] Číslo p odpovídá celkovému počtu předmětů.

Kapitola 8

Měření a testování

V rámci této popíšeme způsob testování navrženého doporučovacího systému a shrneme výsledky tohoto testování. Výstupem bude srovnání různých voleb hyperparametrů, na základě kterého budeme schopni vybrat hodnoty, které vedou k nejlepším výsledkům při doporučování.

8.1 Způsob testování

V rámci předzpracování jsme data rozdělili na dvě množiny. Množinu testovací, která obsahuje všechny záznamy o všech proběhlých zápisech všech předmětů, a množinu trénovací, kde jsme několik zápisů skryli. Skryté zápisy, které jsme odseparovali v části 6.4, budeme považovat za očekávané hodnoty na výstupu doporučovacího systému.

Spustíme celý proces doporučování na trénovací množině. U studentů, kterým jsme skryli předměty, vypíšeme, jaké předměty jsme očekávali a jaké předměty byly doporučeny. Napříč všemi takovými studenty sečteme počet úspěšných předpovědí. Takové testování je sice člověku velmi dobře čitelné, ale z pohledu měření kvality doporučování zcela nedostatečné.

Proto implementujeme několik funkcí pro výpočet metrik z kapitoly 3 a ty měříme.

Testování probíhá pomocí tabulky 8.1, kde definujeme všechny hyperparametry, které chceme otestovat. Pro každou kombinaci hyperparametrů vždy nad trénovacími daty vytvoříme doporučovací systém. Změříme hodnoty jeho metrik a ty uložíme pro budoucí analýzu výsledků.

Hodnoty počtu nejbližších sousedů jsou vybrány s ohledem na pokrytí celé škály možných výběrů, zároveň – protože máme pouze 812 studentů – nebudeme zkoušet více než 500 sousedů.

Tabulka 8.1 Zkoumané hodnoty hyperparametrů

Počet sousedů	Metrika vzdálenosti
5	Manhattanská
20	Euklidovská
50	Cosinová
100	Manhattanská upravená
200	Euklidovská upravená
500	Cosinová upravená

Tabulka 8.2 Nejlepších 5 modelů podle naměřených hodnot MAE, MSE a RMSE na datech FIT ČVUT

Pořadí	Míra vzdálenosti	Počet sousedů	MAE	MSE	RMSE
1.	Cosinová upravená	200	0,482	0,318	0,540
2.	Cosinová upravená	500	0,485	0,319	0,541
3.	Cosinová upravená	100	0,486	0,322	0,544
4.	Cosinová upravená	50	0,496	0,335	0,556
5.	Cosinová	50	0,504	0,344	0,563

Tabulka 8.3 Nejlepších 5 modelů podle naměřených hodnot accuracy, precision a recall na datech FIT ČVUT

Pořadí	Míra vzdálenosti	Počet sousedů	Accuracy	Precision	Recall
1.	Cosinová	20	0,914	0,269	0,538
2.	Euklidovská	20	0,913	0,267	0,533
3.	Cosinová	50	0,913	0,265	0,530
4.	Euklidovská	50	0,913	0,264	0,529
5.	Manhattanská	200	0,912	0,261	0,523

8.2 Výsledky měření

Po vzoru vyhodnocování výsledků implementace doporučovacího systému předmětů pro střední školy v Nizozemsku (3.2.2), vyhodnotíme systémy postupně podle každé měřené metriky a okomentujeme, jaký závěr můžeme pro nastavení hyperparametrů z metrik vyvodit.

Hodnoty metrik měříme pouze na prvních osmi doporučených předmětech a snažíme se hledat nejlepší nastavení hyperparametrů. Jako zdroj pravdy bereme čtyři předměty, které jsme studentům skryli. Pro metriku NDCG jsme určili pořadí předpokládaných předmětů podle známky, kterou z předmětu student v testovacích datech má.

8.2.1 Statistické měření chybovosti

Tyto metriky v naší aplikaci počítají rozdíl mezi známkou, kterou student dostal[1], a systémem vypočtenou relevancí předmětu pro studenta. Tyto dvě veličiny spolu nemají téměř nic společného, proto výpočty těchto metrik jsou zavádějící.

Abychom je mohli využít, museli bychom buď doporučovacím systémem předpovědět známku, kterou by student dostal v případě, že by si předmět zapsal, nebo známky, které jsou součástí testovací množiny nějakým způsobem převést na relevanci předmětu pro studenta.

Oba způsoby byly vyhodnoceny jako velké množství práce navíc téměř bez žádného efektu. Protože se dají vypočítat, můžeme se podívat na výsledky v tabulce 8.2. Pro určení nejlepšího nastavení na ně ale nebereme ohled.

Kdybychom slepě následovali tyto metriky, mohli bychom říci, že námi upravená cosinová vzdálenost je nejlepší metrikou pro algoritmus k-nejbližších sousedů.

8.2.2 Měření z hlediska kvality doporučování

V tabulkách 8.3 a 8.4 vidíme metriky, jejichž využití nemá zavádějící výsledky, protože už nepracují s hodnotami interakcí studenta s předmětem, ale s interakcí samotnou.

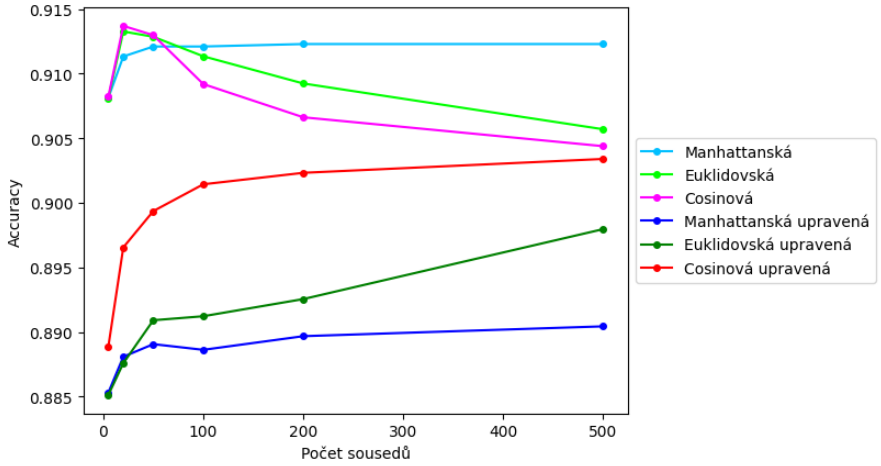
V tabulce 8.3 je přesně vidět problém, ke kterému tato metrika často inklinuje. Nabývá obrovských hodnot (až přes 90 %), a doporučovací systém tak vypadá jako velmi přesný software.

