FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Návrh heuristických algoritmů v rámci aplikace Wowee
Student:	Bc. Michal Šanda
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem práce je analyzovat, navrhnout a implementovat nástroje pro podporu prodeje eshopů s využitím platforem Shoptet a Wowee.

1) Prostudujte možnosti API platformy Shoptet, existující BI systémy vhodné pro e-shopy a využití služby Wowee.cz.
2) Na základě získaných informací analyzujte a navrhěte heuristický algoritmus pro vyhodnocování vhodných produktů pro reklamní kampaně a
heuristický algoritmus odhadující vhodnost dárku pro uživatele Wowee.
3) Implementujte prototyp, který bude využívat data z doplňku pro Shoptet a bude nabízet integraci s vybranými BI systémy.
4) Otestujte a vyhodnoťte výstupy implementovaných algoritmů.
5) Zhodnoťte náklady a přínosy zvoleného řešení a navrhněte další možnosti využití.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 9. prosince 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Návrh heuristických algoritmů v rámci aplikace Wowee

Bc. Michal Šanda

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petra Pavlíčková, Ph.D.

26. května 2020

Poděkování

Děkuji Ing. Petře Pavlíčkové, Ph.D. za vedení této práce a mnohočetné konzultace. Dále děkuji Bc. Michalu Maněnovi za příležitost podílet se na vývoji aplikace Wowee, další poděkování patří Mgr. Petru Novákovi, Ph.D. za konzultaci časových řad. Největší díky patří mým rodičům za korekturu textu a přítelkyni za trpělivost.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu 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 předpisu, 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 26. května 2020 . . . . . .

Odkaz na tuto práci

Šanda, Michal. *Návrh heuristických algoritmů v rámci aplikace Wowee.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Cílem práce je nalézt vhodné metody pro vybrání co nejlepší možnosti pro reklamu a ohodnocení vhodnosti sbírky v rámci aplikace Wowee. Práce vychází z historických dat e-shopů, které využívají platformu Shoptet. K získání těchto dat je použito Shoptet API. Teoretická část se zabývá poznatky ohledně chování zákazníka, podobností se známými problémy, procesem vývoje a technologickými trendy. Praktická část využívá poznatky z teoretické části: Popisuje návrh databázového modelu, implementaci jednotlivých skriptů, datovou analýzu a využití dat k výše zmíněným účelům v rámci aplikace Wowee.

Klíčová slova Shoptet, Wowee, API, analýza dat, e-shop, algoritmus, časové řady, trendy

Abstract

The purpose of this thesis is to find how to select the best options for advertising and rating collections for a specific user of Wowee application. This thesis is based on historical data from e-shops running on the Shoptet platform. Data are obtained using Shoptet API. The theoretical part is about customer behavior, similarities with known issues, development process, and technological trends. The practical part makes use of the data from the theoretical part - It describes the database model, scripts implementations, data analysis, and the usage of data for the purposes above.

Keywords Shoptet, Wowee, API, data analysis, e-shop, algorithm, time series, trends

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Narozeniny, den, který má rád snad každý. Tradicí je takového člověka obdarovat a to ideálně tím, co by si přál. Ne vždy je však možné, aby jedinec mohl obdarovanému dát to, co si právě přeje. Může to být třeba proto, že nezná jeho skutečné tužby nebo tím, že nemá dostatek finančních prostředků. A právě pro takové případy je tu aplikace Wowee.

Wowee nabízí uživatelům tvorbu takzvaných sbírek. Typicky se jedná o sbírky na nějaká přání nebo dárky. Na tyto sbírky se uživatelé pak mají možnost složit formou jednotlivých příspěvků. Téma jsem si zvolil, protože s vývojem aplikace Wowee mám již zkušenosti z bakalářského předmětu Softwarový projekt a též jsem o tomto projektu psal bakalářskou práci.

Cílem práce je nalézt způsob, jakým zhodnotit historická data e-shopů, ke kterým má Wowee přístup skrze doplněk pro platformu Shoptet. Tato data je nutno analyzovat a vybrat vhodný způsob jejich dalšího zpracování pro implementaci. Cílem je stanovit algoritmus, který vybírá vhodnou kategorii (nebo produkt) k reklamě. Těchto vztahů dále využít ke stanovení vhodnosti sbírky pro konkrétního uživatele.

Toto téma je zcela jistě aktuální. S reklamou se stýká každý uživatel internetu prakticky neustále a její správné zobrazení je žádoucí jak pro inzerenta, tak pro uživatele. Druhým důvodem je vzestup platforem využívající princip crowdfundingu a tedy příležitost pro projekt Wowee prorazit.

Postup pro řešení problému jsem zvolil následující: Pochopit zákazníkovo chování, prozkoumat podobné práce, které se zabývají obdobnými problémy a prozkoumání možností samotné implementace. Dále dle získaných dat z API Shoptet zvolit vhodný přístup k implementaci a vytvořit prototyp.

Práce má ve výsledku dvě části — teoretickou a praktickou. Teoretická část se nejdříve zabývá výše zmíněným chováním zákazníka a jeho potřebami. Tyto znalosti jsou potřeba k pochopení proč je třeba aby reklama v aplikaci Wowee odpovídala trendům a uživatelskému chování. Obdobně aby ohodnocení při vytváření sbírky nabývalo smysluplné hodnoty určující její vhodnost.

Dále jsou v teoretické části rozebrány známé obdobné problémy jako je ten, který řeší tato práce v praktické části, tedy primárně zvolení co možná nejlepší volby produktu, či kategorie pro reklamu.

Součástí teorie je i rozebrání samotných jednotlivých přístupů implementace vedoucí k řešení tohoto problému. Těch existuje celá řada a každý má poměrně specifické nároky na vstupy. Samozřejmostí v teoretické části je seznámení čtenáře s aktuálními trendy v IT technologiích jako jsou REST nebo Docker kontejnery, těchto technologií se pak využívá i v praktické části.

Praktická část se pak přímo věnuje projektu Wowee a možnostmi API Shoptet. Jako první je zde představení projektu Wowee, kde se lze dočíst, že projekt vzniká dlouhodobě na ČVUT. Dále se praktická část zaobírá možnostmi Shoptet API, skriptu pro migraci dat z tohoto zdroje a databází určenou pro tato data. Další částí je datová analýza. Ta spočívá ve dvou částech — první je analýza jak se chovají provozovatelé e-shopů vůči jednotlivým atributům a druhá, jaké informace lze ze získaných dat vyčíst.

Konečně poslední část praktické části nabízí náhled na algoritmus a implementaci API, realizující samotný výběr co možná nejlepší položky k reklamě a ohodnocení vhodnosti sbírky pro uživatele.

Kapitola 1

Představení problematiky

Internetové obchody stejně jako internetová reklama jsou dnes již spojeny s každodenním životem. Navíc v období koronavirové karantény toto platí více než kdy jindy.

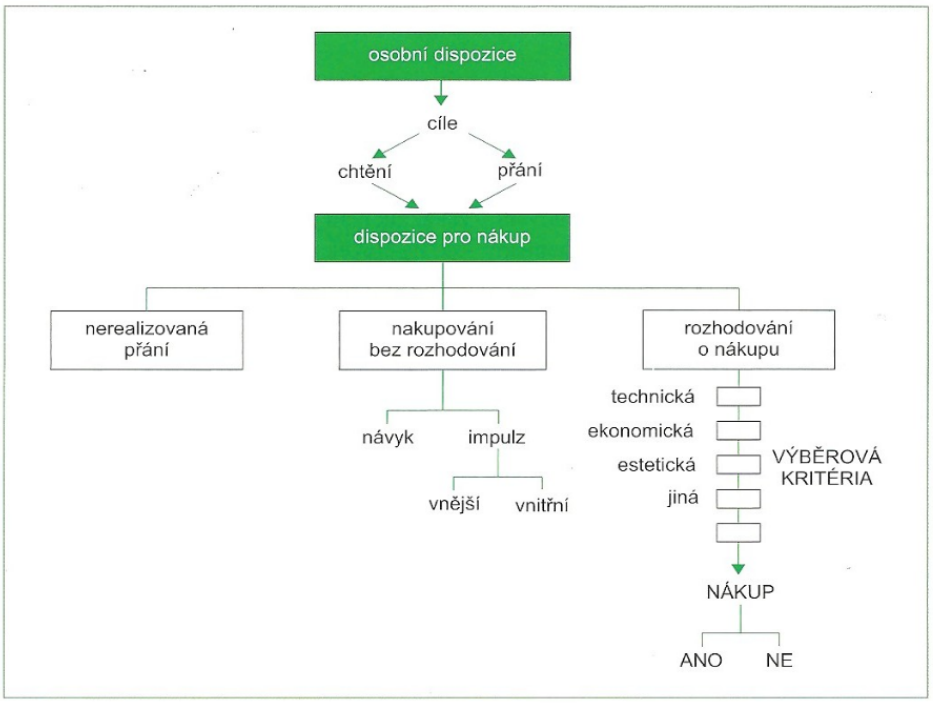
V případě reklamy to dokazují i snahy uživatelů a dnes často i vlád, kteří se snaží omezit sběr dat — ostatně na to jsem již narazil ve své bakalářské práci, která je taktéž spojená s tímto projektem [1]. Díky těmto datům lze vytvářet profily jednotlivých uživatelů a předkládat jim obsah, který je s velkou pravděpodobností pro ně relevantní.

Zobrazování „správného“ výsledku však není věc jen profilace, ale objevuje se i v samotném chování uživatele. Díky pokusům již poměrně jasně víme, jak se zákazník zachová v případě navrácení výsledku vyhledávání produktu na e-shopu. V případě, že produkt nebyl nalezen, zákazník odejde. V případě, že produkty nalezeny byly, ale nejsou relevantní, zákazník s největší pravděpodobností odejde a už se nenavrátí. Neboli jinými slovy, navrácení relevantního výsledku je pro získání uživatele a následnou konverzi velice důležitou vlastností [2] a to jak pro e-shop, tak pro uživatele.

1.1 Chování zákazníka

Tato kapitola čerpá z [3], o tom jak přemýšlí zákazník resp. to jak se rozhoduje je vidět na obrázku 1.1 na straně 4 z této knihy. Každý zákazník má nějaké představy o tom, co by chtěl pořídit v dané kategorii obchodu. Tyto potřeby lze rozdělit na to, co skutečně chce a to, co by si přál. Ostatně na právě druhou větev této úvahy je založen celý projekt Wowee (viz 5 na straně 31). I samotné „přání“ má svá stádia:

1. touha
2. přání
3. snažení



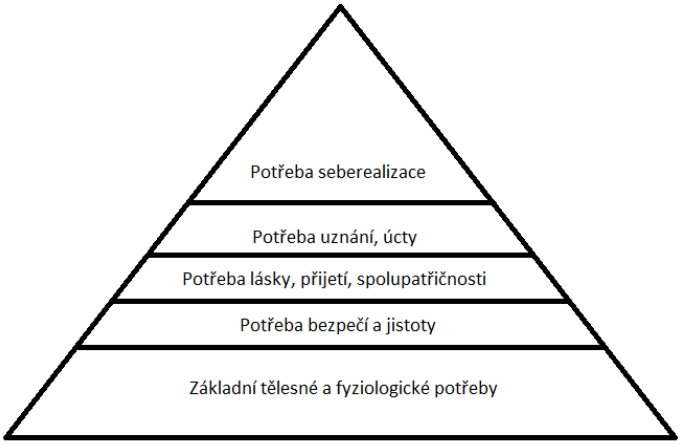
Obr. 2.1 Vliv osobních dispozic na rozhodování při nákupu

Zdroj: Komárková R. – Rymeš M. – Vysekalová J.: Psychologie trhu, 1998

Obrázek 1.1: Osobní dispozice zákazníka z knihy [3]

Touha nemá konkrétní podobu, tu získává teprve, když přejde do fáze přání. Ve stádiu snažení již jedinec ví, co přesně chce. Obecné pořadí potřeb zachycuje nejlépe známá Maslowova pyramida viz obrázek 1.2 na straně 5. Další důležitou vlastností při rozhodování zákazníka je, aby měl dostatek informací a nepřecházel do stavu „bezmoci“. Tomu odpovídá i výše popsané chování při vyhledávání. Ze zmíněných pokusů jsou zajímavé dva závěry — zákazník využívá online nákupu v první řadě pro úsporu peněz a poté času a díky dnešnímu dění by se zcela jistě dalo říci, že i kvůli ochraně svého zdraví. I tyto fakty podporují nutnost navracení co nejrelevantnějších výsledků a to tak, aby co možná nejvíce podpořily zákazníkovo chování. Podpora chování zákazníka resp. uživatele je dnes vlastně hlavním nástrojem (nejen) internetové reklamy (jde o často zmiňovanou profilaci).

To, zda zákazník ve finále nakoupí, či nikoliv, však neovlivňují jen tyto faktory, ale i to, jaký má internetový obchod design, zda-li je bezpečný a důvěryhodný a například i to, zda není „technickým overkillem“ a tím pádem zapomíná na skutečné potřeby zákazníka. Závěrem této podkapitoly bych rád



Obrázek 1.2: Maslowova pyramida potřeb

uvedl jeden citát hovořící za vše: „Lidské chování je proměnlivé a závislé na řadě faktorů, takže ho lze predikovat jen s jistou dávkou pravděpodobnosti.“[1].

1.1.1 Model placení za internetovou reklamu

Reklama založená na platbách za zobrazení případně za uživatelův úplný proklik (PPC — pay per click) je dnes již standardem na internetovém trhu. S rozšířením počtů internetových obchodů a obecně internetových služeb tento systém pokročil ještě dále.

V dnešní době je již poměrně běžně aplikován model, kdy je provozovateli reklamy placeno nejen za výše zmíněné úkony, ale i za dokončenou registraci nebo provedený (dokončený) nákup v případě internetového obchodu. Tento model se nazývá CPA (z anglického cost per action). V důsledku nemusí jít ani o interakci skrze reklamu, ale například o referenční link.

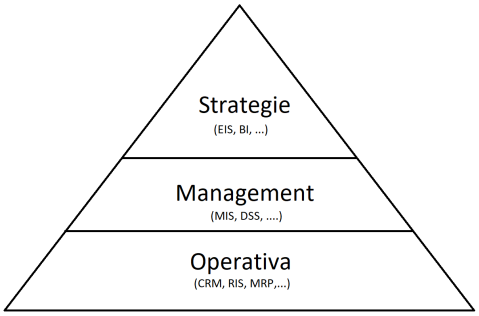
Samotné PPC je tedy jen jednou podmnožinou CPA.

1.2 Business inteligence systémy

Jednou z častí zadání je možnost integrace s business inteligence (dále jen BI) systémy. Tyto systémy jsou nedílnou části pro další plánování a rozhodování vrcholného managementu. Již z předchozí věty vyplývá, že jde o poměrně složitý, ale mocný nástroj. Tyto nástroje mají často na vstupu data, která jsou plné chyb, nedodržují dané standardy nebo jich například části chybí. Naopak na výstupu je situace naprosto opačná. Výstupy z BI systémů jsou srozumitelné a koncový uživatel se nemusí vůbec starat o to, jak data ze kterých

[1]Jitka Vysekalová a kolektiv: Chování zákazníka: Jak odkrýt tajemství černé skříňky, Praha, 2011, str.237

vychází vlastně vypadají. Nejčastěji se dnes řešení, kde databáze obsahuje agregované a konsolidované záznamy nazývá data warehouse (DWH). Zpracování dat před uložením do DWH je proces známý jako ETL. ETL je akronym z angličtiny a znamená extract, transform, load — neboli extrakci, transformaci a uložení dat. U těchto systému se neklade důraz na jejich rychlost — DWH udržuje historická data — ale právě na důležitost a smysluplnost výstupu[4]. Na obrázku 1.3 na straně 6 je zobrazena hierarchie informačních



Obrázek 1.3: Hierarchie IS [5]

systémů, BI systémy řadíme na vrchol této pyramidy, konec konců jedná se o systémy ze kterých se tvoří strategie. Strategie je svým způsobem reakce na řadu výstupů v podobě analýz, reportů nebo ukazatelů jako je KPI, které nám tyto systémy přináší. Pro pokročilou analýzu dat se dnes využívá data miningu — nejčastěji rozhodovacích stromů, lineární regrese nebo shlukování pomocí K-means. Systémy jsou tedy poměrně složité, což by v dnešní době nemělo platit o jejich uživatelském rozhraní — to by mělo být naopak co nejsnadněji použitelné[4][6].

1.2.1 Vybraní zástupci

Keboola

Jedním z významných zástupců pro ETL proces je společnost Keboola. Jde původně o pražskou společnost, která se bez investic dostala až do Silicon Valley, které je považováno za nejvýznamnější místo pro svět IT. Samotný produkt nabízí cloudové zpracování dat až z tisíců zdrojů. Platforma je pak postavena na technologiích Amazon Web Services a Microsoft Azure. Nástroj nabízí uživateli plnou kontrolu a přizpůsobení nad tím, co se s daty děje, přičemž se jedná o kompletní analytický stack — je zde zázemí pro jazyky jako je R, Python, Ruby a mnoho dalších. Známé české společnosti, které služby využívají jsou například Heureka.cz nebo Rohlík.cz. Lze si jen těžko představit, že by data pocházející z těchto projektů někdo analyzoval a transformoval bez nástroje, který mu to usnadní [7][8].

Nejvíce však Keboola asi v současné době vešla do povědomí díky projektu „Chytrá karanténa“ a sdružení COVID19CZ, které vznikly během koronavirové pandemie. Řada jedinců z laické veřejnosti projektu nevěří a to zejména kvůli případnému sdílení dat o pohybu se státní infrastrukturou. K tomuto by však nemělo vůbec docházet [9]. Navíc stát jako takový, tak či tak, má k daným informacím přístup [10].

Microsoft PowerBI

Microsoft PowerBI je nástroj pro grafickou vizualizaci dat, díky kterému má vrcholný management skvělý přehled o tom, co se děje v jeho společnosti. Jde o interaktivní panely a sestavy. Nástroj je oblíbený zejména díky podobnosti s nástroji sady Office 365 (Word, Excel, ...). Samotná integrace s aplikací Excel je samozřejmost a to řadě spolčeností vyhovuje. Microsoft dále nabízí další služby, které lze propojit s PowerBI jde například o Power Apps (rychlá tvorba aplikací) a Power Automate (nástroj pro automatizaci obchodních procesů — může jít například o automatické reakce na příchozí email)[11][12][13].

1.3 Známé problémy

Známý problém se kterým spatřuji jistou podobnost mého zadání je **Internet shopping optimalization**. Tento problém označuje situaci, kdy je třeba řešit situaci, ve které je na vstupu množina produktů, o které má zákazník zájem a dále M internetových obchodů ve kterých je možnost produkty nakoupit. Celkovým výstupem je splnění „nákupního seznamu“ za co možná nejmenší cenu. Tento problém se vyskytuje i v dalších variacích, kdy se například uvažují i slevy. Všechny tyto varianty problému jsou NP-těžké. Pro řešení v rozumném časovém horizontu s rozumně velkou pamětí je tedy nutné využít heuristický algoritmus[14][15].

Další podobnost lze nalézt s prací [16], která se se věnuje přímo doporučování produktů na základě jejich atributů. Autor zde využívá Bayes klasifikátoru a genetického algoritmu evoluce.

Zmíním zde několik zajímavých myšlenek z této práce. Základní stavební kámen je, že většina doporučovacích systémů vychází z historických dat — to je poměrně logické, protože statisticky je zde velká šance na úspěch. Dále je zde uvedeno, že je lepší se zabývat každou kategorií produktů zvlášť — uživatelé mají různé preference skrze různé kategorie a s tím je víceméně spojeno i to, že každá kategorie má své vlastní vlastnosti. Z toho dále autoři zmíněné práce usoudili, že je nejlepší použít pro jednoho uživatele i jeden jeho vlastní klasifikátor strojového učení.

Kapitola 2

Naskýtající se možná řešení

Z předchozí kapitoly se jako jedno z možných řešení jeví použití strojového učení (resp. některého z klasifikátorů) s kombinací některého heuristického algoritmu jako je genetická evoluce. Přístupů, jak k datům přistupovat, je ale více. Jedním z dalších přístupů, vzhledem k tomu, že může jít čistě o historická data, je zpracování časových řad. V nadcházejících podkapitolách bych tedy rád tyto přístupy představil.

2.1 Co pro nás znamená heuristika

Heuristika obecně znamená, že pro daný problém je nalezeno sub-optimální, přibližné řešení. Často pro tyto problémy neznáme přesný algoritmus, nebo nalezení jejich řešení trvá neúměrně dlouho. K řešení problému tedy využíváme zkušenosti, získané informace nebo ověřené vztahy[17].

Může jít tedy o jakýkoliv algoritmus splňující výše popsané podmínky. Za jeden z takových přístupů považuji například analýzu časových řad a následnou identifikaci trendů.

Dalšími přístupy, které jsou zmíněny v kapitole 1.3 na straně 7 může být strojové učení nebo například genetické algoritmy[18].

2.2 Predikce strojovým učením

Algoritmy pro predikování si lze představit matematickým zápisem jako

$$Y \approx f(x_1, x_2, x_3, \ldots, x_n)$$

Proměnné $x_1, \ldots, x_n$ představují tzv. příznaky a Y je vysvětlovaná proměnná. Algoritmus se snaží naučit vztahy mezi příznaky tak, aby co nejlépe nebo nejvíce platila vysvětlovaná proměnná [19]. Pro příklad využiji tabulky 2.1 na straně 10. Vysvětlovaná proměnná zde může být například pohlaví, predikční

funkce tedy na vstup přijme příznaky (sloupce) výška a město, na základě kterých se snaží určit výstup.

Výška	Město	Pohlaví
160	Olomouc	Ž
190	Praha	M
170	Olomouc	Ž
168	Olomouc	?

Tabulka 2.1: Příklad data setu

Podle typu proměnné je pak nutné volit algoritmus resp. hledat funkci pro predikci.

2.3 Druhy atributů

Atributy jsou dvou druhů, které se ještě dále dělí[20]:

- Kvalitativní
  - Nominální – názvy nějaké vlastnosti, bez pořadí
  - Ordinální – lze určit pořadí mezi nimi (pořadí běžců: 1,2,3)
  - Binární – lze rozhodnout ano / ne
- Kvantitativní
  - Spojité – podobně jako spojitá matematická funkce, atribut může nabývat hodnoty z intervalu – např. váha, cena
  - Diskrétní – Konečný počet možných hodnot např. PSČ, den v týdnu

2.4 Regrese vs. klasifikace, učení s učitelem a bez

V případě, že hodnoty vysvětlované proměnné mají relativně málo hodnot, tak se jedná o problém klasifikace. V případě, že vysvětlovaná proměnná je spojitá nebo má velmi hodně hodnot, jedná se o problém regrese. Pro problém klasifikace i regrese existuje řada přístupů a algoritmů. Algoritmy lze rozdělit ještě na dvě další podstatné kategorie – **učení bez učitele a naopak supervizované učení**.

Do první zmíněné kategorie patří např. algoritmy K-means nebo hierarchické shlukování. Do supervizovaných naopak patří lineární a logistická regrese, algoritmus K-nn, SVM a **rozhodovací stromy**.

2.5 Rozhodovací stromy a lesy

Strom i les jsou obecně grafové struktury. Les je takový graf, který neobsahuje kružnici. Strom je graf, který neobsahuje kružnici a navíc je souvislý[21].

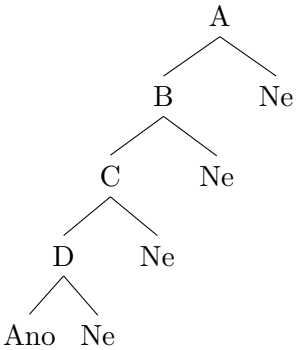
Rozhodovací stromy jsou nejčastěji binární. Každý vnitřní uzel má tedy právě dva potomky. Takovými stromy lze poměrně jednoduše ukládat vztahy mezi otázkami, na které je odpověď ano nebo ne. Všechny vnitřní vrcholy reprezentují posloupnosti pravidel a v listech nalezneme odpovědi na původní otázku. Z výše popsaného vyplývá, že původní otázka má formu logické formule. Tedy například pro otázku „Mám jít do restaurace na oběd?“, kdy příznaky budou:

- A = mám hlad
- B = je poledne
- C = cena obědu < 150
- D = nemám oběd v lednici

Je ekvivalentní logická formule:

$$A \ \wedge B \wedge C \wedge \neg D$$

p Pro kterou lze postavit například následující strom — předpokládejme následující odpovědi na otázky příznaků a to vlevo odpověď „Ano“ a odpověď „Ne“ vpravo:



Les takových grafů může obsahovat N. Každá komponenta souvislosti grafu, který je les, tvoří strom.