[1] A my jsme mu jí v trénovacích datech skryli.

Tabulka 8.4 Nejlepších 5 modelů podle naměřených hodnot NDCG@8 na datech FIT ČVUT

Pořadí	Míra vzdálenosti	Počet sousedů	NDCG@8
1.	Euklidovská	20	0,524
2.	Cosinová	20	0,519
3.	Euklidovská	50	0,516
4.	Cosinová	50	0,512
5.	Euklidovská	100	0,505

Problém nastává, protože jsme v rámci doporučovaných předmětů vybrali pouze osm nejlepších a zbytek nedoporučujeme. Předměty, které předpokládáme, že doporučíme, jsou pouze čtyři, a tak je množina správně nedoporučených předmětů velká a hodnota metriky je touto v mnoha případech správnou klasifikací ovlivněna.



Obrázek 8.1 Graf závislosti accuracy na počtu sousedů pro jednotlivé metriky vzdálenosti

Následující dva grafy, 8.2 a 8.3, vypadají na pohled stejně jako graf zavádějící metriky accuracy 8.1, liší se ale škálou, na které se nachází hodnoty těchto metrik.

Mezi *accuracy* a *precision/recall* je rozdíl vidět u pěti sousedů a manhattanské vzdálenosti.

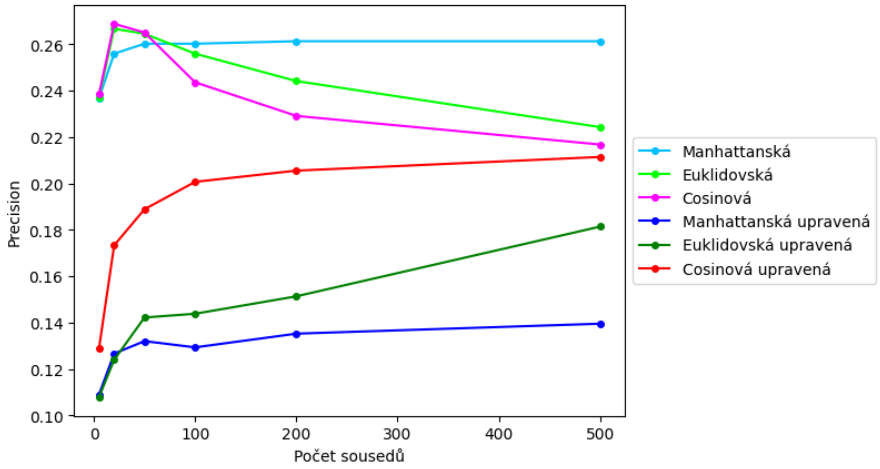
Grafy 8.2 a 8.3 pro vypadají stejně *precision* a *recall* vypadají stejně až na rozdíl ve škále, kde jsou čísla u *recall* přesně polovinou čísel u *precision*. To se děje, protože když se vrátíme zpět ke vzorcům, podle kterých se tyto metriky představené v 3.1.2 počítají, všimneme si, že čitatel je u obou stejný. Ve jmenovateli pak dosazujeme pro *precision* počet očekávaných doporučení a pro *recall* počet námi provedených doporučení.

Očekáváme 4 doporučení a predikujeme 8 doporučení. Rázem vidíme, proč v našem případě platí:

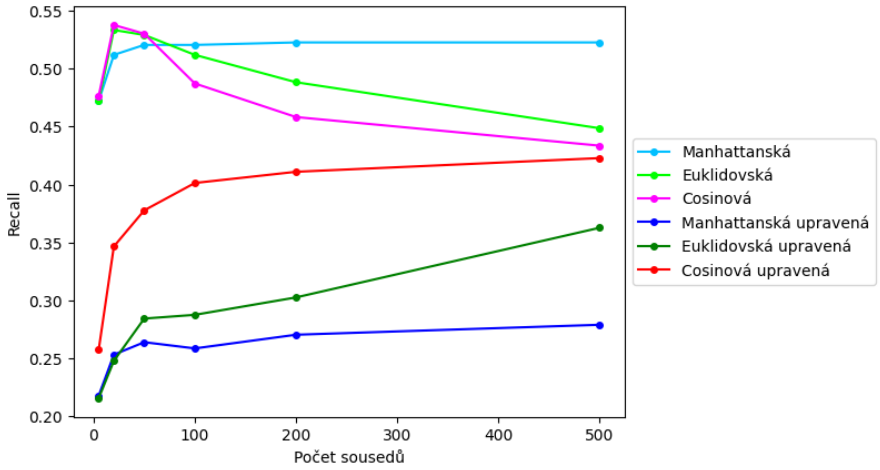
$$precision = \frac{recall}{2}$$

U metriky *NDCG@8* jsme očekávali změnu trendů, které mají ostatní metriky měření kvality doporučení. V trendech jsou změny sice minimální, ale můžeme si povšimnout, že se nám změnilo pořadí nejlepších hyperparametrů, kde nyní vede euklidovská vzdálenost na 20 sousedech.

Znamená to, že v přesnosti doporučené množiny předmětů je lepší cosinová podobnost s 20 sousedy, zatímco pokud bych chtěli doporučovat předměty v pořadí daném výslednou známkou, je lepší zvolit euklidovskou metriku vzdálenosti.



Obrázek 8.2 Graf závislosti precision na počtu sousedů pro jednotlivé metriky vzdálenosti



Obrázek 8.3 Graf závislosti recall na počtu sousedů pro jednotlivé metriky vzdálenosti

Co se týče změn v grafu na obrázku 8.4, stojí za zmínění, že s rostoucím počtem sousedů byla euklidovská metrika horší než manhattanská až na hranici 200 sousedů, zatímco u předešlých metrik tato tato výměna pozic nastala už na 100 sousedech.