O tom jakým způsobem se staví stromy a lesy bude následující kapitola.

2.6 Stavba rozhodovacího stromu

Konstrukce optimálního stromu je problém z kategorie NP-complete. Pro stavbu je tedy třeba použít algoritmus, který nám dá s jistými omezeními

přibližný nejlepší výsledek. V případě stromů se jedná o hladové algoritmy označované jako ID3 a jeho varianty. Osobně jsem se strojovým učením setkal v Pytohnu, proto i nadále budu vycházet z jeho knihovny **scikit**. V té je použit algoritmus CART, což je dle dokumentace vyšší verze algoritmu C4.5, který je odvozen od ID3[22].

ID3 algoritmus používá pro sestavení informační zisk, který je definován pomocí entropie. Entropie je laicky řečeno „míra překvapení“ definována jako:

$$H(X) = -\sum p(X) \log p(X)$$

Jde tedy o minus součet pravděpodobnosti jevu X, který je vynásoben logaritmem této pravděpodobnosti. Samotný informační zisk pak jako entropie jevu X minus entropie jevu X za podmínky jevu Y:

$$I(X, Y) = H(X) - H(X|Y)$$

Algoritmus C4.5 naopak používá Gini index, který je mírou toho, že prvek nebude klasifikován špatně:

$$GI(D) = 1 - \sum_{i=0}^{k-1} p_i^2$$

Rozdíl ve zmíněných algoritmech je ten, že ID3 neumí pracovat s chybějícími daty a vyžaduje kategorická data. C4.5 i CART již umí navíc pracovat i s numerickými daty a vypořádají se s chybějícími hodnotami. Algoritmus CART je schopen vypořádat se jak s regresí tak i s klasifikací[23].

Samotný běh lze nejsnadněji ilustrovat na algoritmu ID3. Ten rozdělí data podle vybraného atributu na dvě části, a snaží se maximalizovat kritérium — informační zisk. Toto je aplikováno do té doby, než se splní kritérium pro zastavení – např. maximální hloubka stromu [19].

2.7 Ensemble metody

Ensemble metody — doslovný český překlad by měl být společné metody. Jde o metody, které využívají skládání více modelů. Samotné stromy mají tendenci se přeučit na konkrétních datech. V ensemble metodách se vytvoří několik modelů a to buď „paralelně“ — příklad náhodný rozhodovací les, nebo „sekvenčně“ — to je svým způsobem například metoda AdaBoost. V prvním případě jde o tzv. bagging — boostrap aggregating, který je vlastně výběrem prvků s opakováním. V druhém případě jde o tzv. boosting, kde modely nejsou nezávislé, ale jdou za sebou. Každý následující model je ovliněn modelem předchozím. V této práci, budu i nadále vycházet z toho co nabízí balíček scikit pro Python a jeho dokumentace [24] [25]. Rozeberu zde dvě respektive tři známé metody, které jsou v tomto balíčku obsaženy.

2.7.1 Random Forest Classifier (Regressor)

Náhodné lesy vycházejí z principů rozhodovacích stromů, viz kapitola 2.5 na straně 11. Jak již zmíněno v odkázané kapitole, les má n stromů. Tyto stromy jsou náhodné a mohou být v případě náhodných lesů jednoduché — tedy nemusí mít velkou hloubku.

Stromy v lese jsou nezávislé. Tato vlastnost tvoří výpočetní výhodu, protože proces klasifikace či regrese lze paralelizovat. Výsledkem klasifikace pak je nejčastější hodnota všech klasifikátorů. Rozhodovací stromy mohou však dosahovat velkých velikostí — k tomu se samozřejmě váže i velikost tohoto klasifikátoru. Velikost náhodného lesa lze samozřejmě ovlivnit. Jednou z variant je omezit počet stromů v lese, další je maximální počet příznaků, nebo jejich hloubku. Větší vliv na chování mají jednotlivé klasifikátory uvnitř lesa. Čím menší budou, tím dosáhneme menšího rozptylu, ale většího zkreslení. Optimální množství pro regresi uvádí dokumentace defaultní nastavení — tedy použití všech. Pro případ klasifikace je to $\sqrt{|features|}$, tedy druhá odmocnina z celkového počtu příznaků. Ve výchozím nastavení se taktéž používá bootstrap, tedy výběr s opakováním. Největší možná celková velikost modelu v O-notaci, kde M je počet stromů a N počet vzorků je:

$$\mathcal{O}(M * N * \log N)$$

2.7.2 AdaBoost

AdaBoost je další z poměrně známých ensemble metod. AdaBoost lze nalézt v několika verzích, já představím tu základní, které se říká diskrétní AdaBoost.

Jde o metodu z kategorie učení s učitelem a i tady tedy nalezneme data set obsahující jak příznaky, tak vysvětlovanou proměnnou na kterých se algoritmus naučí ohodnocení. V tomto případě se počítá s tím, že vysvětlovaná proměnná nabývá hodnot +1 nebo -1. Tak jsou rozlišeny třídy do kterých daný prvek spadá.

Dále algoritmus udržuje jednotlivé váhy jednotlivých prvků. Váha se v každém u správně ohodnoceným prvků snižuje a naopak špatně ohodnoceným prvkům zvyšuje (a budou tedy dále algoritmus i více ovlivňovat). Na počátku jsou samozřejmě tyto váhy u všech prvků stejné.

V každém cyklu se vybírá tzv. slabý klasifikátor (h_t) ze všech možných a to tak, aby se minimalizoval horní odhad chyby (ϵ_t) klasifikátoru — to je vlastně součet vah chybně klasifikovaných vzorků (zde se využijí předchozí zmíněné váhy prvků).

Pro každý takový klasifikátor pak algoritmus vypočítá jeho váhu (a_t), který opět jako v předchozím případě říká, jak moc nám celkové ohodnocení ovlivní. Dále platí, že čím nižší chyba klasifikátoru, tím vyšší je jeho váha. Dále se aktualizují váhy jednotlivých prvků[26].

Výsledný klasifikátor H je pak lineární kombinace tzv. slabých klasifikátorů[27]:

$$H(x) = sgn(\sum_{t=1}^{T} a_t h_t(x))$$

2.7.3 Voting classifier

Voting classifier (dále jen VC) jsem zde zařadil, protože mi to osobně přijde jako zajímavá built-in funkcionalita. V případě skládaných klasifikátorů se vždy jedná o klasifikátory stejného typu — např. u lesa jsou to jednotlivé stromy. VC je třída, která nám umožní použít jakýkoliv klasifikátor, který nám knihovna nabízí. Princip je následující: Vytvoříme klasifikátory, které jsme si zvolili a ty předáme VC. VC poté funguje podobně jako vyhodnocení u náhodného lesa. Výsledek, který je nejpočetnější zvítězí. V případě rovností se zvolí poslední klasifikace. Nejde tedy vlastně ani tak o samotný klasifikátor, jako spíš o přístup k vyhodnoceným datům.

2.8 Genetický algoritmus evoluce

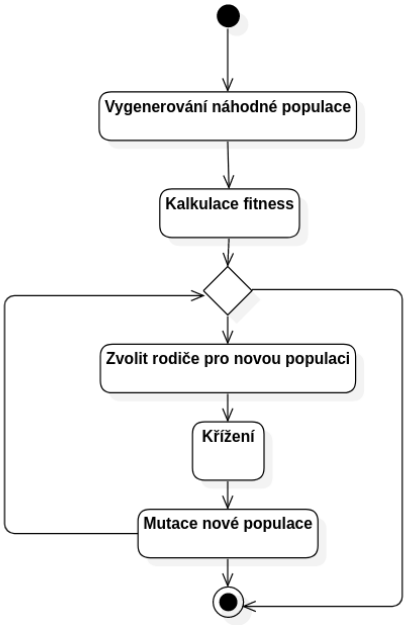
Genetický algoritmus evoluce, jak již název napovídá, lze přirovnat ke skutečné evoluci v lidské populaci. Evoluční algoritmus lze použít k hledání extrémů, nebo jiných optimálních hodnot. Příklad situace, kdy tento algoritmus chceme využít buď funkce:

$$f(x_1, \ldots, x_n)$$

do které potřebujeme dosadit parametry $x_1, \ldots, x_n$, které však neznáme. Nemusí nutně jít o nějakou funkci, ale o logickou formuli:

$$(A \wedge B \neg C) \vee (A \ \wedge \neg D)$$

a mnoho dalších případů. Základní princip GA evoluce na diagramu 2.1 na straně 15. Jako první se vygeneruje náhodná populace, spočítá se hodnota fitness — tedy to, jak moc jedinec vyhovuje v rámci hledání. Dále se vyberou rodiče ze kterých vznikne nová populace. Podobně jako v přírodě i na rodiče se použije křížení jednotlivých genů a tím vznikne nový jedinec. Takto postupně vygenerujeme novou populaci. Jako poslední úkon novou populaci ještě tzv. zmutujeme neboli provedeme náhodné změny, které nejsou založeny na žádném jedinci z populace. Proces vytváření nových populací může být (a také často je) omezen jejich počtem.



Obrázek 2.1: Princip genetického algoritmu evoluce

2.8.1 Stanovení fitness funkce

Fitness funkce hodnotí konkrétního jedince, jak moc vyhovuje hledaným hodnotám. Jde nejčastěji o zobrazení:

$$f : R^n \rightarrow R$$

Může jít o nějaké maximum, minimum nebo například objem materiálu. Stanovení funkce je vůbec nejtěžší částí pro evoluci. „Je nutné vědět čeho se má dosáhnout a z čeho lze vycházet“[2]. Lze vycházet z průměrů, četností, odchylek od nich nebo jen hledat prosté zmíněné maximum, či přibližování se k určité konstantě, nechť K je konstanta:

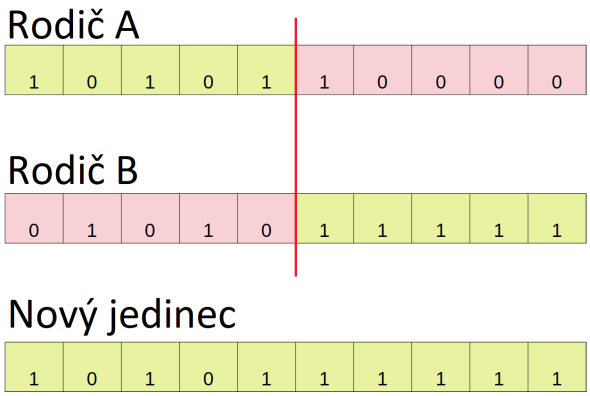
$$fitnessRating = |K - value|$$

Funkce by však měla být normalizovaná a být v rozsahu od nuly do jedné. Správně tedy po ohodnocení každého jedince je nutné ještě jeho hodnotu fitness podělit součtem všech fitness hodnot celé populace[28].

2.8.2 Výběr rodičů

Pro výběr rodičů z aktuální generace lze volit několik způsobů [29]. Každý z nich vyžaduje seřazení jedinců podle vypočteného fitness skóre popsaného

[2]Ivan Zelinka: Umělá inteligence v problémech globální optimalizace, Praha, 2002, str. 33



Obrázek 2.2: Křížení jedinců – zelená část bitů se předává novému jedinci.

výše. Důvod, proč se zde volí sofistikovanější metody, je snaha předejít přeučení algoritmu resp. předejít zamrznutí v lokálním extrému. Vybranými zástupci těchto metod jsou elitismus a výběr ruletou.

Elitismus

Elitismus je asi nejjednodušší z přístupů a spočívá jednoduše v tom, že část nejlepší populace (=elita) přežije do další generace. Jedince o které přijdeme, nahradíme novými pomocí křížení z vybraných rodičů. Procento elitismu je pak jeden z parametrů pro spuštění GA.

Ruletový výběr

V tomto případě je šance každé jedince na přežití úměrná jeho hodnotě fitness. Ruletou se tento přístup nazývá, protože si lze každého jedince představit tak, že zabírá n míst na ruletě (přesně podle fitness) a kulička představuje náhodu. Jedinci s více políčky mají větší šanci přežít.

2.8.3 Křížení

Jedinci jsou nejčastěji reprezentováni jako binární řetězce. Ve chvíli kdy algoritmus vybere rodiče, přichází na řadu křížení. Křížení jedince probíhá náhodným zvolením pozice nebo rozsahu kopírovaných části prvního z rodičů. Doplněk rozsahu nebo pozici je pak určen pro geny druhého rodiče, viz obrázek 2.2 na straně 16

2.8.4 Mutace

Mutace probíhá již nad novou generací. Spočívá ve zvolení počtu mutací (resp. zvolení jedinců určených k mutaci) a následnou náhodnou změnou (v případě

binárního řetězce inverzí bitu, v případě jiných hodnot může jít například o generování náhodného čísla z daného rozsahu parametru).

2.9 Časové řady

Jedním s dalších přístupů k získaným datům je analýza tzv. časových řad. Časová řada se vyznačuje tím, že vývoj její hodnoty (často např. objemy, ceny) jsou závislé právě na čase. Taková data jsou nejčastěji získána v nějakém intervalu, jsou tedy od sebe nejčastěji vzdáleny stejnou dobu. Příkladem můžou být všem známé grafy zobrazující přehledy za rok, týden, apod.

Časové řady lze charakterizovat podle jejich vlastností — může tak jít o řady intervalové (vývoj v měsíci), okamžikové (průměrný plat od roku x do roku y). Dále je lze dělit podle periodicity, nebo druhu sledovaných dat[30].

2.9.1 Vybrané operace s řadami

Informace, které lze získat, mohou být buď souhrnné nebo přímé charakteristiky. Mezi souhrnné řadíme v případě intervalových řad prosté součty nebo aritmetické průměry (definice 2.1 na straně 17). V případě okamžikových řad se nejčastěji využívá chronologického průměru (definice 2.2 na straně 17).

$$\bar{x} = \frac{1}{n}\left(x_1 + x_2 + \ldots + x_n\right) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.1}$$

$$\bar{x} = \frac{\frac{x_1}{2} + x_2 + \ldots + x_{n-1} + \frac{x_n}{2}}{n-1} \tag{2.2}$$

Pro řady, které mají stejné vzdálenosti mezi jednotlivými hodnotami lze použít i další zmíněné charakteristiky:

Absolutní přírustek

$$\Delta y_t = y_t - y_{t-1} \qquad t = 2, 3, 4, \ldots, n$$

Průměrný absolutní přírustek

$$\bar{\Delta} = \frac{y_n - y_1}{n-1}$$

Relativní přírustek

$$\delta_t = \frac{y_t - y_{t-1}}{y_{t-1}} \qquad t = 2, 3, 4, \ldots, n$$

Koeficient růstu

$$k_t = \frac{y_t}{y_{t-1}} \qquad t = 2, 3, 4, \ldots, n$$

Průměrný koeficient růstu

$$\bar{k_t} = n - 1\sqrt{\frac{y_n}{y_1}}$$

2.9.2 Rozklad a hledání trendu

Grafická reprezentace časové řady tvoří křivku, proto je správné ptát se, zda-li je možné tuto křivku reprezentovat jako matematickou funkci. Tato funkce by při jejím správném stanovení mohla fungovat i jako funkce predikční.

Rozklad časové řady má čtyři složky:

- trend D_t – specifikuje dlouhodobé chování – růst či klesání
- sezónnost S_t – opakování v rámci dne, měsíce, ...
- cyklická složka C_t – jevy, které se sice opakují, ale nezávisí na čase
- náhodná složka E_t – zbývající složka ke které nelze najít vztah

Výsledná funkce je pak součet těchto složek (tzv. aditivní rozklad):

$$X_t = D_t + S_t + C_t + E_t$$

Samotný trend (složka D_t) může být různého typu:

- lineární – $D_t = \alpha_0 + \alpha_1 t$ (první dif. cca konstantní)
- kvadratický – $D_t = \alpha_0 + \alpha_1 t + \alpha_2 t^2$ (druhá dif. cca konstantní)
- exponenciální – $D_t = \alpha_0 \alpha_1'$ (koef. růstu cca konstantní)
- logistický – $D_t = \frac{K}{1+\alpha_0 \alpha_1'}$ (křivka prvních dif. se podobá normálnímu rozdělení)

Při hledání samotné funkce trendu se používá metoda nejmenší čtverců, pokud ale hledáme samotný typ, vychází se z tzv. diferencí, což znamená spočítání absolutního přírůstku. Pokud od sebe odčteme první dva absolutní přírůstky dostaneme druhou diferenci a tak dále.

2.9.3 Alternativní přístup

Alternativním způsobem jak stanovit trendy z dat je pomocí tzv. klouzavých průměrů. Tato metoda umožňuje nalézt trendy i tam, kde data nelze popsat jedinou křivkou. Klouzavé průměry nejsou nic jiného než průměr z n po sobě jdoucích. Existuje několik přístupů, jak tyto průměry počítat — uvedu alespoň dva z nich.

Jedním je obyčejný klouzavý průměr (simple moving average - SMA). Jde o spočítání průměru za posledních n dní (nebo jiné veličiny) - blíže přesná definice 2.3.

$$\overline{x}_{\text{SMA}} = \frac{x_M + x_{M-1} + \cdots + x_{M-(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} x_{M-i} \tag{2.3}$$

Druhou variantou je tzv. centrovaný klouzavý průměr (central moving average). V tomto případě je třeba, aby počet hodnot za které klouzavý průměr počítáme, byl lichý. V takovém případě se klouzavý průměr počítá pro „prostřední“ hodnotu a $\frac{n-1}{2}$ hodnot „vpřed“ a „zpět“ od středu — pro lepší představu uvedena tabulka 2.2 na straně 19, která zobrazuje klouzavé průměry v délce tři.

Tabulka 2.2: Rozdíl mezi klouzavými průměry

rok	**cena**	**SMA**	**centrovaný MA**
2000	1		
2001	2		2
2002	3	2	3
2003	4	3	4
2004	5	4	

Identifikace trendů je pak již snadný úkonem. Stačí porovnat aktuální hodnotu v daném měsíci s klouzavým průměrem. **Pokud je hodnota větší než klouzavý průměr, jde o trend rostoucí, v opačném případě o trend klesající**.

Kapitola 3

Proces vývoje

Proces vývoje (nejen informačního systému) je složení několika dílčích etap. Každá etapa by měla mít přiřazeno dostatečně zdrojů a to jak finančních, lidských, tak času. Tento cyklus je znám především jako „System development life cycle“ neboli zkráceně „SDLC“. Následující odstavce vycházejí především ze zdrojů [31] a [32].

3.1 Sběr požadavků

Sběr požadavků má za cíl ujasnit proč se systém nebo aplikace vyvíjí a definovat, jak přesně se má chovat a vypadat. Pro tento sběr informací by zákazník měl vyčlenit dostatek zaměstnanců, kteří by měli poskytnout co největší možnou součinnost — podchytí se tak skutečné potřeby lidí, kteří s daným systémem budou skutečně pracovat.

Požadavky jsou dvojího druhu: Funkční a nefunkční. Pod funkčními požadavky se skrývá vše spojené s funkcionalitami daného systému. Pod nefunkčními jsou naopak veškeré věci, které jsou se systémem spojené, ale nejedná se přímo o jeho funkci. Nejčastěji jde o požadavky na hardware, licence, nebo celkové nároky na infrastrukturu.

Jasně specifikované zadání předchází velkému počtu následných změnových řízení.

3.2 Analýza a návrh řešení

Na základě získaných dat je třeba sestavit jednotlivé případy užití, identifikovat jednotlivé procesy, jejich návaznosti a zainteresované osoby. Tato etapa je často zachycena v podobě UML diagramů.

Pomocí UML lze zachytit jak jednotlivé procesní průběhy, tak jednotlivé akce v aktivitách nebo celý objektový návrh. Notaci UML by měla rozumět většina vývojových týmů. Jednotnost značně ulehčuje práci. Dále řada nástrojů

jako je Enterprise architect umožňuje exporty z diagramu jako skeletony tříd přímo do programovacího jazyka.

3.3 Implementace

Implementace by měla respektovat zjištěné skutečnosti z analýzy. Na vhodných místech využít návrhových vzorů tak, aby se architektura aplikace dala snadno udržovat a případně rozšiřovat.

Samotný kód by měl podléhat tzv. „code review“, tedy kontrole kódu někým dalším, kdo není samotným autorem. Snáze se se tak odhalí prohřešky proti zavedenému „code style“, nebo hrubé chyby v návrhu.

S implementací nedílně souvisí i dokumentace celého řešení.

3.4 Testování

Testování lze rozdělit do několika kategorií. Může jít o :

- funkční testy — ověření, že dodávka splňuje zjištěné a požadované funkce
- unit testy — testy funkčnosti jednotlivých tříd a jejich metod
- integrační testy — ověření funkčnosti produktu jako celku v celém ekosystému
- systémové testy — ověření jednotlivých případů užití, lazení aplikace
- akceptační testy — provádí se na straně zákazníka s jeho týmem

Pro testování se velmi často používá metoda tzv. mockování (nejčastěji se tato technika dá nalézt u unit testů). Jde o princip, kdy testovací balíček umožňuje vytvořit objekt, který se „tváří“ jako ten, který potřebujeme pro zavolání nějaké funkce, či metody. Místo toho abychom použili příslušné volání funkce, či metody k získání tohoto objektu, testovací balíčky mají často metodu „mock“, která vytvoří svým způsobem „simulaci“ tohoto objektu. Snadno se tak dají otestovat jednotlivé části, návratové hodnoty apod. celého algoritmu.

3.5 Provoz a údržba

Pokud byl systém objednatelem akceptován, je dále třeba věnovat dostatečnou dobu proškolení jeho zaměstnanců, kteří budou systém využívat. Mimo samotných koncových uživatelů může jít i o celé ICT oddělení, které bude řešit případné problémy a chyby z provozu.

Co bude objednatelovo ICT oddělení řešit, záleží na definovaných smluvních vztazích. Je možné podepsat tzv. SLA (Service Level Agreement), což je

smlouva o dohodě poskytovaných služeb, kde dodavatel může i nadále poskytovat za úplatu další služby a především podporu. Vzhledem k cenám těchto služeb je častým jevem rozdělení samotné podpory na několik vrstev tak, aby v rámci SLA mohly být řešeny jen nejzávažnější problémy.

Nahlašování problémů je nejčastěji řešeno přes tzv. „Issue tracker system“, tedy systém který udržuje informace o jednotlivých uživatelských problémech. Z popisu problému v tomto nástroji by mělo být zřejmé co a za jakých okolností nastalo.

3.6 Dodávky a integrace

Samotné dodávky softwaru jsou dnes často automatizovány pomocí tzv. „continous delivery“ (dále jen CD) a „continoues integration“ (dále jen CI) nástrojů. Tyto nástroje zároveň často slouží jako verzovací systémy pro kód.

Systémy CI zajišťují otestování změn projektu a jeho následnou integraci do hlavní větve ze které vzniká později vydání.

Systémy CD jsou svým způsobem pokročilejší CI – navíc provádějí automatické nasazení, které se neprovede jen v případě, že se nepovedou testy[33].

Kapitola 4

Trendy v technologiích

4.1 REST

Následující odstavce budou vycházet z přednášky MI-MDW [34], které se věnují tomuto tématu. REST je zkratka pro Representational State Transfer. Jedná se o architektonický styl. Druhým slovem spojený s touto technologií je „RESTful“, což označuje webové služby implementující REST.

REST přináší 5 „omezení“:

- client-server — architektura musí být tohoto typu
- stateless — server neuchovává žádné informace o klientské session
- cachebility — možnost cachování
- layered system — API může být vystaveno na jiném místě než jsou uložena data
- uniform interface — jednotné rozhraní

REST pracuje s tzv. „resources“ česky by se řeklo zdroji. Tyto zdroje jsou identifikované pomocí URI — Uniform Resource Identificator (přesná definice viz níže). Jehož části jsou URL — Uniform Resource Location (sloužící pro hledání zdrojů) a URN (konečná identifikace zdroje) — Uniform Resource Name.

Část URI „scheme“ není totéž co protokol, může jít například o ”mailto”pro odesílání emailů.

Zdroj může být buď reálný např. konkrétní objekt (auto), nebo abstraktní (adresa).

Přesná definice URI

```
URI = scheme":"[ "//" authority ][ "/" path ][ "?" query ][ "#" frag ]
```

Typickým příkladem implementace REST je HTTP protokol, který budu uvažovat i nadále. Tento protokol je široce pokryt a je to jakýsi dnešní standard pro webovou komunikaci.

Při implementaci mají vývojáři dvě možnosti pro skladbu URI, příklad ve zmíněné přednášce uvádí:

- /customers/{customer_id}/orders
- /orders/{customer_id}

Je tedy třeba rozmyslet na základě zadání jakou „posloupnost“ zdrojů zvolit. Samotný zdroj může mít mnoho reprezentací známé jsou například XML, JSON, HTML. Reprezentace pak může být ještě v různém formátu — v podobě zdrojových dat, binární, textové, ...

4.1.1 Uniform interface

Jednotné rozhraní představuje operace tzv. CRUD pro HTTP jde o následující:

- C — CREATE — POST(nebezpečná), PUT (idempotence)
- R — READ — GET (idempotence)
- U — UPDATE — PUT, PATCH (bezpečná)
- D — DELETE — DELETE (idempotence)

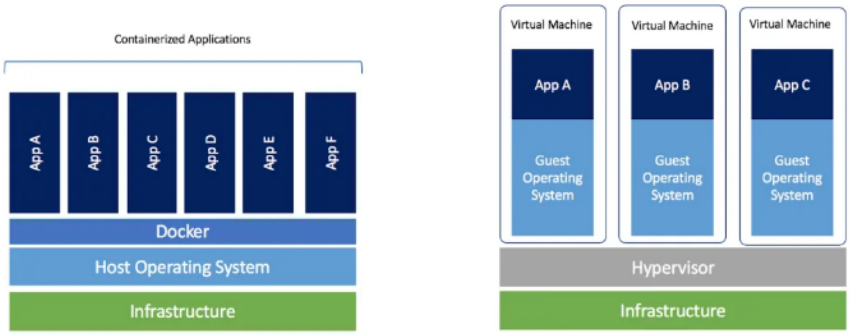
Každá z těchto operací může být pak bezpečná (neměnící stav zdroj) nebo nebezpečná. Další vlastnost je pak idempotence – to znamená, že více stejných požadavků skončí se stejným výsledkem. Bezpečné metody jsou implicitně bezpečné.

Dalšími typy HTTP dotazů jsou HEAD (získání hlavičky odpovědi) a OPTIONS (získání popisu toho, jak má vypadat komunikace). Tyto dotazy jsou oba bezpečné.

4.2 Technologie Docker

Technologie resp. nástroj Docker dnes slýcháme často. Jedná se o technologii, kde se vytváří tzv. kontejnery, které mohou připomínat virtualizaci. Rozdílů mezi kontejnerem a virtualizací je hned několik. Virtualizace ke svému běhu potřebuje vlastní operační systém a virtualizační software zjednodušeně řečeno emuluje veškerý hardware. Kontejnery neběží na vlastním operačním systému, sdílí prostředky hostujícího stroje ba dokonce sdílí společně jádro systému. Procesy docker kontejnerů jsou pak oddělené. Na technologiích nás může

zmást, že se zde často vyskytuje slovo „image“ neboli obraz. V případě virtualizace jde nejčastěji o obraz instalačního média operačního systému. V případě Dockeru jde spíš o předpřipravený „snapshot“ – příkladem může být Docker image pro PGAdmin (utilita pro administraci databáze Postgres) – v případě spuštění se vše stáhne a nastaví a po dokončení tohoto procesu je vše připraveno k použití. Vývojáři tak například nemusí řešit kompatibilitu různých balíčků. Rozdíly mezi virtualizací a kontejnerem nejlépe popisuje obrázek přímo ze stránek společnosti 4.1 na straně 27.



Obrázek 4.1: Rozdíl mezi kontejnery a virtualizací převzato z `https://www.docker.com/blog/containers-replacing-virtual-machines/`

K definici toho jaký image se má použít, co se má nastavit, spustit slouží tzv. Dockerfile. Jeho struktura značně připomíná obyčejný bash skript obohacený o klíčová slova. Zmíním zde některá klíčová slova, které jsou dle mého názoru nutné minimum pro jeho použití[35]:

- FROM {image} – vybrání image
- COPY {co} {kam} – překopírování dat z hostujícího OS do kontejneru
- ADD {co} {kam} – obdobné jako COPY, lze použít url nebo dekompresi
- ENTRYPOINT {co} – co se má spustit (například Shell)
- WORKDIR {cesta} – pracovní adresář
- RUN {cesta} – spuštění příkazu pro build např. instalování balíčků
- CMD {cesta} – spuštění příkazu v kontejneru

4.2.1 Docker compose

Docker compose je nástroj umožňující propojovat jednotlivé kontejnery mezi sebou. Ty jsou ve výchozím stavu odděleny, ostatně to již bylo popsané v odstavci výše. S definicí návazností a vztahů pomáhá právě tento nástroj. Obdobně jako pro jednotlivé kontejnery existují Dockerfile, tak pro tento nástroj specifikujeme soubor „docker-compose.yml“. Tento soubor definuje pomocí formátu Yaml jednotlivé kontejnery, které se mají pustit, jejich parametry a pořadí. Může jít o: Dockerfile k danému kontejneru, sítě ve kterých nachází, hostanames nebo závislost na jiném kontejneru. Uvedu níže krátký příklad a krátký výčet zajímavých klíčových slov s vysvětlením [35]. Příklad takového souboru:

```
services:
   cont1:
      build:
         context: .
         dockerfile: Dockerfile.db
      volumes:
         - ./media:/app/media

      ports:
         - "1234:1234"
      depends_on:
         - cont2
      networks:
         - default
      environment:
         - ENV_A=AAA
         - ENV_B=BBB

      cont2:
         image: image_name
         hostname: cont2
         command: bash
         ports:
            - "1234:4567"
```

Nyní pár vět k vysvětlení:

- image: {název image} – nastavení image ze kterého se kontejner tvoří
- build:
  - context: {cesta ke kontextu} – kontext ve kterém je chápán
  - dockerfile: {název/cesta k Dockerfile} – cesta k Dockerfile

- volumes: – mapování souborového systému mezi hostujícím OS a kontejnerem
- enviroment: – následuje výčet proměnných prostředí
- ports: – následuje mapování portů mezi kontejnerem a OS
- networks: – následuje výčet sítí
- depends_on: – výčet názvů kontejnerů na kterých je tento kontejner závislý

Základní příkazy pro docker a docker-compose jsou následující:

- docker build {cesta}— vytvoření image z Dockerfile
- docker run {image}— spuštění kontejneru z image
- docker exec {nazev kontejner} {prikaz}— spuštění příkazu v kontejneru
- docker-compose build — vytvoření sady kontejnerů v adresáři, kde je docker-compose.yml
- docker-compose up — spuštění sady kontejnerů v adresáři s kontejnery

4.3 ORM — Objektově relační mapování

ORM z angličtiny „Object relational mapping“, jedná se o mapování záznamů v databázi na objekty v příslušném programovacím jazyku. S těmito objekty je daný programovací jazyk schopný pracovat snadněji. Kdyby ORM nebylo použito, záznamy by se museli ručně zpracovat a z vytvořit z nich objekty definované programátorem ručně.

ORM často nabízí celé definování databáze skrze definice právě objektových modelů, k dispozici je pak konzolový nástroj, který by vygeneroval příslušné SQL, případně ho také rovnou pustil na databázovém stroji.

Při definování relací mezi jednotlivými entitami lze v případě ORM poměrně jednoduše definovat vztah obousměrně, což opět programátorům ulehčuje značně práci.

Mezi známé zástupce této technologie spadá např. Sequelize nebo Doctrine.

4.4 Node.js

Node.js je backend technologie založena na JavaScriptu. Technologie využívá asynchronní přístup zpracování a modelu událostí. Nejčastěji jsou v něm psané aplikace zpracovávající právě webové požadavky[36].

4.4.1 Express

Express je balíček pro Node.js usnadňující vývojáři tvorbu REST API. Jeho využití nejlépe lze ukázat na následující příkladu[37]:

```
1 const express = require('express')
2 const app = express()
3 const port = 3000
4
5 app.get('/test', (req, res) => res.send('Hello World!'))
6
7 app.listen(port, () => console.log(`Example app listening at
  ↪  http://localhost:${port}`))
```

Pokud bychom z internetového prohlížeče přišli na adresu „http://localhost:3000/test“ aplikace by vypsala „Hello world“. Analogicky to platí pro ostatní HTTP metody. Místo metody get na řádku 5 by se volala např. metoda post.

4.5 Shoptet

Shoptet je jedním z nejčastějších řešení pro internetové obchody v Čechách a na Slovensku. Uvádí až třetinové zastoupení ze všech e-shopů. Jedná se o kompletní řešení s vlastním cloudem a spousty analytickými nástroji. Zajímavostí je i vlastní pokladní systém, který umožňuje obchodům mající jak kamennou prodejnu, tak internetový obchod pohodlné propojení a správu na jednom místě.

Shoptet ve spojení se Zboží.cz je k dispozici i web — `https://www.ceska-ecommerce.cz/` — dokládající stav e-commerce v české republice. Nabízí zajímavá data jako oblíbenost druhů plateb, doprav nebo obrat e-commerce a počet e-shopů v ČR.

Kapitola 5

Co je to projekt Wowee?

Projekt Wowee byl dříve znám pod názvem ElateMe. Vznikl a je dále vyvíjen zde na fakultě FIT ČVUT pod vedením Michala Maněny, který projekt vlastní. Možnost podílet se je v rámci předmětů BI-SP1 a SP2, v rámci bakalářských a diplomových prací.

Projekt je využívá principu crowdfoundigu, který je celosvětově znám především díky platformě Kickstarter. Tento projekt nabízí lidem podporu zajímavých věcí a projektů a to tím způsobem, že každý může přispět nějakou částkou. O proti tomu tvůrce takové sbírky stanoví v aplikaci Wowee cílovou částku, kterou chce vybrat.

V případě Wowee je zde tedy podobný princip. Služba je založena sociálních kontaktech na Facebooku a umožňuje uživatelům vytváře sbírku na dárky nebo jiná přání. Přání může být vytvořeno pro stejného uživatele, který sbírku vytváří, nebo i pro někoho dalšího. V takovém případě může jít o překvapení, nebo o dárek, který by si daná osoba za normálních okolností sama nepořídila.

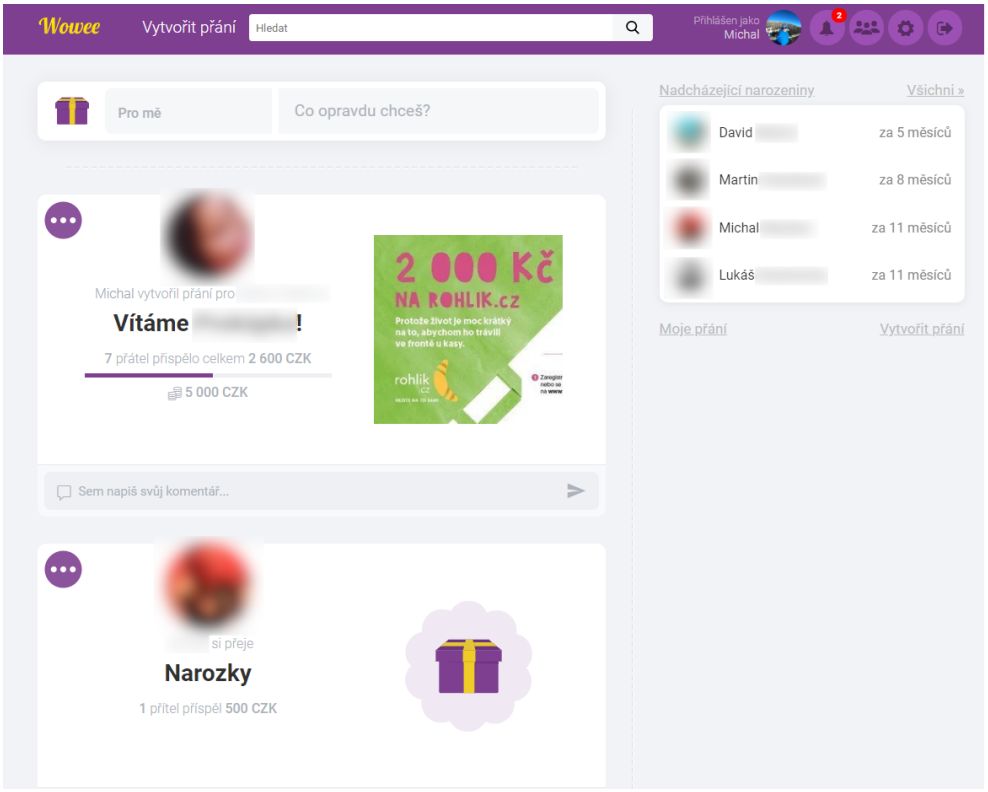
Moje předchozí práce na Wowee

Osobně jsem se s Wowee setkal v předmět BI-SP1 a BI-SI1, jedná se o předměty „Softwarový projekt“ a „Softwarové inženýrství“. Dále jsem v projektu pokračoval při vypracování bakalářské práce na téma GDPR.

5.1 Jak Wowee vypadá?

Front-endová část Wowee je založena na technologii ReactJS. Za touto technologií stojí společnost Facebook a je tedy snadné integrovat prvky této sociální sítě do aplikace. Aplikace samotná je poměrně podobná samotné sociální síti. Po vstupu zde vidíme tzv. dashboard na kterém jsou vidět interakce mezi kontakty. Informace o kontaktech, jako jsou nadcházející narozeniny, se zo-

brazují na panelu vpravo podobně jako kontakty na Facebooku — vše je vidět na obrázku 5.1 na straně 32.



Obrázek 5.1: Wowee dashboard (jména a fotografie jsou záměrně skryta)

5.2 Nakládání s osobními údaji

Osobní údaje a jejich zpracování je dnes citlivé téma a i v této práci se s nimi částečně setkal. V databázové modelu 7.1.2 na straně 38 se ale v důsledku neuchovávají jména ani konkrétní adresy. Snažil jsem se model samotný navrhnout co nejlépe tak, aby už od začátku obsahoval jen data relevantní k účelům této práce.

Samotné Shoptet API nabízí mnoho informací včetně konkrétních adres, tyto data však nejsou třeba a ani jsem se k nim nikdy přímo nedostal. Vývoj skriptu, který data migruje z Shoptet API do mé databáze 9 na straně 47, proběhl proti vzorovým datům platformy API Shoptet, viz o testování kapitola 8.1 na straně 44 a další navazující kapitoly. Skripty se dále testovaly proti testovacímu e-shopu Michala Maněny. Zisk konečného data setu přes reálné tokeny e-shopů proběhl pod dohledem taktéž Michala Maněny, aby se zaručila co

možná nejvyšší bezpečnost. Data sety tedy neobsahují žádné konkrétní informace jako jsou adresy ve smyslu ulic, telefonní kontakty, emaily ani konkrétní jména.

5.3 Partner API

Aplikace Wowee nabízí možnost internetovým obchodům využívající platformu Shoptet (více o Shoptetu v kapitole 4.5 na straně 30) nainstalování doplňku Wowee do jejich e-shopu. Tento doplněk umožňuje import jejich produktů do Wowee, které má za následek zobrazení produktů uživatelům, kteří dále mohou vytvořit sbírku na konkrétní produkt v konkrétním e-shopu. Toto funguje i z druhé strany. E-shop používající tento doplněk zde má tlačítko vytvořit sbírku na Wowee.

Právě pro funkci tohoto doplňku má Wowee Partner API a nyní je třeba jej rozšířit o další specifikované vlastnosti v zadání této práce.

Tato práce navrhuje rozšíření Partner API o předpovědi vhodného produktu k inzerci a ohodnocení vhodnosti produktu pro uživatele při vytváření sbírky.

Kapitola 6

Analýza dostupných dat z API Shoptet

Dokumentaci API platformy Shoptet lze nalézt na shoptet.docs.apiary.io. Samotné API pak na adrese api.myshoptet.com. Jedná se o standardní REST API, které vrací odpovědi ve formátu JSON. API podporuje stránkování, nebo vracení některých sekcí až na vyžádání. Endpointy pak nabízejí získání očekávaných seznamů, jako jsou seznamy produktů, uživatelů, objednávek a faktur. V této práci využiji jen některé z nich, které představím v následujících odstavcích.

Všechny endpointy tohoto API, jak je v REST zvykem fungují následovně: Chceme-li získat informace o nějakém uživateli nebo uživatelích vytvoříme HTTP GET požadavek na /customers, který navrátí seznam (neboli list) a teprve při požadavku na /customers/{guid} dostaneme detaily o konkrétním uživateli. Více o REST přímo v kapitole, která se REST věnuje — 4.1 na straně 25.

6.1 Zákazník

K získání informací o zákazníkovi slouží endpoint /customers resp. /customers/{guid}. Poskytnuty jsou následující informace: Standardní data jako jsou fakturační a adresa dodání. Tyto data jsou rozšířena o podrobnosti, jako je například okrsek nebo jméno regionu, dále zde můžeme najít kategorii zákazníka – e-shopy často mají kategorie pro velkoobchodní a maloobchodní odběratele, podobně je tomu u ceníků.

6.2 Objednávka

Pro získání informací o objednvkách je zde endpoint /orders resp. /orders/{code} pro získávání objednávek navrací informace kdo a kdy ji vytvořil, fakturační a dodací adresu, v detailu pak nalezneme informace o tom jaké produkty

obsahuje, kolik položka váží, nebo GPS souřadnice kam se má objednávka doručit.

Tento endpoint navrací „snapshot“ objednaných produktů — respektive jejich variant. Tato vlastnost je důležitá, neboť varianty produktů se v průběhu času mohou měnit – zákazníkovi je tak zaručeno, že dostane správnou variantu. Pro tuto práci je ale zajímavá úplně jiná vlastnost a to, že jednoznačné identifikátory, které detail objednávky poskytují nemusí směřovat na existující objekt. Nejjednodušší je ukázat to na příkladu: Zákazník si objednal A za cenu 150,-. Tento produkt se „otiskne“ do objednávky a v průběhu času jej obchod z nabídky stáhne. V objednávkách tento produkt zůstane, ale endpoint s produkty ho již nevrátí. Tato vlastnost vede k tomu, že ani v databázovém modelu nejsou vazby s cizími klíči mezi tabulkou Order a Item Variant viz kapitola o návrhu databáze 7.1.2 na straně 38.

6.3 Produkt

Pro informace o produktech má Shoptet API endpoint /products resp. /products/{guid}. Každý produkt má alespoň jednu tzv. variantu. Varianta má očekávané vlastnosti jako je cena, barva nebo EAN či ISBN. Každý produkt může být v několika kategoriích více v kapitole o návrhu databáze 7.1.2 na straně 38.

6.4 Kategorie

Kategorie jsou pro mou práci jeden z nejdůležitějších endpointů — jedná se o endpoint /categories. Platforma internetového obchodu Shoptet při vytváření kategorie v administraci e-shopu nabízí uživateli mimo vytvoření kategorie, kterou vidí uživatel v obchodě i další. Jedná se o kategorie pro katalogy — známými zástupci jsou například Heureka.cz nebo Zboží.cz. Tyto katalogy resp. jejich kategorie zaručují seskupení jednotlivých produktů (a jejich variant) do stejných kategorií.

Příkladem může být následující: e-shop mající kategorie „Bílá vína z Moravy“ a „Červená vína z Itálie“, tak katalogový název by mohl být jen „Vína“. Shlukování produktových variant napříč všemi e-shopy přes jejich jméno není dost dobře možné (zcela určitě se jeden produkt bude jmenovat různě — např. „iPhone 8“ vs. „iPhone xx GB“ vs. „iPhone 8 černý + dárek“ apod.) a ne vždy bude obsažen u produktu EAN. Tedy jednotné kategorie katalogů a srovnávačů jako je Heureka.cz, jehož služby využívá až 93 % e-shopů[3], je poměrně dobrá volba. Dalším užitečným katalogem může být Google nebo zmíněné Zboží.cz, jejichž kategorie má provozovatel e-shopu také možnost nastavit.

[3]K 11.2.2020 uvádí Heureka.cz 38 100 e-shopů a web ceska-ecommerce.cz uvádí 40 750 e-shopů celkově v ČR.

Kapitola 7

Návrh databázového modelu pro získaná data

Návrh databázového modelu je na obrázku 7.1 na straně 42. Tento model slouží k uložení dat z Shoptet API a dále je nad ním prováděna datová analýza, která je popsána v kapitole 10 na straně 55 této práce.

7.1 Entity pro ukládání dat

7.1.1 E-shop

Základní entitou je e-shop, od kterého se vše odvíjí. Tato entita má následující atributy

- id – identifikátor e-shopu např. alza.cz či jiné globální ID a zároveň **primární klíč**
- name – celý název e-shopu
- last_processed_date – datum posledního zpracování resp. poslední migrace dat z API

7.1.2 Customer

Entita customer je první z relací, kterou má entita e-shop. Každý e-shop může mít n customerů. Naopak konkrétní zákazník patří právě k jednomu e-shopu. U této entity jsem definoval jako zajímavé následující atributy:

- id – interní unikátní id v této databázi **a primární klíč**
- billing_city – město z fakturační adresy
- billing_district – okrsek z fakturační adresy

- billing_price_ratio – koeficient cen produktů pro uživatele [38]
- billing_region_name – název regionu z fakturační adresy
- billing_zip – směrovací číslo z fakturační adresy
- client_code – identifikátor klienta resp. zákazníka v rámci e-shopu [39]
- customer_group_id – id skupiny zákazníků uvnitř e-shopu (např. id pro skupiny student, velkoodběratel)
- customer_group_name – název samotné skupiny t.j. samotný text „student“, „velkoodběratel“ apod.
- guid – globální jednoznačný identifikátor klienta
- price_list_id – id ceníku, kam je zákazník zařazen
- price_list_name – název ceníku, kam je zákazník zařazen
- eshop_id – id e-shopu ke kterému zákazník patří

Omezení: Kombinace e-shop_id a guid musí být unikátní.

7.1.3 Item variant

Další relace, kterou entita e-shop disponuje je s entitou item_variant. Tato entita udržuje informace o produktech resp. jejich variantách. Každý produkt má alespoň jednu variantu. Varianta má svoji cenu, ean, barvu, ...V důsledku jsem vybral následující atributy:

- id – jednoznačné id v rámci této databáze **a primární klíč**
- price – cena nabízené varianty produktu
- guid – jednoznačný identifikátor produktové varianty
- item_ean – EAN kód
- code – kód varianty
- name – název varianty
- brand – název výrobce resp. značky
- parameter_color – barva produktu
- type – typ prodktu, může jít například o „product“ či „gift“, apod.
- eshop_id – id e-shopu ke kterému varianta patří
- manufacturerCode – kód výrobce
- isbn – kód ISBN

Omezení: Kombinace guid a e-shop_id musí být unikátní.

7.1.4 Order a Order Item Variant Pair

Další, řekl bych z hlavních entit, je entita order udržující následující informace o objednávkách daného e-shopu, pro každou objednávku jsou v databázi následující atributy:

- creationTime – datum vytvoření
- paid – informace zda-li je objednávka zaplacená
- status – v jakém stavu se nachází
- payment_method_name – jméno platební metody
- shipping_latitude – zeměpisná šířka, kam se má objednávka odeslat
- shipping_logitude – zeměpisná délka, kam se má objednávka odeslat
- code – číslo objednávky
- id – jednoznačný identifikátor v rámci této databáze a **primární klíč**
- last_processed_date – datum kdy byla zpracována
- customer_guid – guid zákazníka, který tuto objednávku vytvořil
- eshop_id – id e-shopu, ke kterému tato objednávka patří

Omezení: Kombinace code a e-shop_id musí být unikátní.

Zajímavostí mezi těmito čtyřmi entitami je, že jediné cizí klíče, které se zde vyskytují jsou je e-shop_id. To je dané tím, že v případě vytvoření objednávky se vytváří „snapshot“ produktů viz kapitola 6.2 na straně 36. Při migraci dat tedy nelze zaručit, že skutečně půjde o cizí klíč a nebo o neplatný identifikátor. Tato vlastnost návrhu je náchylná na případné nekonzistence, ale vzhledem k tomu, že tato databáze je určená jen a pouze pro účely této práce, ničemu to tedy dle mého názoru nevadí. Cizí klíč je zaručení provázanosti záznamů, nemá však vliv na spojování tabulek skrze SQL JOIN. Je tedy stále možné dostat všechna provázaná data.