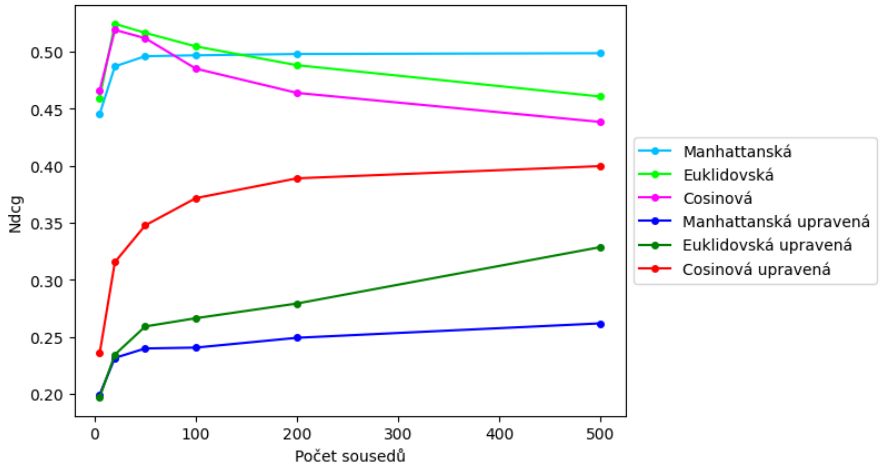
Zároveň při zvýšení počtu sousedů není u cosinové podobnosti tak rapidní zhoršení při změně z 50 na 100 nejbližších sousedů.

8.2.3 Míra pokrytí

Naměřené hodnoty míry pokrytí předmětů jsou vidět v tabulce 8.5 a na grafu na obrázku 8.5. Míra pokrytí nám udává poměr předmětů, které byly doporučeny, oproti předmětům, které mohly být doporučeny.

Pokud chceme, aby systém doporučoval co nejširší škálu volitelných předmětů, měli bychom podle tabulky 8.5 využít 5 nejbližších sousedů a cosinovou podobnost.

Z grafu na obrázku 8.5 lze vyčíst, že pro většinu metrik platí, že pokud chceme doporučit co nejvíce různých předmětů, musíme volit nižší počet sousedů. To je očekávané chování, protože podle bias-variance tradeoff z části 2.1.4 vede nižší počet sousedů k vyššímu rozptylu.



Obrázek 8.4 Graf závislosti NDCG@8 na počtu sousedů pro jednotlivé metriky vzdálenosti

Tabulka 8.5 Nejlepších 5 modelů podle naměřených hodnot míry pokrytí na datech FIT ČVUT

Pořadí	Míra vzdálenosti	Počet sousedů	Míra pokrytí
1.	Cosinová	5	0,895
2.	Manhattanská	5	0,886
3.	Euklidovská	5	0,886
4.	Manhattanská	20	0,867
5.	Manhattanská	50	0,848

Zároveň se můžeme domnívat, že s vyšším počtem sousedů se objeví volitelné předměty, které mělo zapsáno více lidí než jiné. Díky tomu tyto častěji zapisované předměty doporučujeme, protože na méně zapisované předměty není dostatek studentů, kteří by systém přesvědčili o tom, že ho má doporučit.

Ukázalo se také, že manhattanská metrika je jako jediná s rostoucím počtem sousedů odolná proti malému pokrytí dat.

8.2.4 Výběr nejlepší metriky

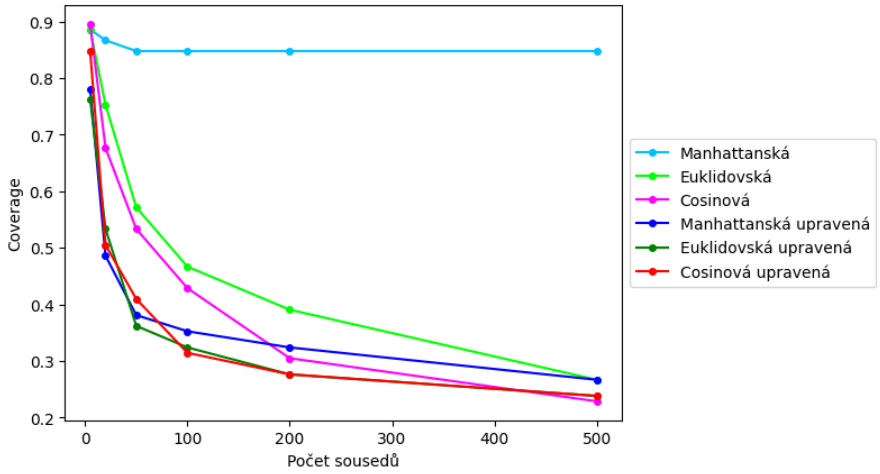
Na závěr zkusíme zprůměrovat všechny metriky, které se týkají kvality doporučování[2], a uvidíme, které hodnoty hyperparametrů jsou globálně nejlepší pro co nejpřesnější doporučování.

V tabulce 8.6 je vidět, že nejlepší volbou pro implementaci doporučovacího systému je cosinová

[2] Jde o accuracy, precision, recall a NDCG@8.