Vzhledem k výše popsanému bych rád nyní popsal tabulku order_item_variant_pair provazující tabulky order a item_variant . Jedná se vlastně o dekompoziční tabulku vztahu N:N nenajdeme tedy zde nic mimo identifikátorů:

- id – interní id v databázi a zároveň **primární klíč**
- order_code – číslo objednávky
- item_variant_guid – guid varianty
- eshop_id – id e-shopu

7.1.5 Category a Item Category Pair

Tabulka category soustřeďuje informace o kategoriích, které existují v daných e-shopech a jejich názvech v katalozích. Uchované atributy:

- name – název kategorie, tak jak je na e-shopu, **část primárního klíče**
- name_heureka – název kategorie na heureka.cz
- name_heureka – název kategorie na zbozi.cz
- name_heureka – název kategorie na google.cz
- eshop_id – id e-shopu ze kterého je kategorie, **část primárního klíče**

Každá kategorie může mít N item_variant a každá item_variant může být obsažena v N kategoriích. Jde tedy opět o vztah N:N, kde se musí vyskytovat dekompoziční – item_category_pair tabulka, která obsahuje opět jen identifikátory:

- item_variant_id – id produktové varianty
- id – iterní id v této databázi a zároveň **primární klíč**
- category_name – název kategorie v rámci e-shopu
- category_e-shop_id – id e-shopu ze které je tato kategorie

7.2 Tabulky udržující informace o stavu migrace

Ve spodní části diagramu 7.1 na straně 42 se nachází pět tabulek sloužící k uchování informace o tom, v jakém stavu je migrace dat k danému e-shopu. Popis entity je v následujících odstavcích.

7.2.1 E-shop Migration Status

Tabulka e-shop_migration_status je triviální. Uchovává data o tom, jestli je zpracování e-shopu hotovo nebo nikoliv. Má následující atributy:

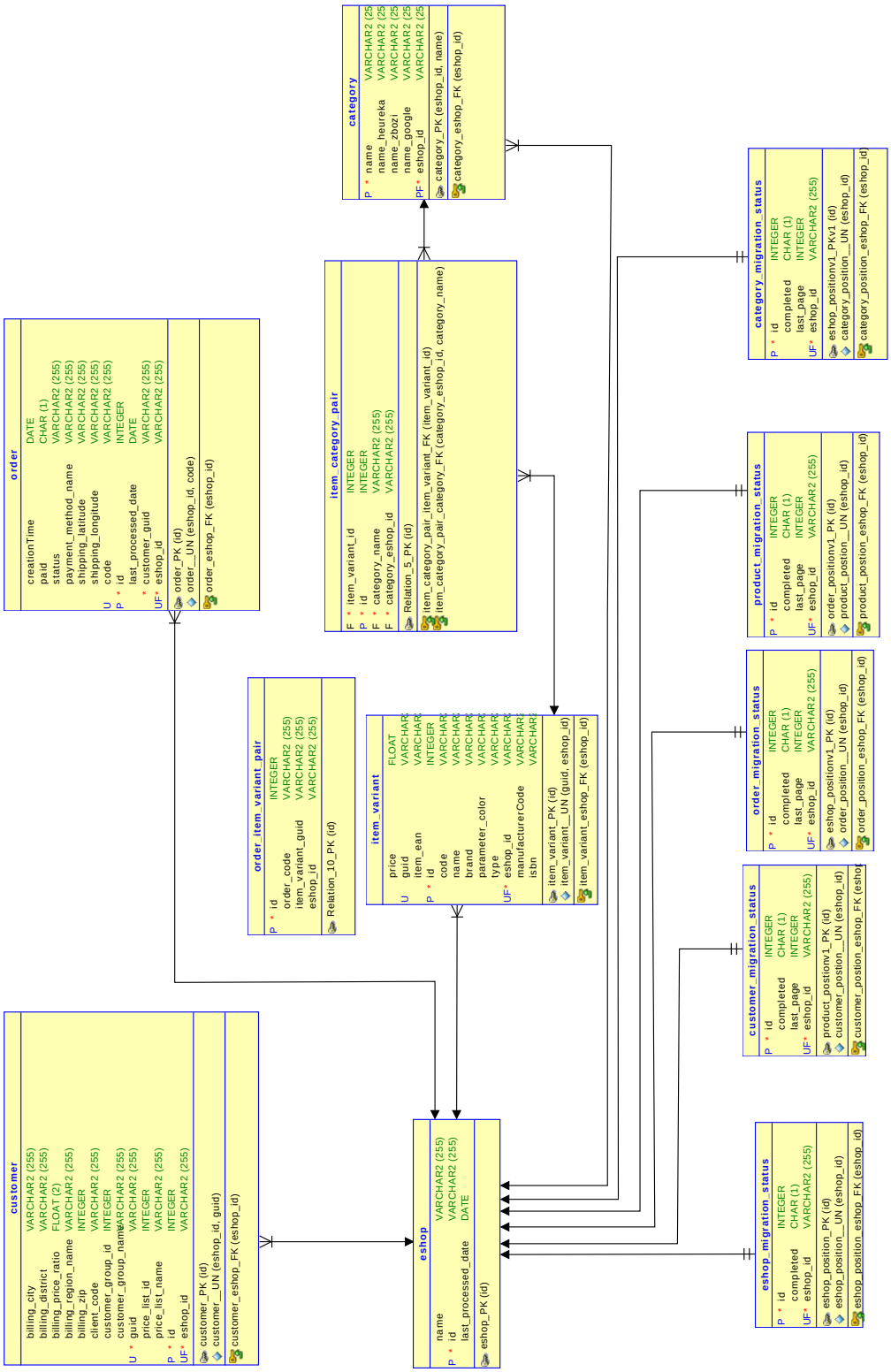
- id – iterní id v databázi a zároveň **primární klíč**
- completed – informace o tom, zda-li je hotovo
- eshop_id – id e-shopu ke kterému se vážou tyto informace

7.2.2 Ostatní Migration Status tabulky

Zbylé čtyři tabulky odpovídají čtyřem endpointům API, které jsou využívány (viz kapitola 6 na straně 35). a obsahují stejné informace jen o rozdílném endpointu. Popíšu tedy jen jednu z nich:

- id – id interní id v databázi a zároveň **primární klíč**
- completed – informace o tom, zda je migrace dat z endpointu dokončena
- last_page – stránka, kterou se skončilo a bude začínat při příštím běhu migračního skriptu
- eshop_id – id e-shopu ke kterému se tyto informace vztahují

Omezení: Pro všechny tyto tabulky platí, že je zde unikátní e-shop_id, tedy e-shop zde může mít pouze jeden záznam.



Obrázek 7.1: Entitně relační databázový model pro získaná data

Kapitola 8

Použité technologie

Použité technologie lze rozdělit do několika kategorií. Celé řešení Wowee Partner API je pak utvořeno několika containery platformy Docker. Má práce je do containeru převedena samozřejmě také. Přímo tato technologie je přiblížena v kapitole 4.2 na straně 26. Samotné kontejnery resp. práci v nich lze rozdělit do tří kategorií. První z nich jsou technologie, které jsou použity pro migraci dat z API Shoptet do mnou navržené databáze viz kapitola 7.1.2 na straně 38. Druhá kategorie je pak analýza dat a jejich transformace. Poslední je pak samotné REST API pro predikci a přístup pro BI systémy.

Celý „ekosystém“ se vytvoří spuštěním následujícího v adresáři s Wowee Partner API:

```
docker-compose build
```

A následně se spustí pomocí:

```
docker-compose up
```

Kontejnery, které obsahuje celé Parnter API (za předpokladu, že jiné na OS neběží) se získají pomocí:

```
docker container ls
```

Výsledkem jsou 4 kontejnery (výpis je zestručněn):

```
CONTAINER ID        IMAGE                  STATUS          NAMES
f988035016fc        partner-api_papi    Up 3 seconds   partner-api_papi_1
e67290ca3ccc        partner-api_python  Up 4 seconds   partner-api_python_1
ae7f4949b160        partner-api_psql    Up 5 seconds   partner-api_psql_1
a1c14f9c1d22        dpage/pgadmin4      Up 5 seconds   partner-api_pgadmin_1
```

Kontejnery jsou celkem čtyři. První je pro Node.js, na kterém běží celé Wowee Partner API. Dále je zde kontejner s Python 3, pro práci nad získanými

daty z Shoptet API. Třetím je databáze PostgreSQL a poslední je aplikace PgAdmin, která slouží jako webové administrační prostředí pro databáze PostgreSQL.

8.1 Migrace dat

V případě migrace dat z Shoptet API bylo třeba vsadit kód do již existujícího projektu tzv. Wowee partner API. Tento projekt využívá databázový systém PostgreSQL a pro backend technologii Node.js (jde tedy o variantu JavaScriptu). Přístupové body jsou poté realizovány pomocí REST API více o REST API kapitola 4.1 na straně 25. Pro snadnější vytváření dat v databázi bylo zvoleno využití ORM Sequelize. Je nutné podotknout, že pro naše účely se v tomto případě nepoužívá jeho plná síla, ale pomáhá nám při validaci vkládaných dat a tvorbě SQL dotazů. Dotazy v tomto ORM (viz kapitola 4.3 na straně 29) připomínají dotazy v MongoDB, které svou strukturou připomínají formát JSON. Vzhledem k tomu, že skript pracuje s daty třetí strany a není vhodné při vývoji dotazovat pokaždé vzdálené API, jsou zde vytvořeny testy ve frameworku Mocha. Pro simulaci HTTP responses je pak použit balíček mockserver, který poměrně jednoduše tuto funkci plní. Více o tzv. mockování viz kapitola 3.4 na straně 22.

8.2 Analýza dat a jejich transformace

Na analýzu dat a jejich transformaci pro další použití jsem vybral jazyk Python. Ten jen dnes již tradičním nástrojem pro datovou analýzu. Vede k tomu zejména kvalitní dokumentace vědeckých balíčků, které jsou k dispozici. Práce je tedy poměrně příjemná.

Jedním z hlavním zástupců těchto balíčku je balíček pandas, který ulehčuje práci s daty. Práce s ním je podobná jako s databázemi respektive jazykem SQL. Pandas umožňuje různé agregace jako je sčítání, průměry, dále také spojování tzv. dataframů, které představují objekty obsahující data sety.

Dalším zástupcem je matplotlib, který slouží pro vykreslování grafů. Objevuje se v různých modifikacích, ale pokud jsou v Pythonu generovány nějaké grafy většinou vždy v důsledku stojí právě na tomto balíčku.

Co se týká komunikace s databází (v mém případě konkrétně PostgreSQL) je nasnadě použít balíčky dva. Jedním je balíček psycopg2, pomocí kterého lze standardně získávat data pomocí SQL dotazů. Druhým je pak balíček sqlalchemy, kterého lze využít při spojení s balíčkem pandas. Díky tomu lze snadno převést DataFrame do databáze jako entitu — jedná se pouze o jeden příkaz.

Pro čtení parametrů skriptu z příkazové řádky je použit balíček click, který definuje parametry snadno pomocí anotací.

8.3 Technologie pro výstupní endpointy

Technologie pro předpověď a ohodnocení je vázána na Wowee Partner API vzhledem k tomu, že jde o jeho část. Partner API pro tvorbu REST (o REST kapitola 4.1 na straně 25) endpointů používá balíček express popsaný v kapitole 4.4.1 na straně 30.

Na REST API je postaveno dotazování na předpoveď reklamy, která se má uživateli zobrazit, ohodnocení sbírek, přidávání statistik i listování ve zpracovaných datech pro business inteligence systémy - viz kapitola 11.5 na straně 92. HTTP dotazy, které se týkají pro predikce a přidávání statistik jsou vhodné pro použití ve výchozí aplikaci Wowee (resp. na front-endové části) v podobě tzv. „AJAX“ požadavků.

AJAX je zkratka pro „Asynchronous JavaScript and XML“. Jde tedy o případ, kdy AJAX požadavek není nic jiného, než vyslání a zpracování asynchronního HTTP požadavku na pozadí aplikace. Taková data pak lze zobrazit bez nutnosti znovu načítat celý obsah stránky. Jméno je poněkud zavádějící, protože nemusí jít nutně o XML, ale nýbrž také JSON, nebo i jiný formát [40].

Kapitola 9

Skript pro migraci dat

Před prvním spuštěním by bylo za normálního stavu třeba nainstalovat zmíněné závislosti v kapitole 8.1 na straně 44. To lze provést příkazem (za předpokladu, že je nainstalován v OS Node.js):

```
npm install
```

V mém případě se závislosti však nainstalují sami (jak pro Node.js, tak pro Python) při zavolání příkazu (viz minulá kapitola):

```
docker-compose build
```

9.1 Testy

Vzhledem k tomu, že se používá API třetí strany (Shoptet) a nebylo vhodné ho neustále dotazovat při vývoji, implementoval jsem řadu testů. Pro testování je využit balíček Mocha. Testy lze nalézt v podadresáři „test“. Tyto testy lze rozdělit do dvou kategorií:

- testy testující základní funkce ORM modelů
- testy logiky migrace

Testy lze hromadně spustit příkazem v adresáři modulu (/src/modules/PredictionDataDownloader):

```
npm test
```

V prvním případě se pro každý model testuje přidání jednoho záznamu, přidání více záznamů a ověření omezení unikátnosti atributů (vyjma tabulky pro párování produktu a kategorie, kde se testuje N:N záznamů rovnou). Výjimkou je tabulka pro propojení objednávek a produktů. Zde vztahy nejsou zaručeny v API, neexistuje tedy relace v databázi a nemá smysl testovat správnost

(důvody viz 6.2 na straně 36). Záznamy v databázi se před a po každém testu mažou, aby se zajistila správnost výsledků resp. unikátnost stavu.

V druhém případě se testuje logika samotného přidávání záznamů do databáze. Pro každý endpoint z API Shoptetu je napsán vlastní test. Databáze se zde jako v předchozím případě **nemaže**, protože mezi získanými záznamy jsou již vztahy. Zmíněné simulování HTTP requestů na vzdálené API je realizováno pomocí tzv. mockování (viz kapitola 3.4 na straně 22). Mocky jednotlivých HTTP odpovědí lze nalézt v adresáři „mocks“, který má následující strukturu:

```
mocks
├── products
│   ├── GET.mock
│   ├── 3c5012051-d7ca-11e0-9a5c-feab5ed617ed
│   │   └── GET.mock
│   ├── 2c5012051-d7ca-11e0-9a5c-feab5ed617ed
│   │   └── GET.mock
│   └── 1c5012051-d7ca-11e0-9a5c-feab5ed617ed
│       └── GET.mock
├── categories
│   └── GET.mock
├── orders
│   ├── GET.mock
│   ├── 12017000092
│   │   └── GET.mock
│   └── 22017000092
│       └── GET.mock
└── customers
    ├── GET.mock
    ├── 3c9e976a5-340b-11e4-b500-ac162d8a2454
    │   └── GET.mock
    ├── 1c9e976a5-340b-11e4-b500-ac162d8a2454
    │   └── GET.mock
    └── 2c9e976a5-340b-11e4-b500-ac162d8a2454
        └── GET.mock
```

Adresa je pak vždy http://localhost:port/název_adresáře/ (dále standardně pro REST) samotný soubor GET.mock obsahuje samotná data HTTP odpovědi, prvních n řádků jsou hlavičky, následuje prázdný řádek a samotná data.

9.2 Spuštění

Samotný skript pro migraci lze spustit následujícím příkazem :

```
docker exec -it partner-api_papi_1 node
↪  /app/src/cron/predictionDataDownload.js
```

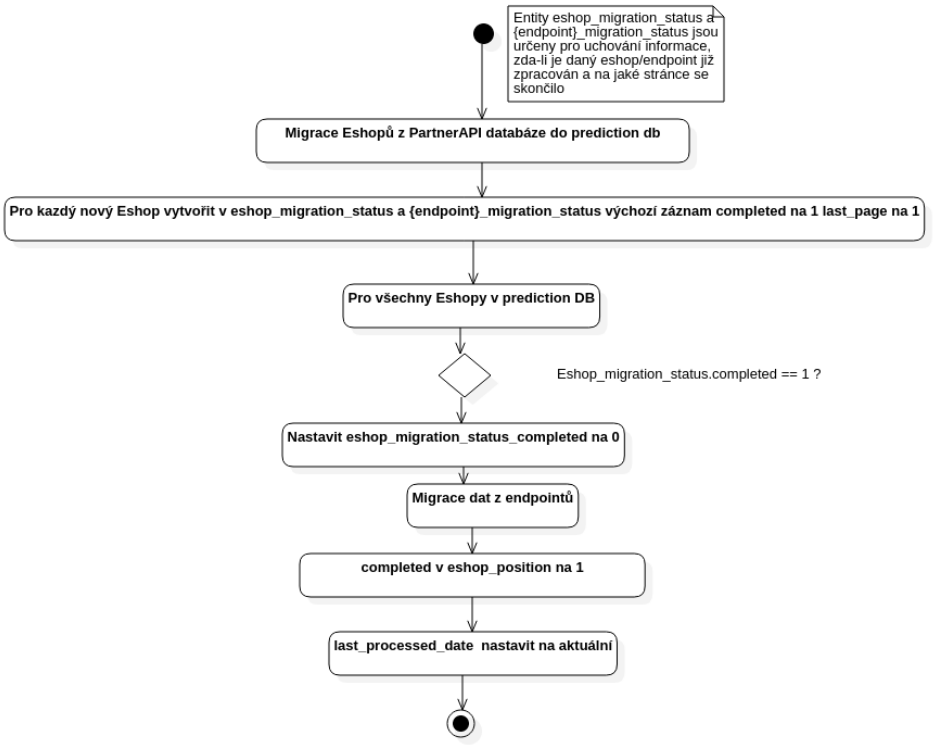
Skript předpokládá nastavené enviromentální proměnné pro připojení k databázi (kapitola 11.1 na straně 84). Jedná se o standardní: Název databáze, jméno a heslo, host, port a případně typ databáze (v našem případě jde o postgres, díky ORM lze využít i jinou). Konkrétně jde o:

```
DB_PREDICTON_NAME = "nazev DB"
DB_USERNAME = "usr"
DB_PASSWORD = "pass"
DB_HOST = "host"
DB_PORT = "port"
DB_TYPE = "postgres"
```

9.3 Flow procesu

Na diagramu 9.1 na straně 50 je popsána flow, jak migrace probíhá.

Nejprve se získají nové e-shopy z Partner API databáze, které ještě v predikční databázi neexistují. Do tabulek (viz kapitola 7.1.2 na straně 38), které udržují informace o stavu zpracování se nastaví výchozí údaje a pokud e-shop není zrovna zpracováván – teoreticky se to může stát – tak se pro něj spustí migrace dat. Samotnou migraci dat popisuje diagram 9.2 na straně 54.



Obrázek 9.1: Hlavní proces migračního skriptu mezi API Shoptet a predikční databází

9.3.1 Počáteční skript

Skripty pro migraci dat jsou napsány v JavaScriptu resp. v Node.js. Více o technologiích v předchozí kapitole 8 na straně 43. Vzhledem k tomu, že je nutné funkcionalitu zasadit do projektu, který již existuje, uvádím zde skript, který se spouští jako první a tuto funkcionalitu obstarává. Tento skript je specifický, protože využívá některé funkcionality a principy, které již v projektu jsou. Tyto některé funkcionality a principy má na svědomí Jan Šafařík, který je vytvořil v rámci předmětu BI-SP. Jedná se o skript **/cron/predictionDataDownload.js**. Začněme popisem funkce *initDownload*. Jako první se vytvoří zámek na soubor, aby nebylo možné skript pustit dvakrát paralelně. Jako další se vyberou všechny přístupové tokeny, které jsme získali do Partner API skrze wish-widget a zároveň jsou aktivní.

```
async function initDownload() {
try {
   await utils.setLockOn();
   logger.log('Creating lock')
```

```
} catch (err) {
   console.log(err)
   logger.logWarn('Lock already exists. Exiting ...');
   process.exit(0)
}

const eshopsList = await ShoptetAccessToken.findAll({
   where: {
      addon_name: 'wish-widget',
      active: true
      }
});
```

Jako další je třeba vyfiltrovat získané záznamy, které nemají žádný autorizační token resp. mají ho nastavený na null. V dalším kroku volám *await PredictionDataDownloader.init(PartnerApiEshopModel)*. Tato metoda migruje data z tabulky e-shops z databáze Partner API do databáze predikční viz diagram 9.1 na straně 50.

```
const filteredEshops = eshopsList.filter((item) => item.auth_token !== null);

await PredictionDataDownloader.init(PartnerApiEshopModel);
```

Nyní již může skript iterovat skrze všechny tokeny, které máme. Práci nám v tomto případě ulehčí soubor funkcí **utils.js** za který opět vděčím Janu Šafaříkovi. V každé iteraci si získám informaci o stavu migrace pro daný e-shop a objekt samotného e-shopu. Poté se volá funkce z utils.js *checkOrUpdateToken(accesToken)*, která nám zjistí zda je token validní v opačeném případě obstará token nový. Shoptet API používá OAuth2 [41].

```
await utils.asyncForEach(filteredEshops, async (accessToken) => {
   const eshopId = accessToken.eshop_id;

   var actuale-shopPosition = await eshopMigrationStatus.findOne({
      where: {
         eshop_id: eshopId
      }
});

var actualeshop = await eshop.findOne({
   where: {
      id: eshopId
   }
});
```

```
if (!eshopId) {
   logger.log('Missing e-shop id, please fix the records')
   return
}
   let token = null;
try {
   token = await utils.checkOrUpdateToken(accessToken)
} catch (err) {
   logger.log('Invalid token, continuing')
   logger.log(err)
   return
}
```

Nyní již lze začít se samotným migrováním dat. Nejdříve je třeba si ověřit, že pro daný e-shop neprobíhá žádná jiná migrace. Pokud ne, nastavím, že ji provádím. Dále volám funkci *PredictionDataDownloader.getDataForPrediction(eshopId, token)*. Tato funkce již volá mnou implementovaný modul, který se postará o přenesení dat ze Shoptet API do mnou navržené databáze. Pokud se proces dokončil opět nastavíme, že jsme skončili (řádky, kde se completed nastaví na 1 a nastaví se datum zpracování).

```
try {
   if (actualeshopPosition.completed === "1") {
      actualeshopPosition.completed = "0";
      actualeshopPosition.save();
      await
      ↪  PredictionDataDownloader.getDataForPrediction(eshopId,
      ↪  token);
      actualeshopPosition.completed = "1";
      actualeshopPosition.save();
      actualeshop.last_processed_date = new Date();
      actualeshop.save();
   }
} catch (err) {
   logger.logErr(`Unable to download products of e-shop
   ↪  "${eshopId}" due to error: ${err}`)
}
});
```

Na konci zrušíme zámek na soubor a zavoláme celou výše popsanou funkci.