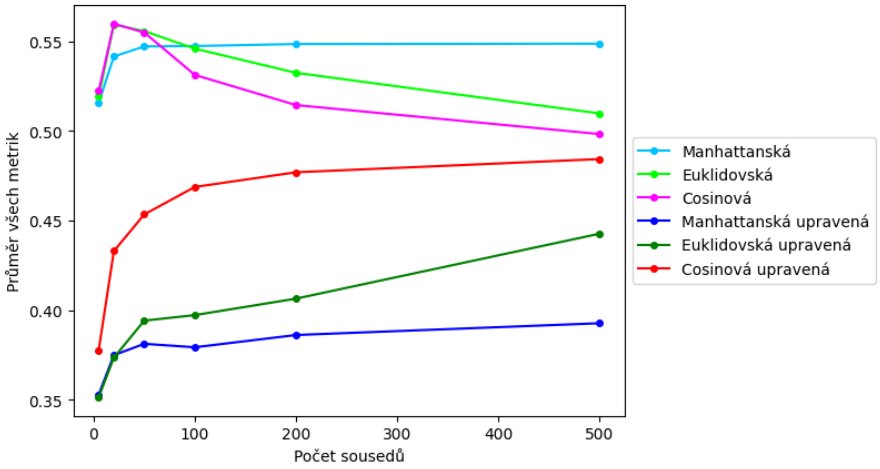
Tabulka 8.6 Nejlepších 5 modelů podle průměru naměřených hodnot na datech FIT ČVUT

Pořadí	Míra vzdálenosti	Počet sousedů	Průměrné hodnocení
1.	Cosinová	20	0,560
2.	Euklidovská	20	0,559
3.	Euklidovská	50	0,556
4.	Cosinová	50	0,555
5.	Manhattanská	500	0,549



Obrázek 8.5 Graf závislosti míry pokrytí na počtu sousedů pro jednotlivé metriky vzdálenosti

podobnost s 20 sousedy. Euklidovská vzdálenost se na doméně doporučování předmětů na FIT ČVUT ukázala také jako velmi dobrá volba.



Obrázek 8.6 Graf závislosti průměru všech evaluačních metrik na počtu sousedů pro jednotlivé metriky vzdálenosti

Z grafu na obrázku 8.6 můžeme vidět, že chováním si jsou euklidovská i cosinová metrika vzdálenosti velmi podobné.

Dále je z grafu poznat, že veškerá snaha o implementaci vlastní metriky nepřináší dobré výsledky. Ze všech námi implementovaných verzí si nejlépe ve všech případech vedla cosinová podobnost. Její rostoucí úspěšnost v závislosti na počtu sousedů nejspíš spočívá v tom, že doporučuje předměty, které jsou obecně oblíbené, a nezkouší tak doporučovat předměty, které mají zapsané pouze velmi specifičtí studenti, a tím pádem dosahuje lepších výsledků.

Protože nás ale nezajímá vývoj metrik v závislosti na počtu sousedů, ale pouze jedno jediné nejlepší nastavení, stává se nejlepší kombinací hyperparametrů cosinová podobnost s 20 sousedy.

Pokud bychom někdy v budoucnu chtěli brát v potaz i pořadí doporučených předmětů – což v této práci není cílem – měli bychom se zamyslet nad zvolení euklidovské vzdálenosti s 20 sousedy, které nám radí metrika NDCG@8.

Pokud se chceme věnovat doporučování co největší škále různých předmětů, nejlepšími hyperparametry pak je cosinová podobnost s 5 sousedy. U této metriky je ale otázka, zdali je přesná. V testech se neumístila dobře, ale protože píšeme systém, který má studentům pouze radit a ne za ně rozhodovat, je i tato volba na zvážení.

8.3 Porovnání výsledků s výsledky z Nizozemské střední školy

Následující sekce porovná všechny metriky nalezené v práci [2] s metrikami změřenými pro náš doporučovací systém. Porovnat je je vhodné, protože obě práce implementovaly stejný algoritmus. Protože je algoritmus velmi závislý na dostupných datech, která byla pro oba systémy úplně jiná, zmíníme porovnání alespoň pro účely rámcové představy.

V našem případě šlo vždy o kolaborativním filtrování na základě uživatele, zatímco v Nizozemsku zkoušeli přístup na základě předmětů.

8.3.1 MSE

V naší práci dříve zaznělo, že tato metrika nemá pro náš doporučovací systém žádnou vypovídací hodnotu. Otázka je, jak byl implementován doporučovací systém z rešerše. Je možné že také trpěl tím, že počítal metriku na dvou neporovnatelných charakterech hodnot.

Náš doporučovací systém má při nejlepším nastavení hyperparametrů hodnotu 0,31, zatímco nizozemský algoritmus má 0,36. Protože jde o metriku, která měří chybovost, nižší hodnota označuje lepší hodnocení.

8.3.2 Precision a recall

V případě těchto metrik už ale nemůžeme slavit takové úspěchy. Opět je to nejspíš dané charakterem testování. Princip byl stejný, konkrétní hodnoty, kolik předpokládají doporučených předmětů a kolik jich doporučují už nebyly zmíněny. Počet doporučených předmětů v této metrice hraje obrovskou roli.

Pro metriku *precision* porovnáváme našich 0,269 s nizozemskými 0,52, tudíž bychom mohli říci, že náš systém je o mnoho horší. Srovnání podle těchto metrik je ale u algoritmu k-nejbližších sousedů na rozdílných datech nemožné.

U *recallu* jsme se ale se svojí cosinovou podobností na 20 sousedech a hodnotou 0,538 přiblížili referenčním 0,60, což vypadá dobře.

8.3.3 Míra pokrytí

Poslední metrikou, ve které budeme řešit porovnání, je míra pokrytí. Ta v našem případě vystoupala na 0,895, zatímco v Nizozemsku na 0,95.

Kapitola 9
Diskuze a budoucí práce

Nyní prodiskutujeme, jakým jsme v rámci práce čelili problémům, jak jsme se rozhodli je řešit a proč. Dále se budeme zabývat porovnáním naší práce s pracemi předchozími. Nakonec nastíníme různé nápady, které jsme nedokázali prací pokrýt a do budoucna stojí za to se jimi zabývat.

9.1 Problémy, kterým jsme čelili

Problémů a učiněných rozhodnutí bylo mnoho, v následujících řádcích zmíníme ty nejpalčivější, na které je třeba si v budoucnu dát pozor, pokud by práce měla pokračování.

9.1.1 Data z Ankety ČVUT

Původní návrh práce počítal s doporučováním předmětů, které by se studentům měly líbit. Na to jsme chtěli využít data z Ankety ČVUT. U těch bylo potřeba, abychom znali jednotlivá hodnocení studentů pro hledání podobnosti mezi nimi tak, jako to v práci nyní provedeno na známkách z předmětů.

Anketa ČVUT je ale schválně anonymizovaná tak, aby nikdo nemohl zjistit, který student co napsal. Není to tak, jak jsme se původně domnívali, že by tyto informace byly pouze skryté.

9.1.2 Vytváření datasetu

S předěláním akreditace se vyskytlo v datech mnoho problémů, se kterými jsme se museli potýkat. Naštěstí Datový sklad ČVUT nám pro práci dodal data, ve kterých se často vyskytovaly vzorce, podle který předzpracování šlo rychle od ruky. Pro vytvoření datasetu, který obsahoval všechno, co bylo třeba, a zároveň neobsahoval nic nežádoucího, bylo sice třeba využít mnoho různých technik, ale data byla strojově dobře čitelná a zpracovatelná.

Kdybychom se chtěli pokusit o doporučování volitelných předmětů studentům magisterského programu, bylo by dobré využít informace o předešlém studiu. To bohužel data neobsahují, protože každý student při zápisu získává identifikátor studia, který se při každém dalším zápisu mění. Tento problém bychom mohli zkusit nějak vyřešit pomocí spojování podle jména, ale pak narazíme na problémy s duplicitami. Realizace tohoto nápadu je tedy velmi problematická.

Na cestě nás překvapilo, že se podle příznaku *zpusob_ukonceni* nedají filtrovat studenti, kteří úspěšně dokončili. Pokud by v datech byl příznak označující tuto skutečnost, měli bychom lépe rozdělenou množinu dat na testovací a trénovací. Nyní jsme museli brát v potaz pouze studenty, kteří ukončili studium a zároveň během studia měli zapsaný vypovídající počet předmětů.

9.1.3 Nová akreditace

Další obrovskou výzvou bylo vymyslet, jak naložit s různými identifikátory předmětů pro ty, které zůstaly stejné, ale pouze změnily kód. Nakonec jsme využili stejného názvu, kdy jsme předměty seskupili a nahradili různé identifikátory jednotnými identifikátory, kdy poté pro každý předmět už existoval jeden nový unikátní identifikátor.

V rámci změny akreditace jsme také narazili na problém se specializacemi, které se buď změnily, nebo vznikly nové. U změněných specializacích je dále problém v tom, že se změnila i skladba pro ně povinných předmětů, tudíž se z některých původně předmětů specializace staly předměty volitelné a obráceně. S tímto jsme se vypořádali tak, že jsme každé specializaci podle toho, jaké máme dnes, označili za předměty specializace všechny, které v rámci ní dříve byly nebo nyní jsou povinné. Pokud bychom to neudělali a systém by doporučoval podle studentů ze staré akreditace, doporučoval by na prvních příčkách dříve povinné předměty, protože je studenti museli mít zapsané.

Doporučovat na základě studentů nové akreditace je pomocí systému možné, ale není změřeno, jak přesné doporučování studentům poskytuje, protože akreditace ještě neexistuje dostatečně dlouho na to, aby obsahovala dostatek studentů, kteří by sloužili jako referenční data.

9.1.4 Vytváření vlastní metriky

Po velmi dlouhém rozmýšlení, jaké atributy by měla metrika splňovat a proč by měla být vhodnější pro řešený problém, jsme odhalili, že klasické metriky počítají podobnost na základě všech předmětů, které studenti mají k dispozici.

Původní myšlenka za vlastní metrikou byla v podobnosti rapidně zvýhodňovat studenty, kteří mají přesně stejné známky (nebo jen o stupeň jiné) a pokud by se známka studentů lišila o dva stupně, tak se k nim chovat, jako že si nejsou vůbec podobní na základě tohoto předmětu.

Poté jsme si ale uvědomili, že filozofie za naší prací je, že na základě předmětů, které studenti absolvovali, vybereme ty nejpodobnější a poté určíme relevanci každého předmětu. Že už úkon zapsání a absolvování předmětu nám dává dostatečné množství informace, které potřebujeme brát v potaz. Proto jsme upravili metriky, které se používají pro obecné řešení problémů tak, aby braly v potaz pouze studenty, kteří měli skutečně zapsané stejné předměty a nad nimi jsem tyto obecné metriky vzdálenosti počítali.

Ukázalo se, že naše myšlenka není správná, a systém, který počítá vzdálenosti na předmětech, které oba studenti měli, neměli a nebo předmět měl pouze jeden z nich, dosahuje lepších výsledků než doporučování na základě předmětů, které měli oba studenti.

9.1.5 Nastavení hodnot při testování

Při modelování ve strojovém učení se obvykle určuje velikost testovací množiny kolem 20–30 %, protože my jsme následně neodstraňovali studentům všechny volitelné předměty, abychom tak drasticky nesnížili podíl informací o studentovi v trénovacích datech, zvolili jsme horní hranici – tedy 30 %.

Čtyři předpokládané předměty vychází z praxe, protože studenti si mají podle doporučeného průchodu zapsat kolem 30 volitelných kreditů při průměrném kreditovém ohodnocení volitelného předmětu čtyřmi kredity to vychází na necelých osm volitelných předmětů, ze kterých jsme vzali polovinu.

Doporučování osmi nejrelevantnějších předmětům se nám k požadovaných čtyřem zdálo jako odpovídající hodnota, ve které systém má možnost při doporučování dělat chyby, ale zároveň jich nemůže udělat mnoho.

Škála nejbližších sousedů je pak volena tak, aby pokryla jak nejmenší, tak největší hodnoty počtu sousedů, ale abychom zároveň neřešili problém zbytečně mockrát bez přínosu nějaké informace. Opět šlo spíše o autorovy dosavadní zkušenosti se strojovým učením.

Závěr

Hlavním cílem práce bylo vytvořit doporučovací systém pro studenty Fakulty informačních technologií ČVUT v Praze. Systém měl být schopný identifikovat podobné studenty podle studijních výsledků a na základě rozdílu množin volitelných předmětů studentů určovat, jaké předměty by se jim mohly líbit. Dalšími cíli získat, zanalyzovat a vhodně předzpracovat data, vytvořit vlastní metriku podobnosti dvou studentů pro doporučování a nakonec testování různých nastavení algoritmu pro co nejlepší výsledky doporučování. Nakonec vybrat a implementovat systém s pomocí nejlepších hyperparametrů.