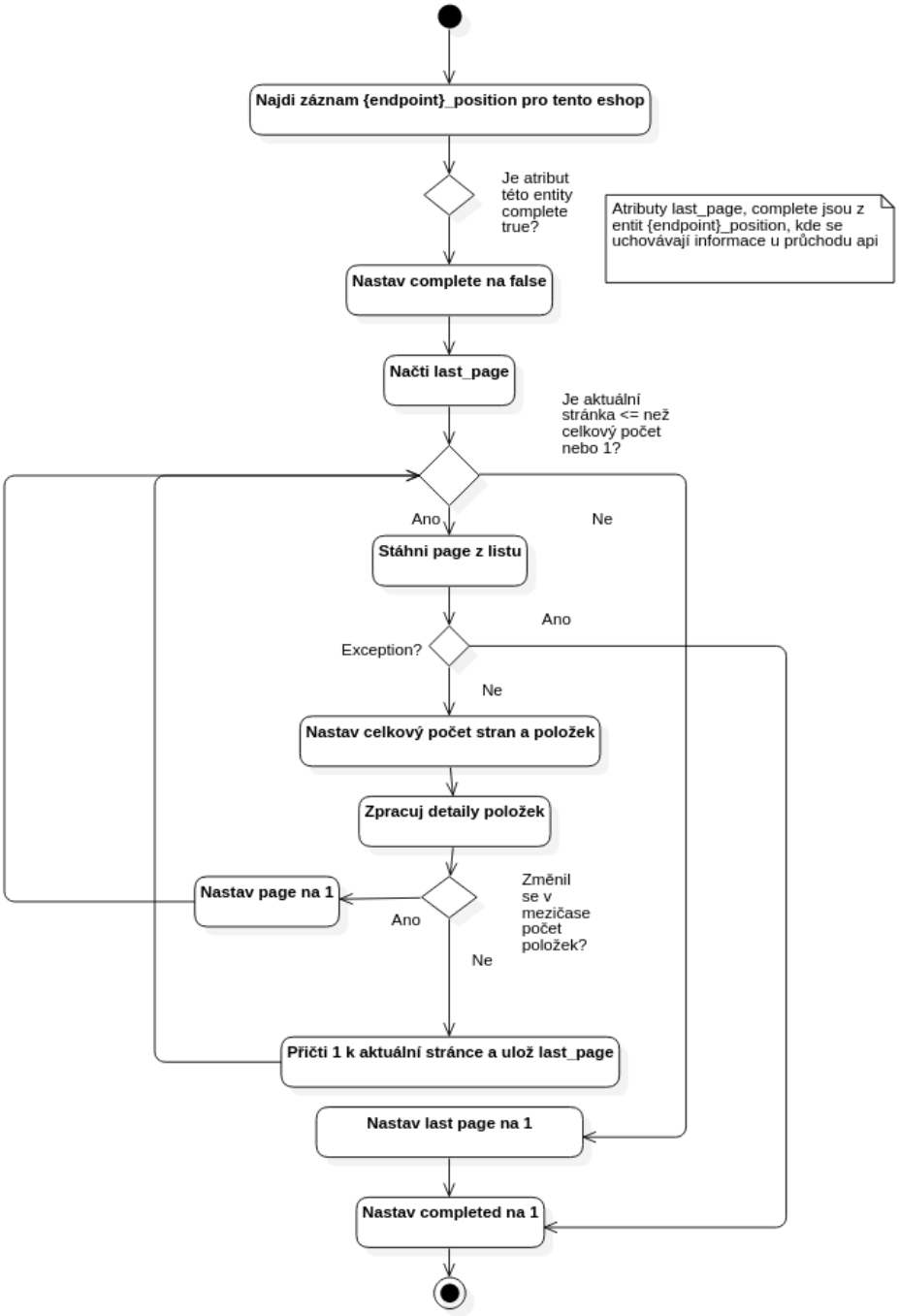
```
  try {
    await utils.setLockOff();
    logger.log('Deleting lock');
    process.exit(0)
```

```
  } catch (err) {
    logger.logWarn('Lock could not be deleted');
    logger.logErr(err);
    process.exit(1);
  }
}

logger.log("Script started");
initDownload()
```

9.3.2 Migrace dat z jednotlivých endpointů

Tato část flow je již o něco složitější. Pro každý z endpointů existuje tabulka, kde je uložen jeho aktuální stav – nejlépe lze vidět v kapitole 7.1.2 na straně 38, která popisuje návrh databáze. Pokud je stav completed, znamená to, že můžeme endpoint zpracovat (žádné jiné zpracování na něm neběží), načteme tedy poslední stránku, kde skript skončil (v případě, že skript skončí s exception máme uloženou poslední stránku). Poté co máme načtenou stránku proiterujeme všechny položky, které nám endpoint navrátil. Pokud se nám podařilo stáhnout první stránku, nastavím si celkový počet stran a položek. Pro každou z položek na stránce si najdeme detail (to neplatí pro endpoint kategorie, žádný pro ně není třeba). Po zpracování si uložíme poslední stránku, kterou jsme navštívili. Pokud se v mezičase změnil počet stránek, proiterujeme stránky od začátku. V případě dokončení nastavíme atribut completed na true. Detail a jeho případně přidružené vztahy přidáváme pouze pokud už entita neexistuje – toto je opět nejlépe vidět na návrhu databáze a zmíněných omezení. Vzhledem ke složitosti a analogii je tento algoritmus znázorněn pomocí UML diagramu 9.2 na straně 54.



Obrázek 9.2: Zpracování konkrétního API endpointu

Kapitola 10

Analýza získaných dat

Na analýzu získaných dat lze pohlížet dvěma způsoby. Jako první je analýza toho, jaké atributy entit jsou v praxi skutečně vyplňovány. Druhým je pak analýza významu samotných dat. Postupně v této kapitole rozeberu obě varianty. Po spuštění migračního skriptu jsem získal do databáze cca 42MB dat [4]. Bližší informace lze získat spuštěním skriptu dpTool.py. Resp. spuštění následujícího za předpokladu, že byl zavolán příkaz „docker-compose up“ a kontejnery jsou tedy spuštěny:

```
docker exec partner-api_python_1 python3 dpTool.py --analysis
```

Tento skript vygeneruje veškeré grafy poskytující informace o vyplněných atributech a i grafy k datové analýze.

10.1 Objem dat a jejich atributy

Konkrétní počty záznamů lze nalézt v tabulce 10.1 na straně 56. Záměrně zde nejsou uvedeny záznamy z tabulek udržující pozice jednotlivých endpointů, neboť tato informace není důležitá — jde o analýzu toho, jak reálně provozovatelé e-shopů vyplňují atributy. Počet záznamů v tabulce e-shop zde nutně neznamená, že tolik obchodů využívá Wowee Partner API. Jde zde o e-shopy, které jej použily třeba v i minulosti, v tabulce jsou uložené úplně všechny možnosti, které se získaly z Partner API databáze, viz diagram 9.1 na straně 50.

10.1.1 Data obsažená v tabulce e-shop

V tabulce e-shop je celkem 148 záznamů. **Z celkového počtu e-shopů je 16 slovenských, které dále v analýzách neuvažuji — jsou zde ceny v eurech a odlišné názvy kategorií v katalozích, které používám**

[4] Analýza dat k 3.2.2020

Název tabulky	Počet záznamů
category	6 489
customer	24 718
e-shop	148
item_category_pair	198 060
item_variant	54 543
order	42 367
order_item_variant_pair	167 604

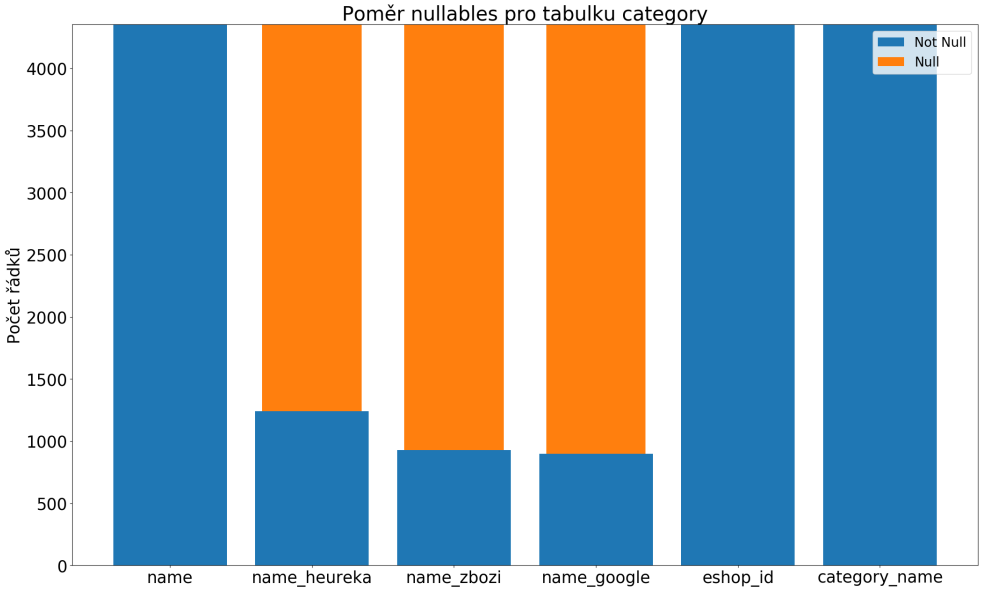
Tabulka 10.1: Objem získaných dat

dále pro sjednocení. Na tento fakt jsem přišel v průběhu analýzy a proto se i v preprocessingu pracuje s některými slovenskými výrazy. Z celkového počtu přístup k jejich API má Partner API ke 48 z nich (32 % z celkového počtu).

10.1.2 Data obsažená v tabulce category

V tabulce je obsaženo 6 489 záznamů z toho 25 % má vyplněnou kategorii pro srovnávač Heureka.cz, 14 % pro zbozi.cz a skoro 29 % pro Google — více viz graf 10.1 na straně 57. Z pohledu jednotlivých e-shopů pak Heureku používá 31 z nich, 25 z nich používá zbozi.cz a 28 Google. Vzhledem k tomu, že z jedním z cílů je předpovídat kategorii, která je vhodná pro inzerci, tak zde mám na výběr jaký zdroj informací zvolit. Buď můžu vybrat ten zdroj, kde je nejvíce záznamů (tedy Google), ale v takovém případě můžeme dostat podobná resp. málo rozmanitá data. Druhý přístup je zvolit srovnávač Heureka.cz, protože ho využívá více e-shopů a je zde větší šance na větší různorodost dat (nepůjde například jen o e-shopy s cyklistickým příslušenstvím).

Pro další práci volím cestu využití srovnávače Heureka.cz.



Obrázek 10.1: Vyplněné atributy v tabulce category

10.1.3 Data obsažená v tabulce customer

Tabulka customer je jedna ze složitějších. Má celkem 13 atributů. Poměr získaných dat nejlépe zobrazuje graf 10.2 na straně 58. Z tohoto grafu lze vyčíst, že nemá smysl k dalšímu zpracování používat atributy billing_district, billing_region a o client_code to lze tvrdit také, jednak není více jak z poloviny vyplněn(47 %) a k další identifikaci lze využít jiné atributy této entity.

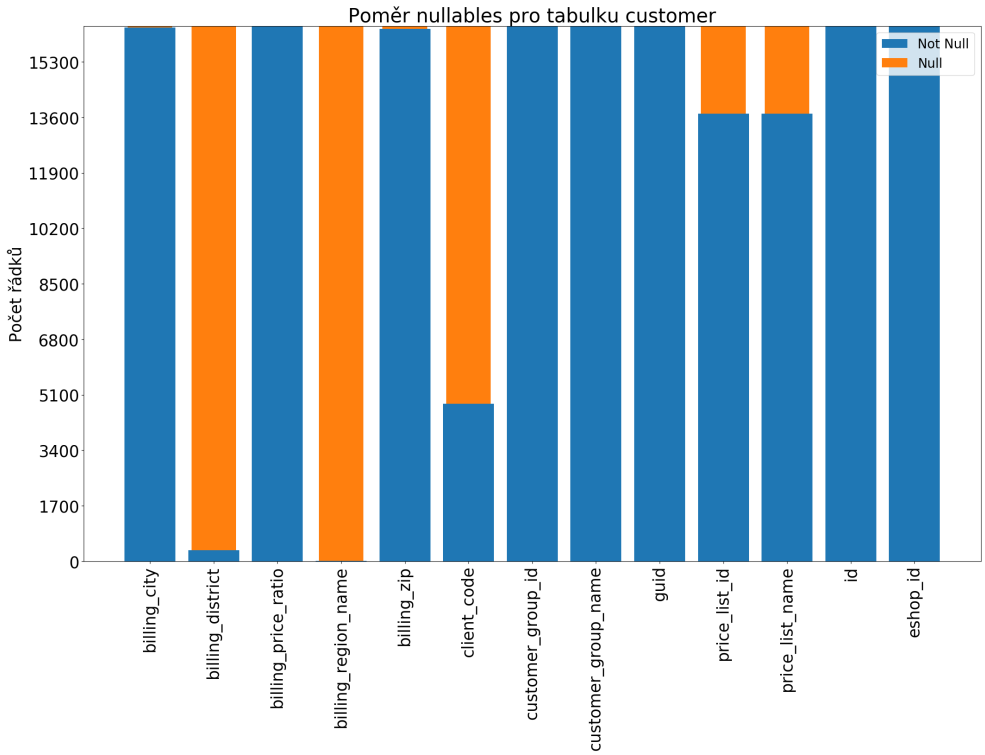
10.1.4 Data obsažená v tabulkách item

item_variant

Tato tabulka obsahuje informace o konkrétních produktech resp. jejich variantách. Z pohledu obsažených dat lze usoudit, že atributy ISBN, manufacturercode a především parameter_code jsou e-shopy zcela nevyužité. Ostatní atributy jsou na tom podstatně lépe – EAN má vyplněné 60 % záznamů, atribut brand 94 % a ostatní parametry jsou na tom už jen lépe – viz obrázek 10.3 na straně 59.

item_category_pair

Tato tabulka slouží jako dekompoziční viz 7.1.2 na straně 38. Jsou zde uložena data pro párování produktu a konkrétní kategorie. Z tohoto důvodu obsahuje společně s tabulkou order_item_variant_pair nejvíce záznamů - 198 060. Zde



Obrázek 10.2: Vyplněné atributy v tabulce customer

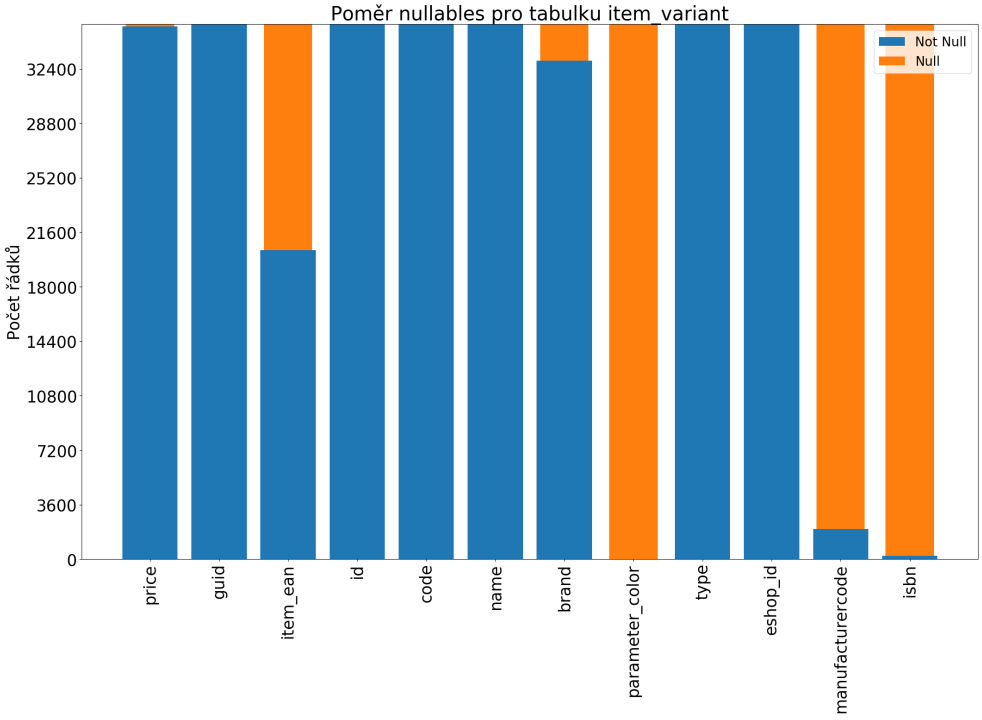
jsou všechny atributy vyplněny, neb pokud má produkt kategorii, tak kategorie určitě existuje.

10.1.5 Data obsažená v tabulkách order

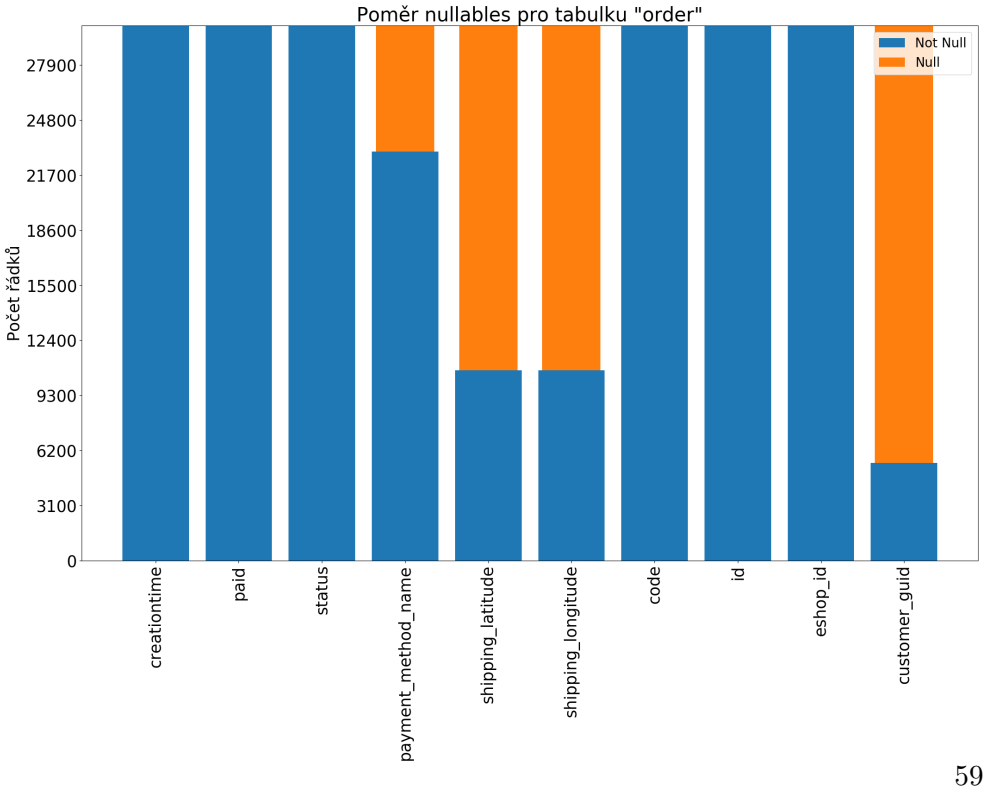
Podobně jako v předchozí kapitole i zde jsou dvě tabulky. Jedna z nich má obsažena data, druhá slouží jako dekompoziční.

order

Vedle uživatele a produktové varianty je to další z obsáhlejších tabulek, znázornění na grafu 10.4 na straně 59. Atributy shipping_latitude a shipping_longitude jsou vyplněny jen ve 28 %, dle mého uvážení lze tyto atributy odstranit. Zajímavé je, že z dostupných e-shopů a jejich objednávek má pouze 18 % vyplněn identifikátor uživatele. Tedy z celkového počtu 42 367 objednávek, lze párovat s uživatelem jen něco kolem 7 600 za předpokladu, že uživatel existuje a nejde o případ podobný jako v kapitole 6.2 na straně 36. Zajímavý je zde atribut creationtime, neb dle něj lze sledovat jak se skladba objednávek mění v čase.



Obrázek 10.3: Vyplněné atributy v tabulce item_variant



Obrázek 10.4: Vyplněné atributy v tabulce order

order_item_variant_pair

Tato tabulka obsahuje taktéž hodně záznamů – 167 604, pro tuto práci jsou nicméně však důležité jen záznamy, které mají vyplněné item_variant_guid a ani to není záruka, že záznam půjde propojit s konkrétními produkty viz opět 6.2 na straně 36. Relevantních záznamů je skoro 53 %, ale číslo bude vzhledem ke skutečnosti výše ještě menší.

10.2 Preprocessing dat k další analýze a zpracování

Tabulky obsahující záznamy zakládající se na uživatelském vstupu od provozovatele e-shopu je třeba zpracovat a „unifikovat“ před další prací s nimi. Jedná se o data jako jsou například kategorie zákazníků, které každý e-shop má víceméně stejné, ale liší se v podobě zápisu příkladem může být VIP vs. V.I.P. Tato zpravování dále rozeberu po jednotlivých entitách.

10.2.1 Pre-processing entity customer

V tabulce customer se nachází hned několik zajímavých atributů, před dalším zpracováním je třeba, aby byly sjednocené a jejich duplikáty odstranit.

Jako první se odstraní atributy billing_district, billing_region a o client_code, které e-shopy nepoužívají, viz předchozí kapitola.

Dalším z případů, které je potřeba zpracovat je dvojce customer_group_id a customer_group_name. Z těchto dvou si ponechám pouze customer_group_name. Je zde několik různých kategorií: základní kategorie je **„Koncový zákazník“**. Další jsou různé varianty na kategorii **„VIP zákazník“** nebo nějaký slovenský výraz, **„Velkoobchodní odběratel“** – zde platí totéž, co v předchozím případě a poté jsou zde kategorie, které de facto jsou taktéž VIP zákazníci jen jde o konkrétnější pojmenování. Následující metoda tyto kategorie sjednotí na čtyři výše zmíněné:

```
1 def unionCustomerCategory(category):
2     if "Velkoobchodní" in category or "Veľkoobchodný" in
      ↪  category:
3         return "Velkoobchodní odběratel"
4     elif "Koncový zákazník" not in category:
5         return "VIP"
6     elif "Koncový zákazník" in category:
7         return category
```

Tuto metodu pak volám aplikací metody apply, kterou poskytuje balíček pandas. Metoda vytvoří nový sloupec a daný kód se zavolá pro každý záznam. Metodě apply lze předat lambda funkci, nebo volat právě celou metodu tak, jak jsem toho využil já.

```
dataFrames["customer"]["customer_group_name"] =
dataFrames["customer"]["customer_group_name"].apply(unionCustomerCategory)
```

Další dvojicí atributů jsou price_list_id a price_list_name. Tyto atributy označují zařazení uživatele do ceníku. Z velké části jde o podobná data jako u předchozí dvojce atributů – dokonce jde občas o stejné pojmenování. Považuji je tedy za nadbytečné.

Dalším atributem je billing_price_ratio, které je v drtivé převaze rovno 1 (a tedy zákazník platí původní cenu). Opět volím cestu tento atribut vynechat, z pohledu predikce bych ho stejně nevyužil a z pohledu BI ho taktéž nepovažuji za extrémně významný.

Atribut billing_city je z pohledu doporučování zajímavý, protože díky němu lze určit místa, která tazatele zajímají. Problém tohoto atributu je, že jména měst nejsou vyplněna vždy ve stejném formátu a ani ve stejném znění. Někdy je uvedeno jen město, jindy město pomlka čtvrť a podobně. Takových hodnot je skutečně mnoho. Dlouho jsem přemýšlel, jak se s tím poprat a zároveň o tuto informaci nepřijít – řešení je přitom jednoduché. Stačí použít atribut **billing_zip**, což je v případě ČR poštovní směrovací číslo.

Pro poštovní směrovací číslo (dále už jen PSČ) platí následující pravidla: PSČ má vždy pět číslic. Jde o trojčíslí a dvojčíslí. První z čísel označuje kraj:

- 1 – Hlavní město Praha
- 2 – Středočeský kraj
- 3 – Jihočeský a Západočeský kraj
- 4 – Severočeský kraj
- 5 – Východočeský kraj
- 6 – Jihomoravský kraj
- 7 – Severomoravská kraj
- 8 – Bratislava a okolí
- 9 – Západoslovenský kraj a jižní část Středoslovenského kraje
- 0 – severní část Středoslovenského kraje a Východoslovenský kraj

Druhá z číslic pak označuje obec nebo městskou část obce. Třetí pak označuje okresní uzel a zbylá dvě jsou identifikátor pošt. Pro potřeby této práce využívám pouze první dvě číslice:

```
def unifyZip(zip):
    zip = str(zip)
    zip = zip.replace(' ', '')
```

```
    zip = zip.ljust(5, "0")
    zip = zip[:-3]
    if(len(zip)>=3):
        zip="00"
        return zip
dataFrames["customer"]["billing_zip"] =
dataFrames["customer"]["billing_zip"].apply(unifyZip)
```

Bohužel i data tohoto atributu po bližší analýze obsahují různé formáty – s mezerou, bez mezery či tvar „Město - PSČ“. Z tohoto důvodu bylo nutné formát sjednotit: Odstranit mezery, zaručit 5 míst (pokud je znaků méně, doplnit nulami) a nakonec nechat jen dvě pozice. Pokud má PSČ delší délku je to nějaký „patvar“, který dávám do speciální kategorie.

Rekapitulace

Krátká rekapitulace toho, co v tabulce customer zbylo — jde o pět atributů:

- billing_zip – první dvě číslice z PSČ
- billing_city – město z fakturační adresy
- customer_group_name – zařazení uživatele do patřičné skupiny
- guid - identifikátor uživatele v e-shopu (bude se hodit pro join dataframů)
- eshop_id – id e-shopu odkud záznam pochází

10.2.2 Pre-processing dat entity order

Pro další práci nebudou využity ani následující atributy, jejich vyplněnost je o proti celkovému objemu dat malá a v případě názvu platební metody se ani nedá nikterak využít, neb se zde vyskytují hodnoty zahashované:

- id
- payment_method_name
- shipping_longitude
- shipping_latitude

Naopak hodnoty atributu **status** je nutné upravit resp. sjednotit. Různé obchody používají různé stavy objednávek. Tyto stavy sjednocuji do pěti kategorií:

- expedováno
- čeká

- nevyřízeno
- storno/vratka
- hotovo

Pro představu jaká hodnoty se v atributu se vyskytují uvedu celou metodu:

```
def uniqueOrderStatus(status):
    status = status.lower()
    if "exped" in status or "odes" in status or "odos" in status
    ↪  or "výzva" in status:
          return "exped"
    if "čeká" in status or "čaká" in status or "vybavuje" in
    ↪  status or "uhradená" in status or "zaplacená" in status
    ↪  or "zaplaceno" in status:
       return "čeká"
    if "nevybavená" in status or "nevyřízená" in status:
       return "Nevyřízeno"
    if "storno" in status or "s t o r n o" in status or "vratka"
    ↪  in status or "nedoručeno" in status or "reklamace" in
    ↪  status:
       return "storno/vratka"
    if "vyř" in status or "vybavená" in status or "doručeno" in
    ↪  status or "kompletně vyřízená" or "dokončena":
       return "hotovo"
    else:
       return "čeká"
```

Dále je nutné pro atribut **creationtime** nastavit správný datový typ místo vychozího „object“:

```
dataFrames["\"order\""]["creationtime"] =
↪  pd.to_datetime(dataFrames["\"order\""]["creationtime"],
↪  format='%Y-%m-%d')
```

Díky tomuto řádku lze s atributem skutečně pracovat jako s datem.