První omezení, se kterým jsme se setkali, bylo, že z dat nestačilo pouze extrahovat volitelné předměty a na nich doporučovat, ale museli jsme vzít v potaz studentovu specializaci, abychom mohli doporučovat i předměty, které sice mohou pro jednoho studenta povinné, ale pro jiného volitelné.

V teoretické části jsme se nejdříve seznámili s problematikou doporučování obecně. Vysvětlili jsme si různé přístupy ke zpracování dat v kontextu doporučování. Následně jsme se seznámili se strojovým učení a dalšími důležitými pojmy, které se kolem tohoto směru informatiky objevují.

Zadání nám ukládalo využívat vlastní metriky pro určení podobnosti studentů. Jedním ze základních algoritmů, který hledá podobnost mezi datovými body, je metoda k-nejbližších sousedů. Pro jeho plné pochopení jsme se seznámili s termínem *vzdálenost* a ukázali jsme si některé často využívané.

Poslední kapitolu teoretické části jsme věnovali zkoumání různých metrik porovnávání modelů strojového učení z několika různých hledisek. Vedle klasických metrik pro doporučování *precision* a *recall* jsme se seznámili s velmi zajímavou metrikou *Normalized discounted cumulative gain*, která vyhodnocuje nejen správné doporučení, ale i jeho pořadí. Dále jsme se zabývali statistickým vyhodnoceváním chyby nebo mírou pokrytí dat.

Data z portálu KOS ČVUT, která bylo pro práci potřeba, nám poskytl Datový sklad ČVUT. Dále od Ing. Michala Valenty, Ph.D. byla k dispozici data z Ankety ČVUT. Sice jsme je při doporučování nijak nezohlednili, ale v datasetu, který je také výstupem této práce jsou u každého předmětu uložená pro budoucí použití.

Po získání dat jsme provedli základní seznámení s nimi, zjistli jsme, jakou informaci nese který příznak, případně jakých tyto příznaky nabývají hodnot.

Dalším krokem bylo předzpracování, které se skládalo z několika fází. Nejdříve jsme se věnovali studentům, kterým jsme na základě některých ostatních příznaků určili specializaci, kterou si student zapsal, abychom mohli posléze odfiltrovat oborové předměty. Dále jsme studenty rozdělili podle toho, jestli ještě studují nebo nikoliv.

Největší výzvou v předzpracování byl příchod nové akreditace. S ní se totiž některé předměty změnily, některé se zrušily a jiné zase přibyly. Ve výsledku se nám podařilo odfiltrovat mnoho zbytečných předmětů a předměty, které zůstaly stejné, spojit do jednoho datového záznamu.

Následně jsme všechna data spojili dohromady a provedli další kroky předzpracování, které

nám toto spojení umožnilo. Nejdůležitějším z těchto kroků bylo zakódování známek do intervalu $(0, 1)$, protože s hodnotami, které jsme zde dosadili velmi úzce souvisí způsob, kterým se doporučovací systém rozhoduje. Další velmi důležitý krok v této fázi bylo označení záznamů, jestli je chceme v oboru hodnot doporučovacího systému. To bylo možné až teď, protože dříve jsme znali pouze identifikátory studentů a předmětů, a neměli jsme informaci o tom, jaký student si zapsal jaký předmět.

Protože využíváme metodu k-nejbližších sousedů, která je založená na kvalitním předzpracování dat, byla implementace jádra doporučovacího systému daleko jednodušší než by člověk čekal. Zde je zároveň to místo, kde je třeba implementovat vlastní metriky podobnosti.

Od vlastní metriky očekáváme, že bude porovnávat pouze studenty, kteří už oba měli zapsané předměty, podle kterých se je snažíme srovnat. Ukázalo se totiž, že klasické vzdálenosti do součtu přidávají i studenty, kteří předměty neměli zapsané.

V rámci doporučování nejdříve podle nějaké metriky vzdálenosti nalezneme nejbližší sousedy, poté převedeme vzdálenost na podobnost pomocí převrácené hodnoty exponenciální funkce a spočítáme relevanci každého předmětu pro každého studenta na základě nejpodobnějších studentů.

Takové výstupy ovšem bylo třeba profiltrovat a odstranit předměty, které student už měl zapsané, nebo které jsou jeho oborovou záležitostí. Nakonec byly předměty seřazeny podle relevance a prvních k jich bylo doporučeno.

V poslední části práce měříme úspěšnost doporučování pro různé hodnoty nastavení hyperparametrů metody k-nejbližších sousedů. Až tady zjišťujeme, že vymyslet a využívat vlastní metriky není tak snadné, jak se zdá. Ani jedna ze tří implementovaných totiž nebyla v žádném nastavení lepší než jakákoliv metrika, které nebrala ohledy na zápisy předmětů studenty.

Nápadů, jak navázat na práci či ji alespoň rozšířit je mnoho. Nyní je systém implementovaný tak, že počítá pouze se studenty českého prezenčního bakalářského oboru. Rozšířit jej na magisterská data by nemělo být s dostupnými prostředky nemožné. Obáváme se akorát, toho, že v magisterském programu není tolik příležitostí zapisovat si volitelné předměty. Zároveň předměty z jiných oborů už často vyžadují znalosti, kterými nedisponuje každý student. A nakonec mnoho studentů si jako volitelné předměty zapisuje předměty z bakalářského programu.

V práci je možné provést další různé experimenty jako například změna reprezentace známe z lineární škály na škálu, která bude kopírovat počty bodů pro zisk známky. Předměty, ze kterých se nedá získat známka byly úspěchu dokončení rozděleny na známky A a F. Další experiment se dá provést tak, že rozdělíme studenty jinak než na ty, kteří mají ukončené studium, a ty, kteří ne.

Co by se ale určitě hodilo je reálné vyzkoušení v ostrém provozu, které bylo provedeno v diplomové práci Ondřeje Nového. [1] Systém by se mohl nějakým způsobem dostat ke studentům v době předběžných zápisů s krátkým formulářem s pár otázkami na spokojenost studentů s doporučením, na základě kterého bychom se mohli dozvědět, jestli je systém dobrý nebo ne.

Poslední nápad je vyzkoušet pro doporučování nějaký jiný algoritmus. Po konzultaci s Ing. Vojtěchem Vančurou ohledně doporučovacích systémů přišel nápad vyzkoušet spustit doporučování pomocí autoencoderů.

Bibliografie

1. NOVÝ, ONDŘEJ. *Doporučovací systém pro výběr volitelných předmětů* [online]. 2017. [cit. 2023-05-03]. Dostupné z: http://hdl.handle.net/10467/70149.
2. DE HEUS, MARIJE. *Design and evaluation of a recommender system for high school courses in the Netherlands* [online]. 2013. [cit. 2023-05-03]. Dostupné z: http://essay.utwente.nl/65029/.
3. KAPITÁNOVÁ, ANNA. *Predikce výkonů studentů* [online]. 2022. [cit. 2023-05-10]. Dostupné z: http://hdl.handle.net/10467/102031.
4. SKOPAL, TOMÁŠ. *Vzhledávání na webu a v multimediálních databázích: Personalizované vyhledávání a sociální kontext* [online]. České vysoké učení technické v Praze, Fakulta informačních technologií, 2020. [cit. 2023-03-02]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/493967/course/section/76282/BIVWM_lecture07.pdf.
5. VED. *RECOMMENDER SYSTEMS 101* [online]. d4datascience, 2016 [cit. 2023-03-02]. Dostupné z: https://d4datascience.com/2016/07/22/recommender-systems-101/.
6. RECOSENSE. *Content Recommendations on Media Websites* [online]. 2020. [cit. 2023-03-02]. Dostupné z: https://recosenselabs.com/blog/content-recommendations-media-websites.
7. JIANG, TAMMY; GRADUS, JAIMIE L.; ROSELLINI, ANTHONY J. Supervised Machine Learning: A Brief Primer. *Behavior Therapy* [online]. 2020, roč. 51, č. 5, s. 675–687 [cit. 2023-04-27]. ISSN 0005-7894. Dostupné z DOI: https://doi.org/10.1016/j.beth.2020.05.002.
8. ROELOFS, REBECCA; SHANKAR, VAISHAAL; RECHT, BENJAMIN; FRIDOVICH-KEIL, SARA; HARDT, MORITZ; A DALŠÍ. A Meta-Analysis of Overfitting in Machine Learning. *Advances in Neural Information Processing Systems* [online]. 2019, roč. 32 [cit. 2023-04-27]. Dostupné z: https://proceedings.neurips.cc/paper_files/paper/2019/file/ee39e503b6bedf0c98c388b7e8589aca-Paper.pdf.
9. FORTMANN-ROE, SCOTT. *Understanding the Bias-Variance Tradeoff* [online]. 2012. [cit. 2023-04-29]. Dostupné z: http://scott.fortmann-roe.com/docs/BiasVariance.html.
10. WEINBERGER, KILIAN. *Lecture 12: Bias-Variance Tradeoff* [online]. 2018. [cit. 2023-04-29]. Dostupné z: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html.
11. KLOUDA, KAREL; LOPEZ, JUAN PABLO MALDONADO; VAŠATA, DANIEL. *Vytěžování znalostí z dat: Metoda nejbližších sousedů, křížová validace* [online]. České vysoké učení technické v Praze, Fakulta informačních technologií, 2022. [cit. 2023-04-27]. Dostupné z: https://courses.fit.cvut.cz/BI-VZD/lectures/files/BI-VZD-04-cs-handout.pdf.

12. PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; A DALŠÍ. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* [online]. 2011, roč. 12, s. 2825–2830 [cit. 2023-04-30]. Dostupné z: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

13. MARIA, E.; BUDIMAN, E.; HAVILUDDIN; TARUK, M. Measure distance locating nearest public facilities using Haversine and Euclidean Methods. *Journal of Physics: Conference Series* [online]. 2020, roč. 1450, č. 1, s. 012080 [cit. 2023-05-01]. Dostupné z DOI: 10.1088/1742-6596/1450/1/012080.

14. MARTINEZ, JOSE. Game AI techniques applied to city simulations. In: [online]. 2020 [cit. 2023-05-01]. Dostupné z: https://www.researchgate.net/figure/Euclidean-and-Manhattan-distance-comparison-3235-Optimizations-The-first-optimization_fig24_343237167.

15. SKOPAL, TOMÁŠ. *Vzhledávání na webu a v multimediálních databázích: Vektorový model v information retrieval* [online]. České vysoké učení technické v Praze, Fakulta informačních technologií, 2020. [cit. 2023-05-01]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/602570/course/section/92614/BIVWM_lecture03.pdf.

16. KNOP, DUŠAN; MALÍK, JOSEF; SUCHÝ, ONDŘEJ; TVRDÍK, PAVEL; VALLA, TOMÁŠ. *Algoritmy a grafy 1: Dynamické programování* [online]. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023. [cit. 2023-05-01]. Dostupné z: https://courses.fit.cvut.cz/BI-AG1/lectures/media/bi-ag1-p10-handout.pdf.

17. PARSANIYA, VATSAL. *K-Nearest neighbor clustering* [online]. 2020. [cit. 2023-04-29]. Dostupné z: https://vatsalparsaniya.github.io/ML_Knowledge/Nearest_neighbours/Readme.html.

18. ANCHALIA, PRAJESH P.; ROY, KAUSHIK. The k-Nearest Neighbor Algorithm Using MapReduce Paradigm. In: *2014 5th International Conference on Intelligent Systems, Modelling and Simulation* [online]. 2014, s. 513–518 [cit. 2023-04-29]. Dostupné z DOI: 10.1109/ISMS.2014.94.

19. *K-Nearest Neighbors Algorithm* [online]. 2022. [cit. 2023-04-30]. Dostupné z: https://www.ibm.com/topics/knn.

20. PAPAGELIS, MANOS; PLEXOUSAKIS, DIMITRIS. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. *Engineering Applications of Artificial Intelligence* [online]. 2005, roč. 18, č. 7, s. 781–789 [cit. 2023-05-04]. ISSN 0952-1976. Dostupné z DOI: https://doi.org/10.1016/j.engappai.2005.06.010.