10.2.3 Pre-processing entity item_variant

Stejně jako v ostatních případech, se skript zbavuje atributů, které nejsou ve velké míře využívány. Jedná se o:

- parameter_color
- ISBN
- manufacturercode

- code
- item_ean

Tyto informace je vhodné mít v databázi pro případné další zpracování, nebo export pro business inteligence systémy. Z pohledu predikce nebo odhadnutí vhodnosti dárku pro uživatele již tak podstatné nejsou.

10.2.4 Další transformace

Skript mimo transformací popsaných v předchozích kapitolách stanovuje ještě další zajímavé informace. Jednou z nich je stanovení atributu **season**, který podle měsíce, ve kterém byla objednávka vytvořena, určí roční období.

Další transformací je získání části měsíce, ve kterém byla objednávka vytvořena. Měsíc je rozdělen po sedmi dnech, výsledkem je tedy číslo 1 až 4 ve sloupci „part_of_month“. Viz následující kousek kódu:

```
def ceilMonth(month):
   val = math.ceil(month / 7)
   if (val == 5):
      return 4
return val

merged["part_of_month"] = merged["day"].apply(ceilMonth)
```

Další transformace, kterou provádím je určení cenových hladin pro varianty produktů. Tyto hladiny jsem stanovil jako intervaly následovně:

- $<0,150)$
- $<150,500)$
- $<500,1000)$
- $<1000,2500)$
- $<2500,5000)$
- $<5000,\infty)$

Skript na konci svého běhu spojí jednotlivé dataFramy, které představují jednotlivé entity z databáze v jeden velký data set. Data jsou spojena pomocí metody merge, která jako parametry přijímá dva dataframy, které jsou určeny ke spojení a pomocí nastavení parametru „on“ se nastaví „cizí klíče“, přes které se data propojí. V jazyce SQL by se dalo říci, že tabulky jsou spojeny pomocí INNER JOIN. Tento výsledek je navracen k dalším analytickým výpočtům, které představím v dalších kapitolách. Ještě před představením dalších analytických kroků zmíním, že skript generuje dvě varianty těchto výsledků.

Jedna varianta je úplné provázání, tj. využijí se všechny získané entity, to má z výsledek, že pro jednu objednávku je zde N produktů a pro každý produkt je zde až N kategorií - tedy části objednávek se opakují. Druhá varianta je pak data set bez kategorií, kde se každý produkt v objednávce vyskytuje pouze jednou.

10.3 Analýza významu samotných dat

Jako poslední nezbývá než analyzovat data z pohledu toho jaké užitečné informace z nich lze vyčíst pro zadání této práce. Data jsem analyzoval dalo by se říci, ze dvou pohledů. Jeden je pohled na globální chování vzhledem k času, druhý je pak vyhodnocení chování v rámci nějakých skupin.

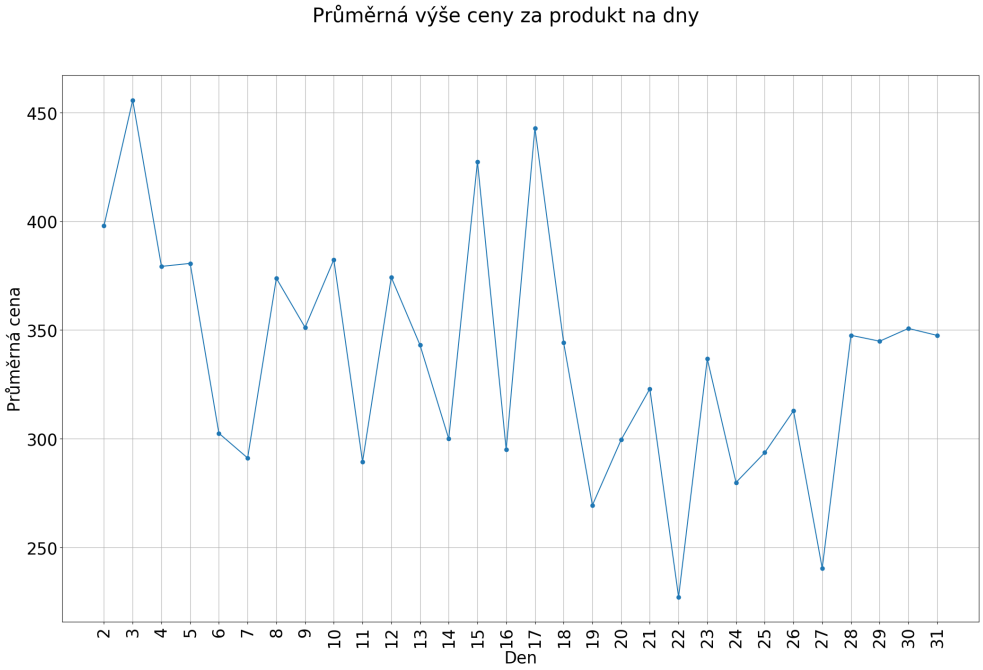
10.3.1 Vyhodnocení dat vůči času

Pro určení vhodné kategorie (nebo produktu) jak k inzerci, tak jako přání pro uživatele lze vycházet z časových řad. A to buď z jejich extrémů nebo jejich dlouhodobého vývoji. Jedním z takových ukazatelů se mi jeví průměrná cena produktu v daný den resp. měsíc. Podobná analýza, která může poskytnout zajímavá data, je počet objednávek.

Analýza vývoje ceny

Nejdříve rozeberu graf 10.5 na straně 66, který ukazuje vývoj ceny za jednotlivý produkt, který se v objednávce vyskytuje v závislosti na konkrétním dni v měsíci. Graf vynechal záměrně první den měsíce, neb se zde vyskytuje anomálie (průměrná cena byla přes 20 000,- Kč). Ostatní hodnoty poukazují na to, že začátkem měsíce se obvykle pohybuje cena v rozmezí 400 - 450,- Kč a koncem prvního týdne tento trend klesá. Poté hodnota osciluje kolem ceny 350,- Kč a v půlce měsíce se opět dostává k hodnotě ze začátku měsíce. Poté již cena opět klesá. Tyto popsané trendy mohou být způsobeny různými vlivy. Jedním může být příchod mzdy zaměstnancům, tím by se vysvětloval i pokles cen ke konci měsíce, kdy již lidé mají méně peněz.

Další graf 10.6 na straně 67 popisuje stejnou situaci, ale tentokrát již v závislosti na měsících. Zde vidíme anomálii (i když ne tak markantní jako v případně dní) v měsíci únoru. To může být dáno i tím, že produkty, které byly drahé se zakoupili např. první den v měsíci únoru. Další vývoj je víceméně konstatní pod hranicí ceny 500,- za produkt a od července do prosince by se dalo říci, že nastupuje do malého normálního rozložení. S nejvyššími cenami se lze dle těchto dat setkat v měsíci září a říjnu.



Obrázek 10.5: Analýza průměrné ceny v daný den v měsíci bez anomálie

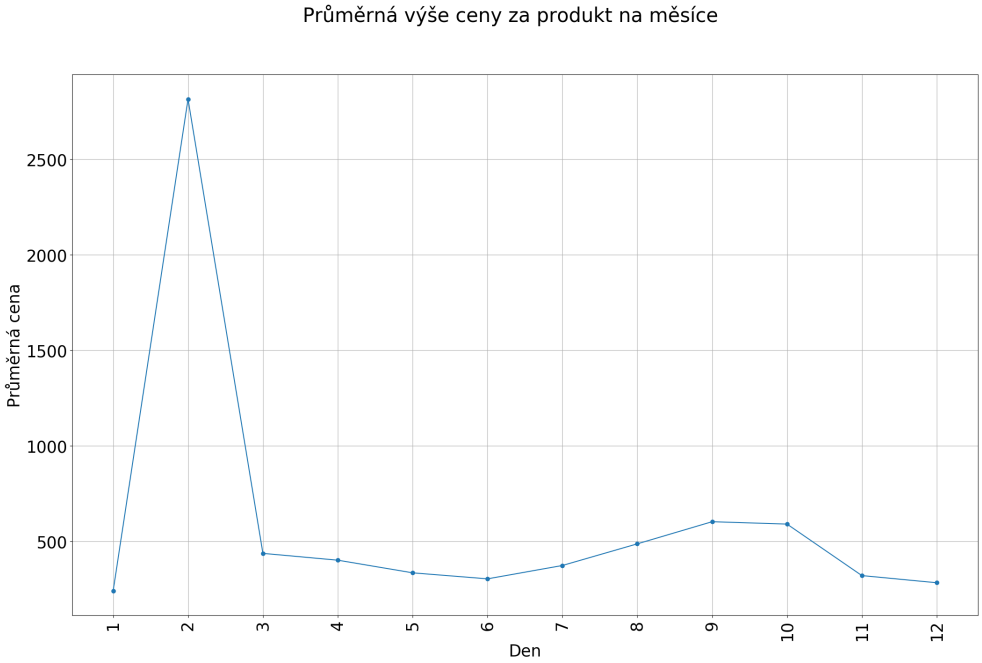
10.3.2 Analýza vývoje počtu objednávek

Obdobně jako na průměrnou cenu se lze podívat na vývoj počtu objednávek v závislosti na měsíci, z těchto dat lze pak zjistit oblíbené kategorie napříč měsíci nebo v sezóně. Dále lze zjistit jaký typ produktů je nejvíce oblíbený — e-shopy můžou mít kategorie například „gift“, „product-set“, nebo čistě jen „product“ a další.

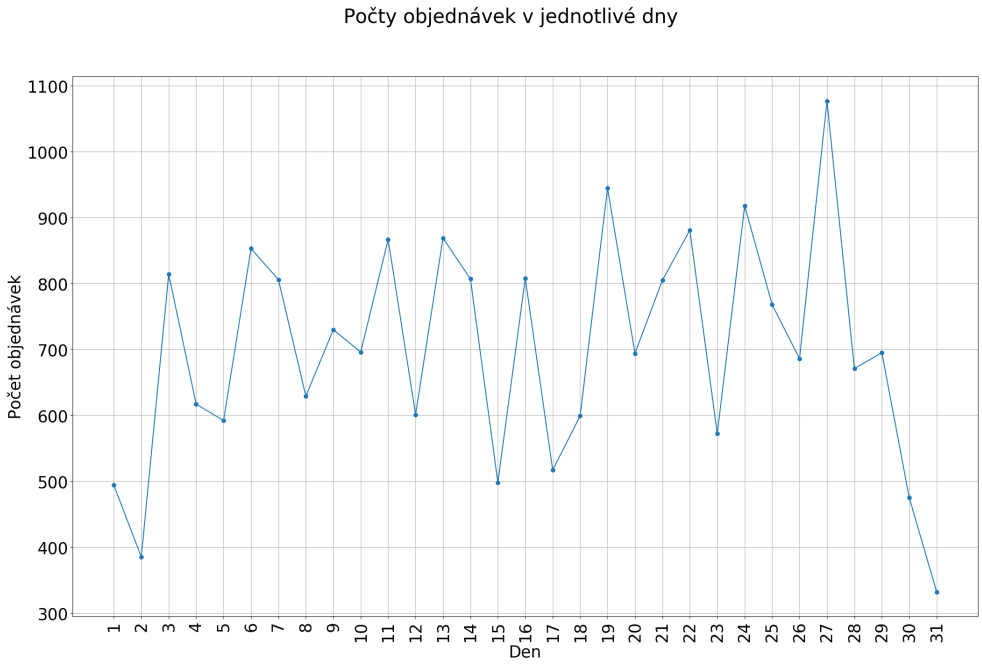
Počty objednávek v jednotlivých dnech

Vývoj počtu objednávek v jednotlivých dnech by měl opět reflektovat trendy z předchozí kapitoly, kdy ke konci měsíce lidé již čekají na výplaty. Jak to je ve skutečnosti, opět nejlépe popisuje graf 10.7 na straně 67.

Z grafu lze vyčíst, že předchozí tvrzení a trendy tak úplně neplatí a počty objednávek v jednotlivých dnech oscilují kolem čísla 700. Ke konci měsíce lze vidět extrém v podobě počtu objednávek někde mezi 1000 až 1100. To může být způsobeno například opačným tvrzením než v teorii o odstavec výše a to, že lidem naopak zbyly nějaké peníze a rozhodli se je utratit. Nicméně řekl bych, že z tohoto grafu spíše nejde moc zajímavých informací vyčíst.



Obrázek 10.6: Analýza průměrné ceny v měsíci

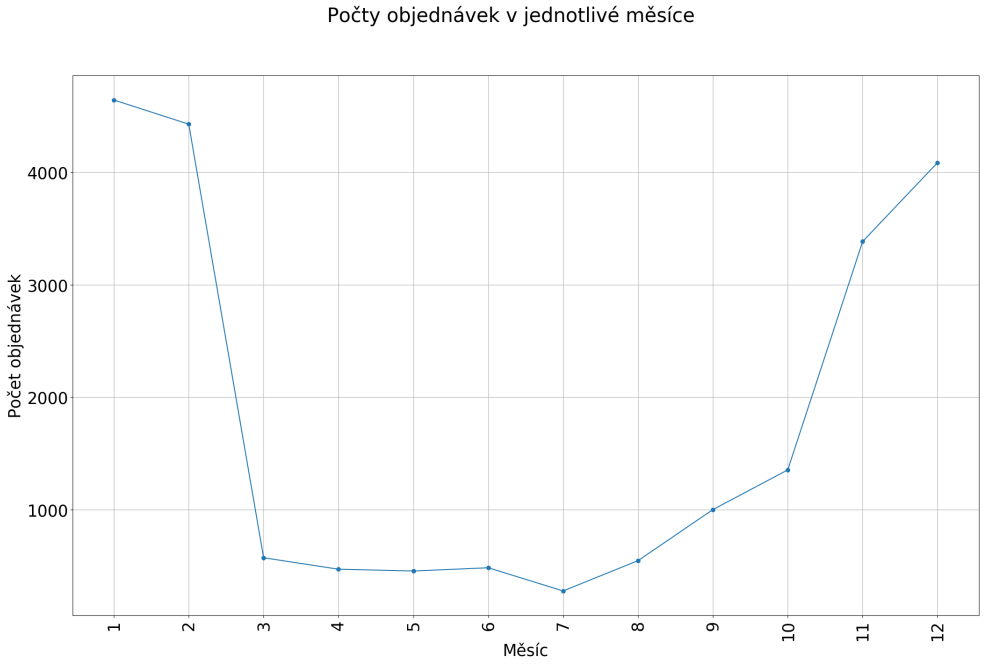


Obrázek 10.7: Počty objednávek v jednotlivých dnech

Počty objednávek v jednotlivých měsících

Dalším důležitým ukazatelem je objem objednávek v jednotlivých měsících. Tento vztah popisuje graf 10.8 na straně 68.

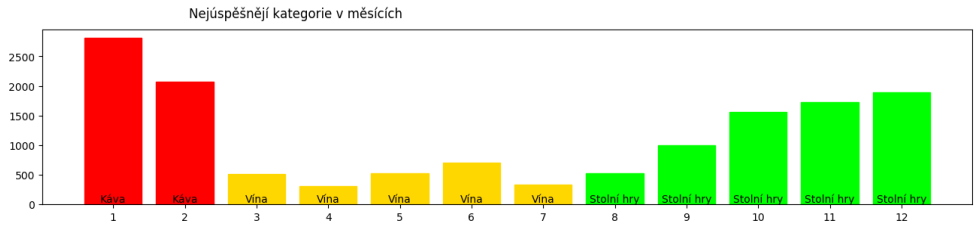
Trendy jsou zde analyticky pěkné. Přes jaro a léto jsou počty objednávek v útlumu ve skoro stejné hladině — cca 500. Od měsíce srpna se objem postupně zvyšuje a dosahuje svého vrcholu v prosinci a lednu, od této chvíle následuje opět klesání. Opět je zde vidět, že kdyby graf začínal červencem a končil červnem, dostaneme víceméně normální rozdělení. Z tohoto grafu lze usuzovat, že internetové obchody mají největší „žně“ právě přes vánoční svátky a při povánočních slevách.



Obrázek 10.8: Počty objednávek v jednotlivých měsících

Nejoblíbenější kategorie v měsíci

Vzhledem k sezónnosti některých produktů je pro e-shopy důležitá informace o tom, jaký produkt, resp. kategorie produktů, je v dané sezóně či měsíci oblíbená. Toto opět popíši na následujících grafech. Na grafu 10.9 na straně 69 jsou zobrazeny nejúspěšnější kategorie (na počet objednávek) vzhledem k měsíci. Začátkem roku jde o kávu, přes jaro a léto o kategorii Vína, od srpna o stolní hry.



Obrázek 10.9: Nejúspěšnější kategorie v jednotlivých měsících

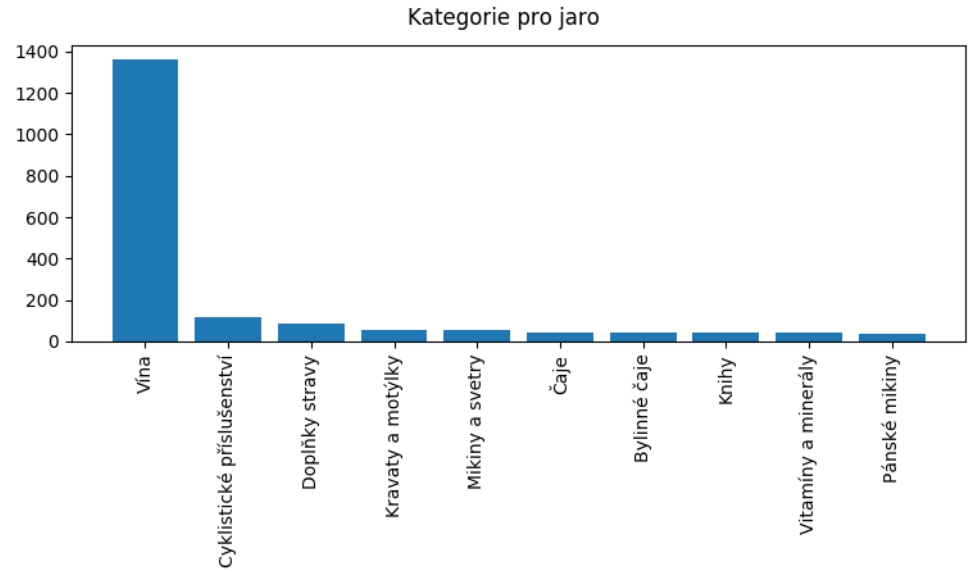
Obdobně je zajímavé se podívat nejen na měsíce, ale na konkrétní roční období. Grafy 10.10 na straně 69, 10.11 na straně 70, 10.12 na straně 71,10.13 na straně 72 zobrazují přesně tyto trendy.

Pro jaro je suveréně nejoblíbenější kategorie Vína, na druhé pozici Cyklistické potřeby a za nimi doplňky stravy — což zjevně reflektuje přípravu na začátek cyklistické sezóny.

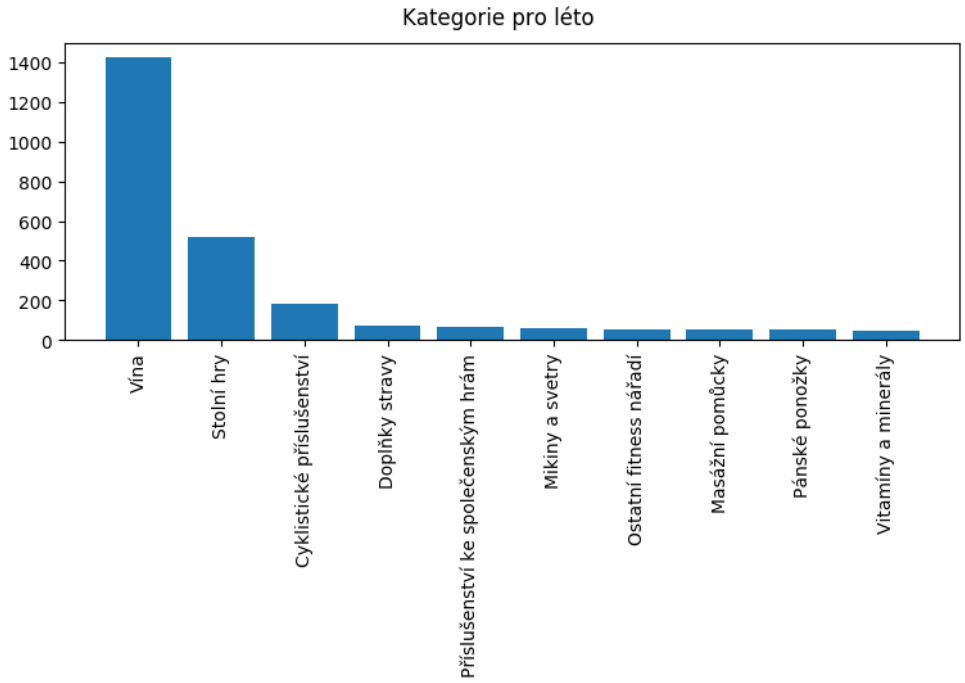
Pro léto jsou nejatraktivnější stále vína, za nimi stolní hry a poté cyklistické příslušenství.

V případě podzimu již trendy opustily kategorii vína a vyhrávají stolní hry, za nimi víno, které následuje káva.

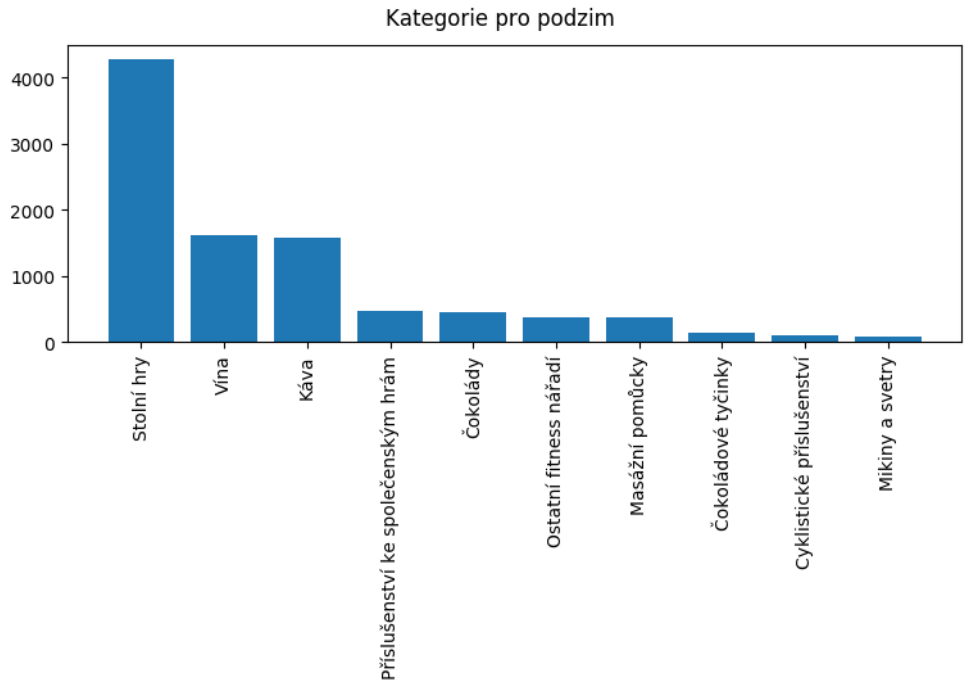
V případě zimní sezóny vévodí již káva, za kterou následují stolní hry a za nimi vína.