21. WEIJIE, WANG; YANMIN, LU. Analysis of the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) in Assessing Rounding Model. *IOP Conference Series: Materials Science and Engineering* [online]. 2018, roč. 324, č. 1, s. 012049 [cit. 2023-05-04]. Dostupné z DOI: 10.1088/1757-899X/324/1/012049.

22. QIAN, YAWEI; ZENG, GUANG; PAN, YUE; LIU, YANG; LI, KUN. A Prediction Model for High Risk of Positive RT-PCR Test Results in COVID-19 Patients Discharged From Wuhan Leishenshan Hospital, China. *Frontiers in Public Health* [online]. 2021, roč. 9, s. 778539 [cit. 2023-05-04]. Dostupné z DOI: 10.3389/fpubh.2021.778539.

23. MA, JUN; DING, YUEXIONG; CHENG, JACK; TAN, YI; GAN, VINCENT; ZHANG, JINGCHENG. Analyzing the Leading Causes of Traffic Fatalities Using XGBoost and Grid-Based Analysis: A City Management Perspective. *IEEE Access* [online]. 2019, s. 1 [cit. 2023-05-04]. Dostupné z DOI: 10.1109/ACCESS.2019.2946401.

24. JÄRVELIN, KALERVO; KEKÄLÄINEN, JAANA. IR Evaluation Methods for Retrieving Highly Relevant Documents. *SIGIR Forum* [online]. 2017, roč. 51, č. 2, s. 243–250 [cit. 2023-05-04]. ISSN 0163-5840. Dostupné z DOI: 10.1145/3130348.3130374.

25. DHINAKARAN, APARNA. *Demystifying NDCG* [online]. 2023. [cit. 2023-05-04]. Dostupné z: https://towardsdatascience.com/demystifying-ndcg-bee3be58cfe0.

26. VALIZADEGAN, HAMED; JIN, RONG; ZHANG, RUOFEI; MAO, JIANCHANG. Learning to Rank by Optimizing NDCG Measure. In: BENGIO, Y.; SCHUURMANS, D.; LAFFERTY, J.; WILLIAMS, C.; CULOTTA, A. (ed.). *Advances in Neural Information Processing Systems* [online]. Curran Associates, Inc., 2009, sv. 22 [cit. 2023-05-04]. Dostupné z: https://proceedings.neurips.cc/paper_files/paper/2009/file/b3967a0e938dc2a6340e258630febd5a-Paper.pdf.

27. GE, MOUZHI; DELGADO-BATTENFELD, CARLA; JANNACH, DIETMAR. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In: *Proceedings of the Fourth ACM Conference on Recommender Systems* [online]. Barcelona, Spain: Association for Computing Machinery, 2010, s. 257–260 [cit. 2023-05-04]. RecSys '10. ISBN 9781605589060. Dostupné z DOI: 10.1145/1864708.1864761.

28. HARRIS, CHARLES R.; MILLMAN, K. JARROD; VAN DER WALT, STÉFAN J.; GOMMERS, RALF; VIRTANEN, PAULI; COURNAPEAU, DAVID; A DALŠÍ. Array programming with NumPy. *Nature* [online]. 2020, roč. 585, č. 7825, s. 357–362 [cit. 2023-05-08]. Dostupné z DOI: 10.1038/s41586-020-2649-2.

29. VERMA, SHIVA. *How Fast Numpy Really is and Why?* [Online]. 2019. [cit. 2023-05-11]. Dostupné z: https://towardsdatascience.com/how-fast-numpy-really-is-e9111df44347.

30. THE PANDAS DEVELOPMENT TEAM. *pandas-dev/pandas: Pandas* [online]. Zenodo, 2020. Ver. 1.5.3 [cit. 2023-05-08]. Dostupné z DOI: 10.5281/zenodo.3509134.

31. ŠAFAŘÍK, JAN; VANČURA, VOJTĚCH; KORDÍK, PAVEL. RepSys: Framework for Interactive Evaluation of Recommender Systems. In: *Proceedings of the 16th ACM Conference on Recommender Systems* [online]. Seattle, WA, USA: Association for Computing Machinery, 2022, s. 636–639 [cit. 2023-05-08]. RecSys '22. ISBN 9781450392785. Dostupné z DOI: 10.1145/3523227.3551469.

32. NOVÁK, J.; HALAŠKA, I. *Studijní program: Informatika 2009 - 4229, prezenční forma studia - bakalářský program* [online]. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023. [cit. 2023-05-08]. Dostupné z: https://bk.fit.cvut.cz/cz/prehled_4229.P.B.html.

33. *Specializace bakalářského studia* [online]. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023. [cit. 2023-05-08]. Dostupné z: https://fit.cvut.cz/cs/uchazeci/programy-a-obory/bakalarske-studium/specializace-bc-studia.

Obsah odevzdávaného archivu

- `README.txt` ... popis struktury implementace v archivu a instalační příručka
- `data` ... adresář s datovými soubory
  - `anketa_predmety` ... adresář s daty z Ankety ČVUT týkajícími se předmětů
  - `anketa_vyucujici` ... adresář s daty z Ankety ČVUT týkajícími se vyučujících
  - `my_data` ... adresář, kam implementace ukládá své výstupy
    - `graphs` ... adresář se grafy závislostí měřených metrik
- `demo.ipynb` ... prezentovatelný výstup celé práce
- `license.txt` ... licence pro užití programu
- `requirements.txt` ... definice prostředí pro spuštění práce
- `src` ... adresář se zdrojovými kódy
  - `implementation` ... zdrojové kódy implementace
    - `demo.py` ... modul s implementací pomocných funkcí
    - `evaluation.ipynb` ... notebook s vyhodnocením doporučování
    - `exploration.ipynb` ... notebook s průzkumem dat
    - `modeling.ipynb` ... notebook s implementací doporučovacího systému a měřením
    - `prepare_test_data.ipynb` ... notebook s rozdělením dat na trénovací a testovací
    - `preprocessing.ipynb` ... notebook s předzpracováním dat
  - `thesis` ... zdrojová forma práce ve formátu LaTeX
- `thesis.pdf` ... text práce ve formátu PDF