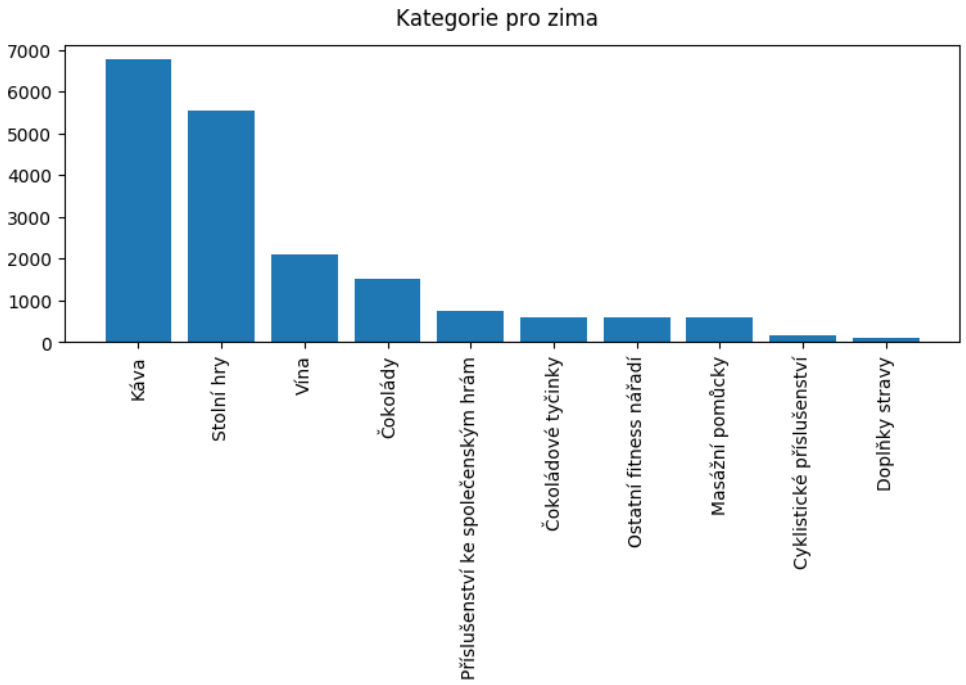
Obrázek 10.10: Nejúspěšnější kategorie pro jaro



Obrázek 10.11: Nejúspěšnější kategorie pro léto



Obrázek 10.12: Nejúspěšnější kategorie pro podzim



Obrázek 10.13: Nejúspěšnější kategorie pro zimu

Typy produktů obsažené v objednávkách

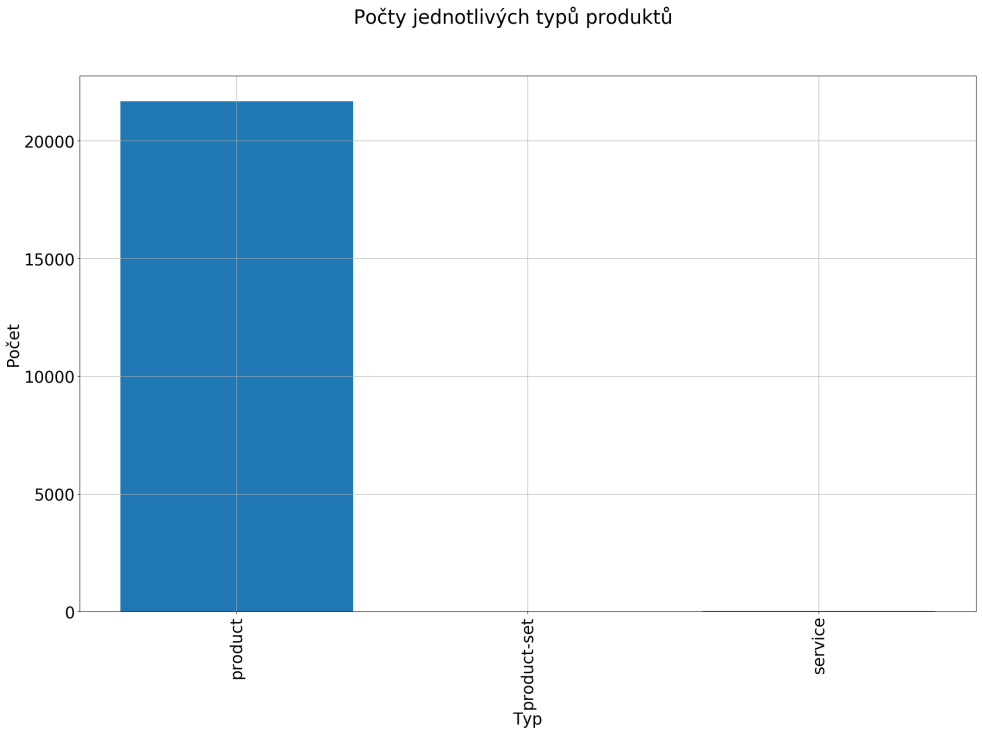
Zajímavé by mohlo být zastoupení jednotlivých typů produktů v jednotlivých objednávkách. Typem se zde myslí, zda-li jde o službu, dárek, produktový set nebo čistě o produkt a tak podobně. Z grafu 10.14 na straně 73 je vidět, že žádnou zajímavou informaci nedostáváme a nejčastějším typem položky je čistě samotný produkt.

10.3.3 Analýza cenových kategorií

Další možností je analyzovat zájem o jednotlivé cenové kategorie, které jsou definované v kapitole 10.2.4 na straně 64. Opět je uvedu grafy pro dny i měsíce. Pro obě varianty uvedu i logaritmickou transformaci.

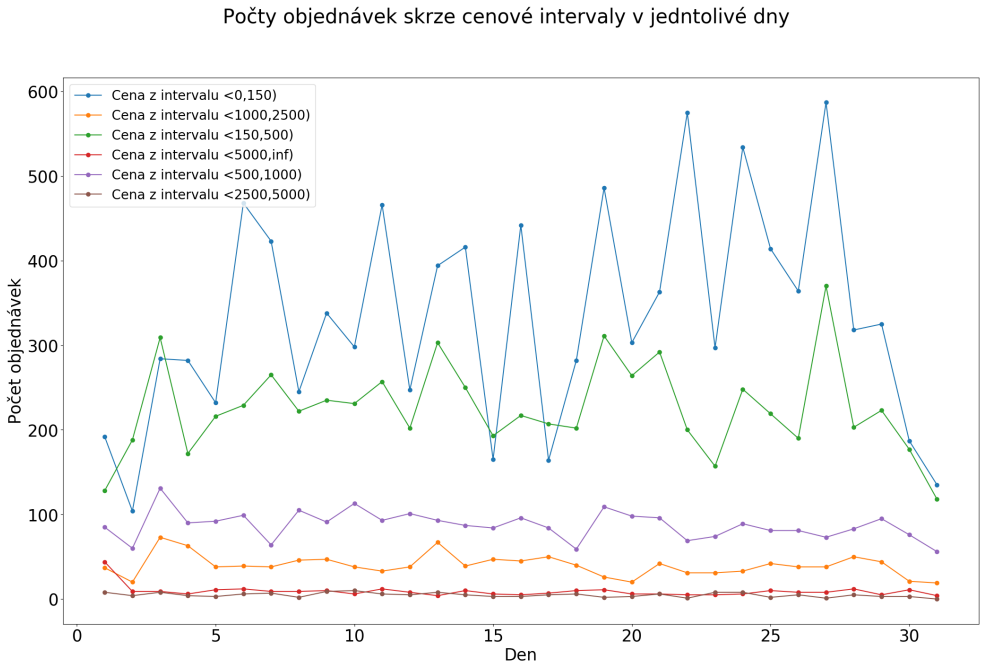
Analýza cenových kategorií v jednotlivých dnech

Grafy 10.15 na straně 74 a 10.16 na straně 75 ukazují četnost cenové kategorie v daný den. Oba grafy ukazují stejnou informaci, avšak graf, který je po logaritmické transformaci je poněkud přehlednější. Z tohoto grafu sice nelze vyčíst přesná čísla, ale ukazuje dost dobře oblíbenost jednotlivých kategorií skrze dny.

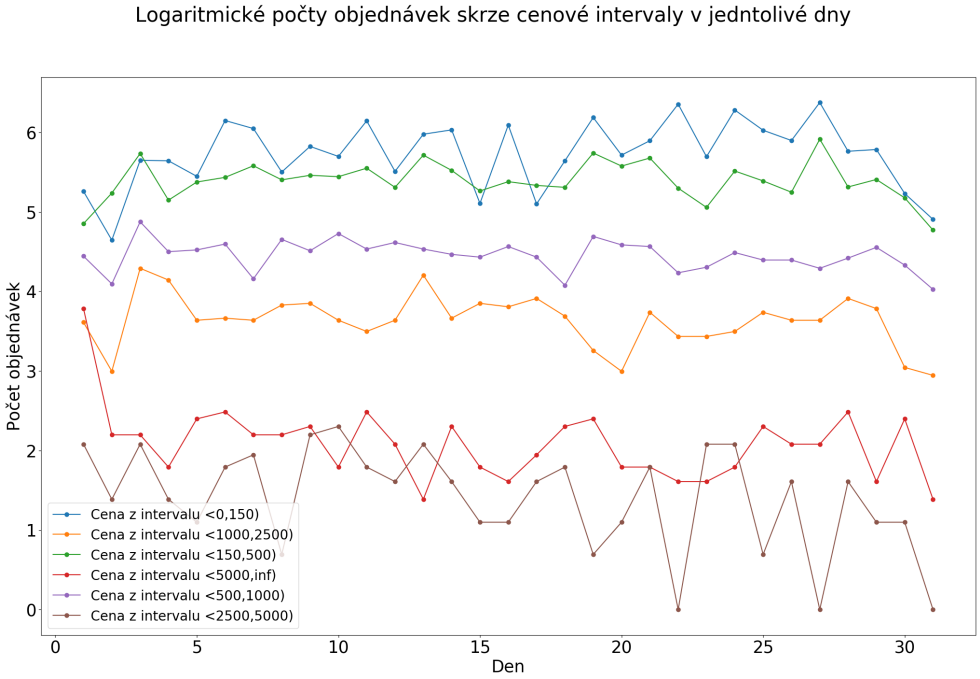


Obrázek 10.14: Nejčastější typy položek

Nejlépe ja na tom nejlevnější kategorie, která pokrývá ceny od 0 do 150,- Kč. Následují produkty z cenové kategorie 150 až 500, takto kategorie neustále rostou až do posledních dvou, kdy není jasné, která je oblíbenější. Ostatně, pokud porovnáme grafy pro dvě nejdražší kategorie, dalo by se říci, že se četnosti pohybují spíše v řádě jednotek pro obě dvě.



Obrázek 10.15: Vývoj počtu objednávek v cenových kategoriích v závislosti na dni v měsíci

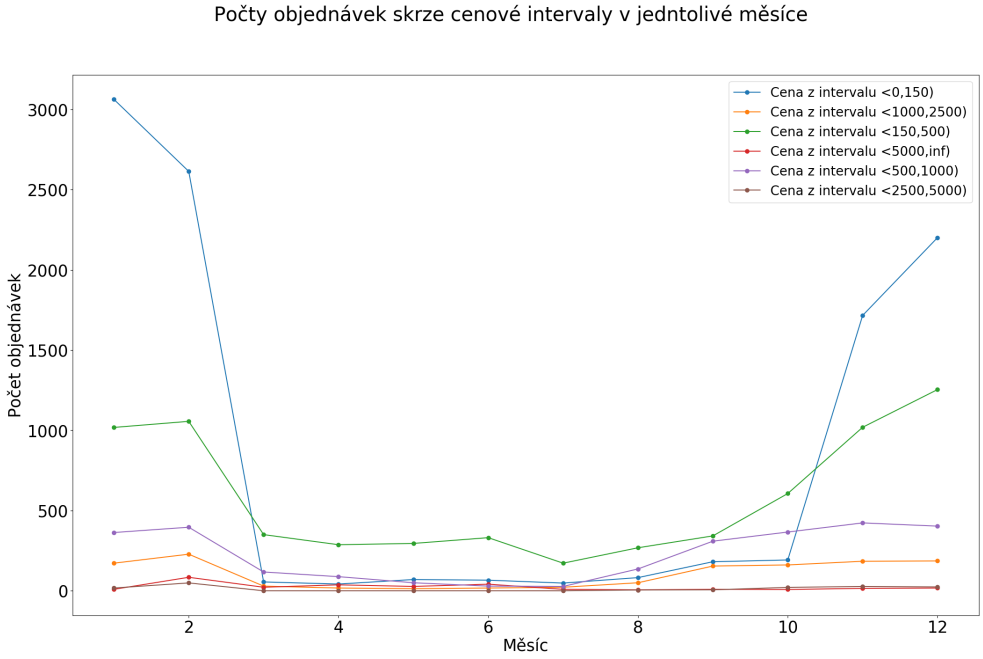


Obrázek 10.16: Vývoj počtu objednávek po logaritmické transformaci v cenových kategoriích v závislosti na dni v měsíci

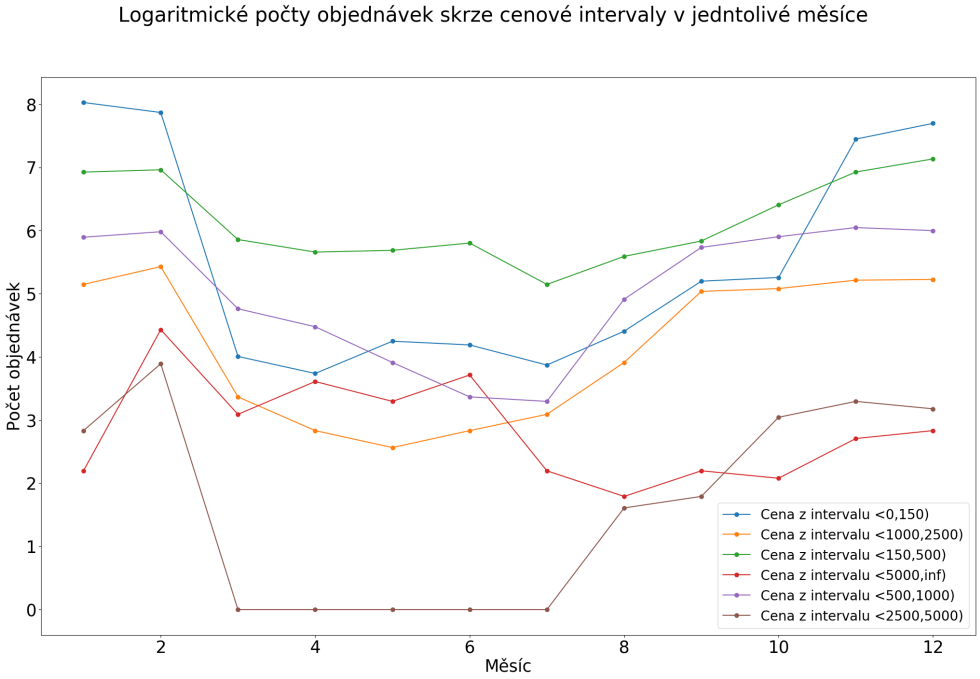
Analýza cenových kategorií v jednotlivých měsících

Stejně jako v předchozích kapitolách i zde ještě setrvám u stejné kategorie měření, ale tentokrát bude pro tuto práci zajímavé rozložení skrze měsíce v roce.

Jedná se o grafy 10.17 na straně 76 a 10.18 na straně 77. Na obou grafech lze vypozorovat, že opět kopírují normální rozložení v cca období od srpna do července. Cenové kategorie si poměrně drží své pozice a stoupají nebo klesají více méně pokaždé úměrně stejně. První zajímavý jev nastává v červnu, kdy se na třetí nejčastější cenovou kategorii dostávají nejdražší produkty. Další zajímavý jev se začíná dít od letních prázdnin, tedy v období od července, kdy kategorie produktů v ceně 500 až 1000 Kč začne stoupat tak, že je četnější než kategorie 0 až 150,- Kč, to se však v říjnu vrací opět ke starým pořádkům.



Obrázek 10.17: Vývoj počtu objednávek v cenových kategoriích v závislosti na měsíci v roce



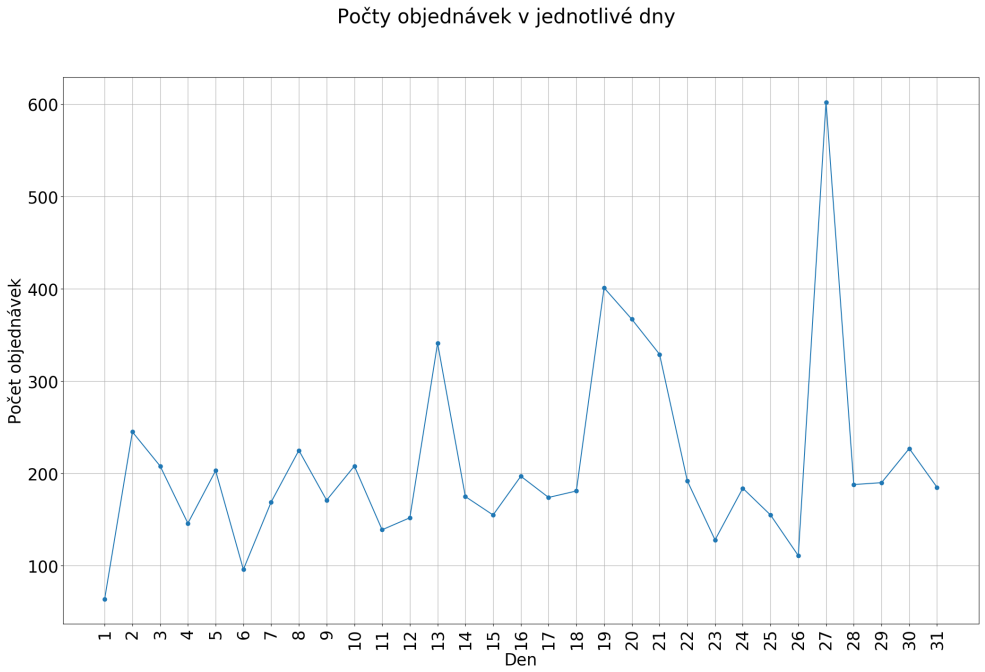
Obrázek 10.18: Vývoj počtu objednávek po logaritmické transformaci v cenových kategoriích v závislosti na měsíci v roce

10.3.4 Analýza jednotlivých produktových kategorií

Data lze analyzovat také pro konkrétní kategorie, zde se jeví poměrně zajímavý ukazatel — průměrný počet objednávek závislý na dni v měsíci, nebo měsíci v roce.

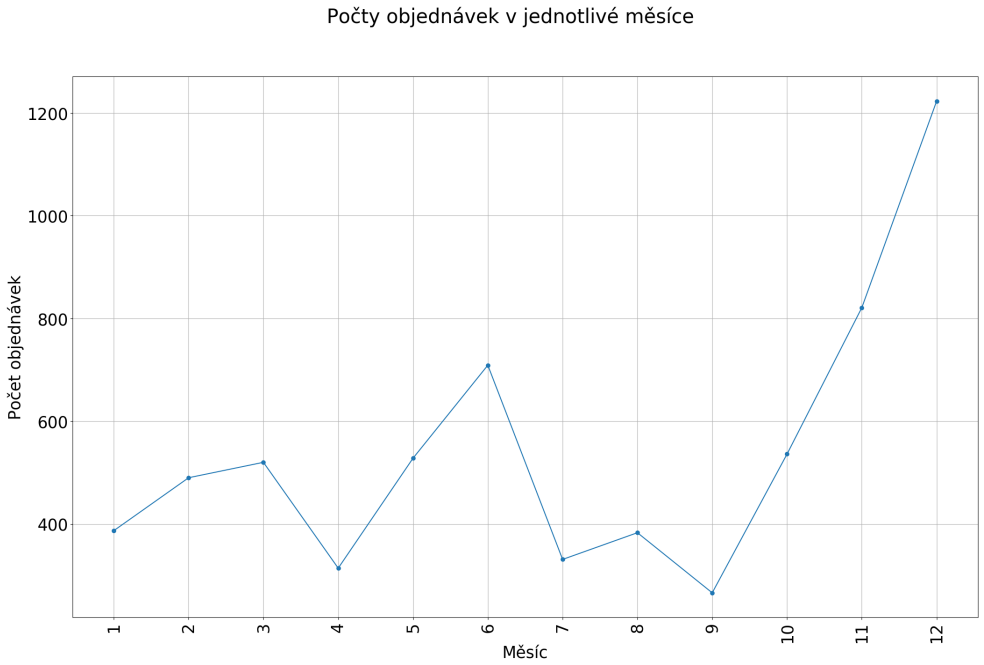
Kategorií je skutečně velké množství – skript v tomto případě generuje 176 grafů jak pro dny, tak měsíce. Není tedy možné v této analyzovat zde kategorii po kategorii. Já jsem samozřejmě grafy s daty pro všechny kategorie prošel viz příloha B na straně 117. Zde bych rád rozebral alespoň pár zajímavých případů.

Jednou z kategorií, která má dostatek dat je kategorie Vína. Grafy 10.19 a 10.20 na stranách 78 a 79 popisují četnost této kategorie v objednávkách v závislosti na dnech a měsících.



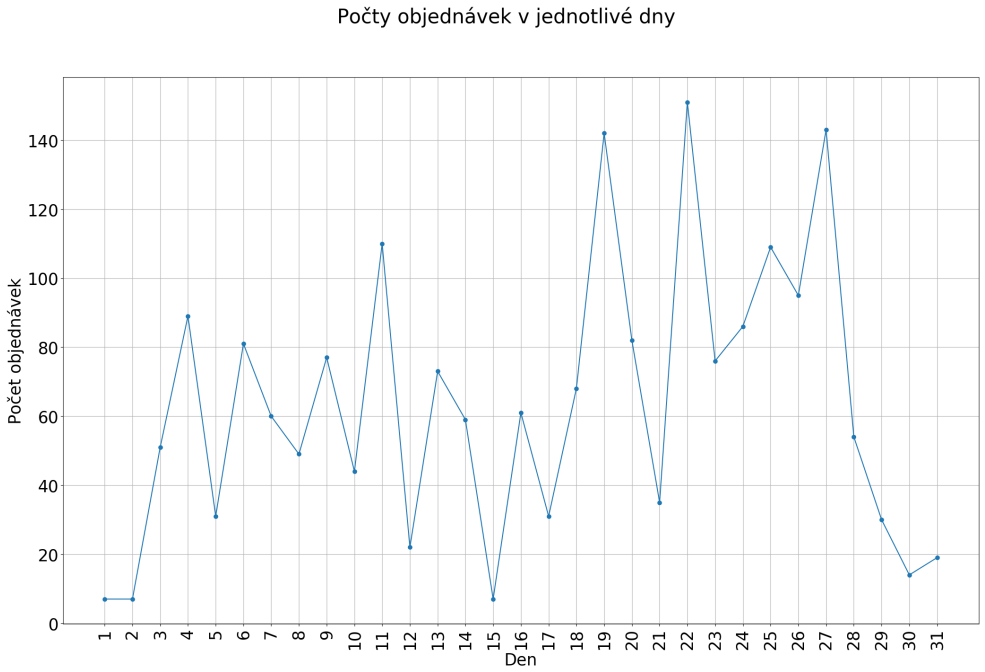
Obrázek 10.19: Vývoj počtu objednávek pro kategorii „Vína“ v závislosti na dni v měsíci

Na prvním z grafů je vidět poměrně stabilní poptávka, která roste kolem třetího týdne v měsíci, následně lehce upadá a následně se dostane do maxima. Data jsou dostupná pro všechny dny v měsíci. V případě druhého zmíněného grafu je vidět vývoj v roce, kdy poptávka stoupá v letních měsících, ale úplně nejvíce roste na podzim a to až do konce roku. Předpokládám, že u této kategorie je to dáno zejména tím, že lidé tráví více času doma a jsou zde svátky jako je Mikuláš, Vánoce a především Silvestr.



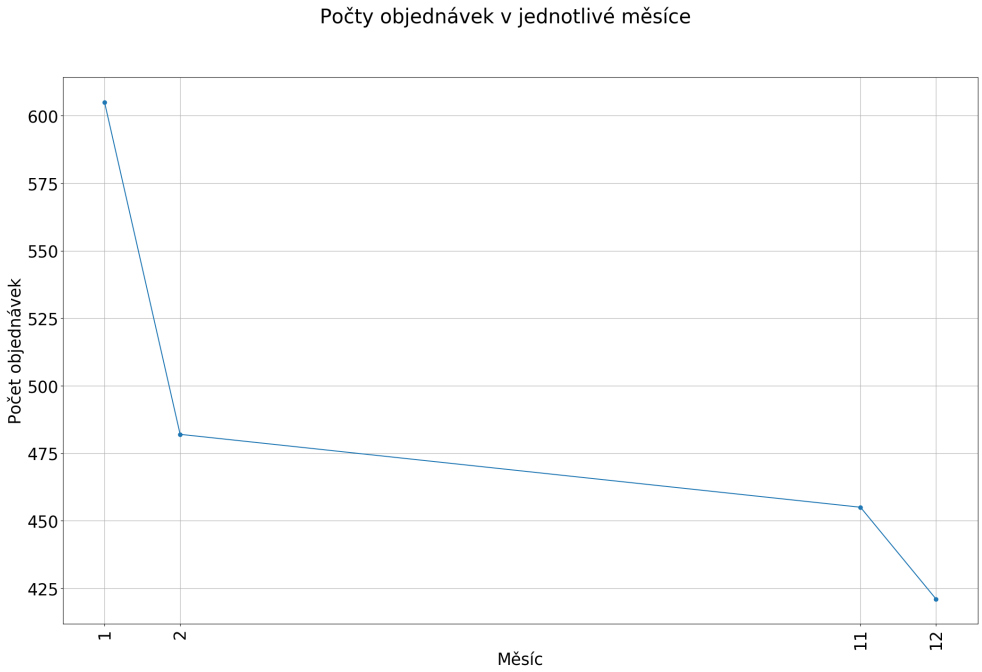
Obrázek 10.20: Vývoj počtu objednávek pro kategorii „Vína“ v závislosti na měsíci v roce

Tento příklad je, řekl bych ideální, bohužel ne pro všechny kategorie jsou data takto pěkná. Relativním zklamáním je pro mě kategorie „Čokolády“, která dosahuje poměrně velké četnosti (1 963), ale graf je pěkný jen pro dny — viz graf 10.21 na straně 80 a graf 10.22 na straně 81. Graf pro dny je poměrně pěkný, dalo by se z něj odvodit, že kategorie je oblíbená zejména vždy na konci měsíce.



Obrázek 10.21: Vývoj počtu objednávek pro kategorii „Čokolády“ v závislosti na dni v měsíci

Graf zobrazující četnost kategorie v průběhu roku nám toho mnoho neřekne o oblíbenosti kategorie, ale lze se díky těmto datům domnívat, že danou kategorii nenabízí mnoho e-shopů, které využívají Wowee Partner API a data jsou tak tímto faktem značně ovlivněna. Další fakt, který lze z tohoto grafu vyčíst je ten, že e-shop je buď krátce na trhu a ještě nemá dostatek dat a nebo data zkrátka chybí. Pokud bychom se podívali na data blíže do databáze, tak zjištění bude takové, že data pro měsíc listopad a prosinec jsou z minulého roku a leden s únorem již z roku letošního. Z tohoto lze odhadovat, že jde skutečně v tomto případě o jeden e-shop, který je na trhu krátkou dobu a data tak skutečně ještě nemá.



Obrázek 10.22: Vývoj počtu objednávek pro kategorii „Čokolády“ v závislosti na měsíci v roce

Další případ, který se v datech objevuje, jsou data v rámci jednotek, nebo jen jeden bod. S dovolením, zde další graf neuvedu, ostatní grafy vypadají analogicky.

10.3.5 Závěr

Data, která jsem získal z jednotlivých e-shopů skrze Shotpet API jsou velkého objemu a v mnoha případech nad nimi lze provádět i další analýzy, než jsou uvedeny zde v této práci. I z tohoto důvodu je jedním z bodů zadání této práce poskytnutí těchto dat BI systémům.

Povaha dat a jejich objem je však v mnoha případech ovlivněna konkrétními e-shopy, které využívají Wowee Partner API. To by bylo zcela přirozené, ale data z velkého zaběhlého e-shopu pak „přebijí“ ta data, která jsou získána z e-shopů menších – extrémní případ kdyby k využití Wowee Partner API přistoupila např. Alza.cz, zcela jistě by elektronikou přebila veškeré současné oblíbené kategorie. A bylo by vhodné hledat trendy ještě v rámci e-shopů seskupených podle například obratu, či jiného ukazatele ukazující jeho velikost.

Partner API dále obsahuje řadu začínajících e-shopů, což nyní sice zkresluje výstupní data, ale je zde velký potenciál do budoucnosti. V průběhu času

tedy i statistiky pro tyto e-shopy mohou tvořit značnou část globálních ukazatelů jako jsou oblíbené kategorie pro daný měsíc nebo roční období.

Kapitola 11

Finální Algoritmizace

Proces finální algoritmizace lze rozdělit do dalších čtyř částí:

1. vytvoření základní datové struktury ve Wowee Partner API databázi
2. synchronizace dat mezi migrační databází a Partner API databází
3. napočítání trendů
4. Algoritmy a API
    a) endpoint pro export do BI systémů
    b) endpointy pro zapisování statistik
    c) endpoint a algoritmus navracející nejvhodnější možnou reklamu
    d) endpoint a algoritmus pro vyhodnocení vhodnosti sbírky

Tyto části popíši v nadcházejících kapitolách.

11.1 Enviromentální proměnné

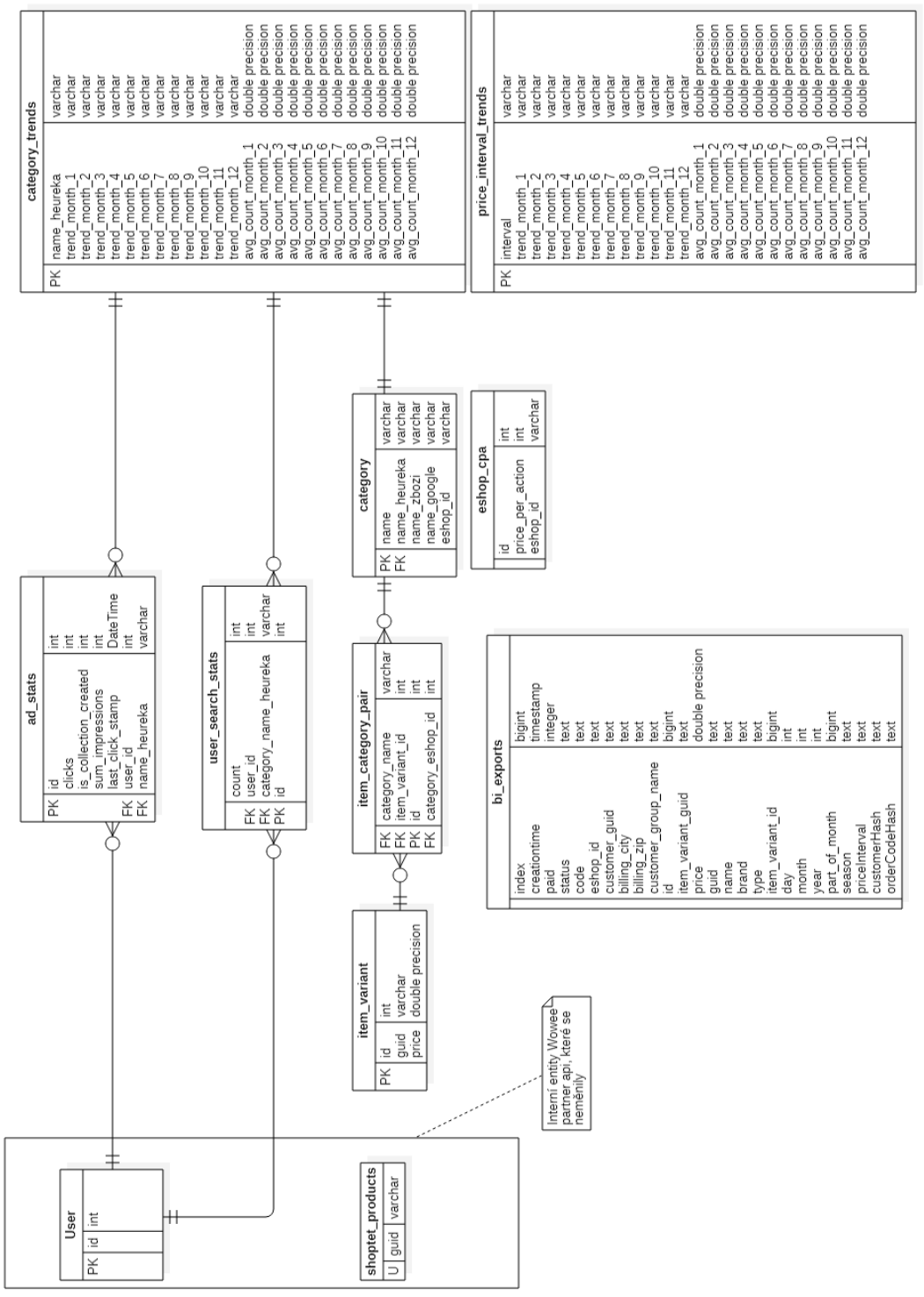
Docker kontejner pro Python předpokládá následující enviromentální proměnné:

- PG_DATABASE=nazev-partner-api-databaze
- DOCKER=true — jedná se o spuštěný kontejner
- PG_USER=uzivatel-partner-api-databaze
- PG_PASSWORD=heslo-partner-api-databaze
- PREDICT_DB=nazev-databaze-pro-data-ze-shoptet-api
- PREDICT_DB_USER=uzivatel-databaze-pro-data-ze-shoptet-api
- PREDICT_DB_PASS=heslo-databaze-pro-data-ze-shoptet-api
- PG_PRIMARY_PORT=vychozi-port
- DB_HOST=psql — hostname partner api databáze v rámci kontejnerů

11.2 Vytvoření základní datové struktury v Partner API

Databáze popsaná v kapitole 7 na straně 37 slouží čistě jen k uložení dat získaných z API Shoptet. Důvod pro tento návrh je především oddělení dat, která mohou být svojí povahou citlivá (a to jak pro e-shopy, tak pro jejich zákazníky). Proto je nutné data zpracovat (o preprocessingu více kapitola 10.2 na straně 60) a následně provést jejich migraci právě do databáze Wowee Partner API. Tato databáze je již poměrně obsáhlá, mým cílem je přidat tabulky, které jsou nutné pro účel této práce — nebudu se tedy věnovat celkovému schématu databáze, ale jen mnou vytvořené části — viz ER model 11.1 na straně 85.

Ze zmíněných přístupů k implementaci popsaných v kapitole 2 na straně 9 jsem zvolil časové řady a analýzu trendů. Data která mám k dispozici tomuto přístupu odpovídají nejlépe. Stejně tak vstupy, kterých je minimum a je tedy důležité vycházet skutečně z historie.



Obrázek 11.1: Entitně relační databázový model nových entit ve Wowee Partner API

S popisem zmíněného modelu začnu v jeho levé horní části entitou **User**. Tato entita je z Wowee Partner API a představuje uživatele Wowee.

Každý uživatel má n záznamů v entitě **ad_stats**. Tato entita uchovává statistiky uživatelských interakcí vůči zobrazeným reklamám. Entita uchovává následující:

1. id – autoinkrementální id pro každý záznam
2. clicks – celkový počet prokliků
3. is_collection_created – celkový počet vytvořených sbírek
4. sum_impression – počet zobrazení (impresí)
5. last_click_stamp – poslední časová značka prokliku
6. user_id – cizí klíč id uživatele
7. name_heureka – cizí klíč názvu kategorie

Další vztah, který má tato entita je s entitou **category_trends**. Zmíněné dva vztahy lze interpretovat následovně: Každý uživatel má n statistik k m záznamům v entitě category_trends.

Entita category_trends představuje základní stavební kámen pro další algoritmizaci. Obsahuje všechny názvy kategorií z Heureka.cz (ty, které jsem získal z dat získaných skriptem v kapitole 9 na straně 47). Ke každé kategorii pak uchovává průměrný počet objednávek pro všech 12 měsíců (atributy avg_count_mont_{cislo_mesice}) a především pak trendy pro daný měsíc (atributy trend_month_{cislo_mesice}), více o identifikaci trendů kapitola 2.9.3 na straně 19) a samotná algoritmizace v kapitole 11.4 na straně 89. Tuto metodu jsem zvolil ze dvou důvodů — jedná se de facto o analýzu historických dat (žádná další data tato práce neuvažuje) a k dispozici pro určení vhodné kategorie bude jen minimum vstupů.

Pro tabulku **user_search_stats** platí vše obdobně jen s tím rozdílem, že tato tabulka uchovává informaci o uživatelském vyhledání nějaké kategorie.

Třetí vazba, kterou má entita category_trends je s entitou **category**, ta má dále vazbu s entitou **item_category_pair** a ta s entitou **item_variant**. Tyto tři entity jsou naprosto identické s těmi uvedenými v kapitole 7.1 na straně 37 až na pár výjimek. První z nich je samotná vazba entity category s category_trends. Ta je poměrně logická neboť entita category uchovává informace o tom, jaká kategorie e-eshopu je synonymem pro kategorii z Heureky. Dalším rozdílem je méně atributů u item_variant — atributů je skutečně méně, produkty jako takové jsou totiž v Partner API již stažené — entita **shoptet_products** (obě části však pracují se stejnými e-shopy) — a to pro účely napovídání při vyhledávání. Je třeba tedy jen několik atributů navíc, které obsaženy nejsou. Oba záznamy mají však atribut **guid**, skrze který lze

entity propojit. V průběhu vývoje jsem neměl bohužel možnost mít aktualizovaná data, tudíž zde chybí přímá vazba.

Entita **price_interval_trends** je obdoba již popsané entity category_trends, uchovává však informace o trendech v cenových hladinách (viz kapitola 10.2.4 na straně 64).

Entita **eshop_cpa** slouží k uchování cen za akci od jednotlivých eshopů (jedná se spíše o prototyp, protože ceny za uskutečněný nákup a např. registrai se budou lišit). Tato entita opět nemá vazbu na entitu eshop z Partner API ze stejných důvodů jako jsou uvedeny u případu item_variant a shoptet_products.

Poslední entitou je **bi_exports**. Tato entita obsahuje zpracovaná data z kapitoly 10.2 na straně 60, určená k exportu do dalších business inteligence systémů. Jedná se o směs atributů, které již byly popsány v kapitole 7.1 na straně 37 a 10.2 na straně 60. Novými atributy jsou **customerHash** a **orderCodeHash**, pro hashe reálných hodnot těchto atributů.

11.2.1 Algoritmus

Datovou strukturu popsanou v předešlé kapitole 11.2 lze v databázi Partner API vytvořit následujícím příkazem:

```
docker exec partner-api_python_1 python3 dpTool.py --init
```

Tento příkaz má za následek, že Python zavolá následující bash skript:

```
#!/bin/bash
psql -h $DB_HOST -p $PG_PRIMARY_PORT -d $PG_DATABASE -U $PG_USER
↪  -f ./bash_scripts/data_structures.sql
```

psql je front-endový nástroj pro databáze PostreSQL. V tomto případě se připojí k databázi specifikovanou čtyřmi enviromentálními proměnnými (viz 11.1 na straně 84) – $DB_HOST, $PG_PRIMARY_PORT, $PG_DATABASE a $PG_USER – následně se provede obsah souboru data_structures.sql, který obsahuje definici uvedených tabulek a vztahů v SQL.

11.3 Synchronizace dat mezi databázemi

Synchronizace dat probíhá opět pomocí stejného Python skriptu jen s jiným parametrem:

```
docker exec partner-api_python_1 python3 dpTool.py --sync
```

Tento příkaz má za následek zavolání následujících řádek:

```
1 initialization.setCategoriesNames(categories_all["name_heureka"],
  ↪ connection_papi)
2 initialization.setCpa(dataFrames["eshop"], connection_papi)
3 initialization.setPriceIntervals(data["priceInterval"].unique(),
  ↪ connection_papi)
4 newData =
  ↪ preprocessing.createBIEndpoint(pure_data_without_categories)
5 newData.to_sql('bi_exports', engine, if_exists='replace')
6 initialization.syncData()
```

Řádky jedna až tři nastavují výchozí hodnoty v entitách tak, aby se daly hodnoty použít jako cizí klíče. Jde o entity category_trends — zde se vloží všechny jména kategorií z Heureky, které jsem dosud získal, dál se vloží všechny e-shopy do entity eshop_cpa a nastaví se jim cena za akci (výchozí je cena 0). Nakonec se vloží unikátní hodnoty pro trendy (viz 10.2.4 na straně 64) do tabulky price_interval_trends (analogický případ jako category_trends).

Řádek číslo čtyři upraví výchozí dataFrame z preprocessingu a doplní do něj hash hodnoty pro **customer_guid** a **code** (tento atribut představuje kód objednávky):

```
1 def createBIEndpoint(mergedDataFrame):
2     mergedDataFrame["customerHash"] =
      ↪ mergedDataFrame["customer_guid"].apply(lambda x :
      ↪ hashlib.sha256((str(x)+
3
      ↪ "estridenteEsapedirreyahoraibaapago").encode('utf-8')).hexdigest())
4     mergedDataFrame["orderCodeHash"] =
      ↪ mergedDataFrame["code"].apply(lambda x :
      ↪ hashlib.sha256((str(x)+
5
      ↪ "estridenteEsape2020dirreyahoraibaapago").encode('utf-8')).hexdigest())
6
7     return mergedDataFrame
```

Řádek číslo pět pak upravený dataFrame (bez propojení s kategoriemi) naimportuje do entity **bi_exports**. Importuje se v režimu replace, což má za následek, že se přepíší hodnoty v případě shody (tato entita by měla fungovat jako read-only, takže by se v důsledku měli jen přidat nové záznamy).

Jako poslední se volá funkce syncData(), která vyvolá spuštění bash skriptu obsahující následující řádky:

```
1 #!/bin/bash
2 psql -h $DB_HOST -p $PG_PRIMARY_PORT -U $PG_USER -d $PG_DATABASE
  ↪ -f ./bash_scripts/truncate.sql # delete old data
```

```
3 psql -h $DB_HOST -p $PG_PRIMARY_PORT -U $PREDICT_DB_USER -d
  ↪ $PREDICT_DB -c "copy(SELECT id, guid, price FROM
  ↪ item_variant) to stdout" > dumpfile
4 psql -h $DB_HOST -p $PG_PRIMARY_PORT -U $PG_USER -d $PG_DATABASE
  ↪ -c 'copy item_variant from stdin' < dumpfile
5 pg_dump -h $DB_HOST -p $PG_PRIMARY_PORT -U $PREDICT_DB_USER
  ↪ --data-only -t category $PREDICT_DB > dumpfile # from db to
  ↪ file
6 psql -h $DB_HOST -p $PG_PRIMARY_PORT -U $PG_USER -d $PG_DATABASE
  ↪ < dumpfile # from file to prod db
7 pg_dump -h $DB_HOST -p $PG_PRIMARY_PORT -U $PREDICT_DB_USER
  ↪ --data-only -T item_category_pair_seq -t item_category_pair
  ↪ $PREDICT_DB > dumpfile #from db to file
8 psql -h $DB_HOST -p $PG_PRIMARY_PORT -U $PG_USER -d $PG_DATABASE
  ↪ < dumpfile #from file to prod db
9 rm dumpfile
```

Skript využívá již dříve popsané utility psql (kapitola 11.2.1 na straně 87) v kombinaci s utilitou pg_dump, která administrátorům PostgreSQL databází umožňuje snadné exportování dat z databáze. Opět se zde vyskytují enviromentální proměnné, které jsou nastavené v konfiguraci docker kontejnerů.

Jako první se vymažou stará data v Partner API databázi, následuje export a na další řádku import entity item_variant s vybranými atributy. Dále export a import entity category a nakonec export a import entity item_category_pair. Poslední příkaz pak vymaže soubor používaný k dumpu.

11.4 Výpočet trendů

Poté co je vytvořena datová struktura a jsou v n obsažena i data, lze spustit výpočet trendů. To opět lze provést pomocí Python skriptu:

```
docker exec partner-api_python_1 python3 dpTool.py --trends
```

Tento příkaz má za následek zavolání následujícího:

```
for category in categories_all["name_heureka"]:
   analyze.setTrendsForCategories(data.copy(), category,
   ↪ connection_papi, "categories")
for interval in data["priceInterval"].unique():
   analyze.setTrendsForPriceIntervals(data.copy(), interval,
   ↪ connection_papi, "price_intervals")
```

Pro každou kategorii z Heureky se nastaví trendy pro každý měsíc, totéž pro každý „priceInterval“ neboli cenovou hladinu. Trend odráží chování zákazníka, které je třeba podpořit.

Ukáži tedy jen jeden příklad:

```
def setTrendsForCategories(data, category, connection,
↪  subdirectory):
   trend_tables, avg_count_tables = init_trend_arrays()
   cursor = connection.cursor()

   data = data.loc[data["name_heureka"] == category]
   Position, grouped = getMA(data)
   trend_table = "category_trends"
   setTrendsForTrendTable(Position, avg_count_tables, category,
   ↪  connection, cursor, grouped, trend_table,
   ↪  trend_tables,"""WHERE name_heureka=%s""", subdirectory)
   return grouped
```

Nejprve se nastaví názvy atributů pro trendy a průměrné počty v daném měsíci (první řádek). Dále se na řádku číslo pět vyberou jen data spadající do aktuálně zpracovávané kategorie (cenové hladiny). Poté se zavolá funkce getMA(data), která napočítá klouzavé průměry (její obsah ukážu níže). Po této metodě se již volá setTrendsForTrendTable, kde se identifikují trendy a uloží se do databáze.

```
def getMA(data):
   data = data.groupby(["year",
   ↪  "month"])["month"].count().reset_index(name="count")
   data =
   ↪  data.groupby(["month"])["count"].mean().reset_index(name="mean")

   for mth in range(1, 13):
      if data.loc[data["month"] == mth]["month"].count() == 0:
         data = data.append({"month": mth, "mean": 0},
         ↪  ignore_index=True)
   data = data.copy().sort_values(by='month', ascending=True)
   data["month"] = data["month"].astype(int)
   grouped = data
   grouped["3ma"] =
   ↪  (grouped.copy())["mean"].rolling(window=3).mean()
      grouped["3ma"].iloc[0] = (grouped["mean"].iloc[0] +
      ↪  grouped["mean"].iloc[11] + grouped["mean"].iloc[10]) /
      ↪  3
   grouped["3ma"].iloc[1] = (grouped["mean"].iloc[0] +
   ↪  grouped["mean"].iloc[1] + grouped["mean"].iloc[11]) / 3
   Position = []
   p = ""
   return Position, grouped
```

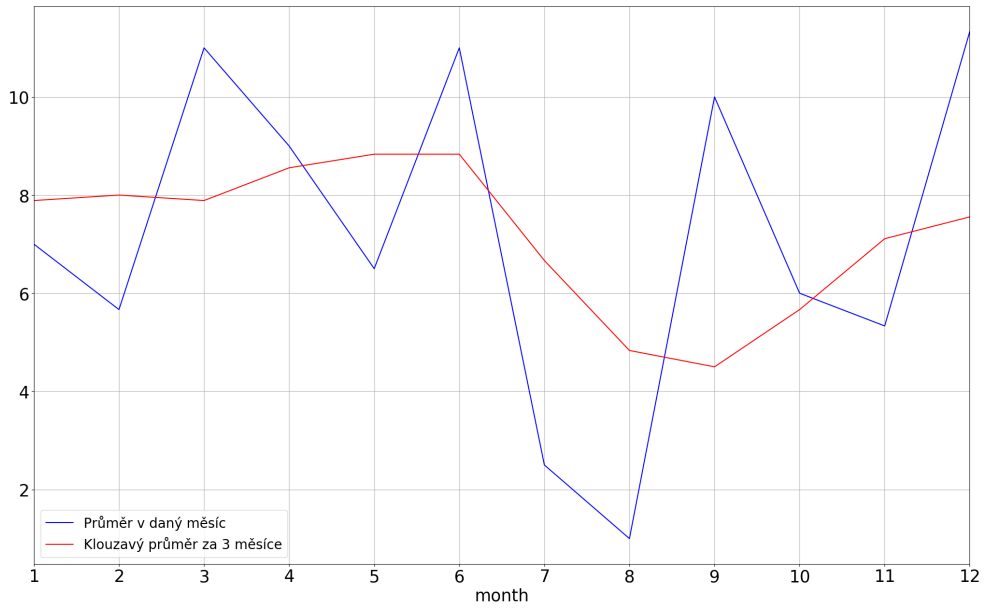
Na druhém řádku vidíme získání počtu záznamů seskupených podle roku a měsíce, následuje řádek, který spočítá průměr na konkrétními měsíci. Cyklus for zaručuje, že se v datech vyskytují vždy všechny měsíce (ty které nebyly v původní datech jsou nulové). Na řádku 11 se využije funkce **rolling** nad sloupcem „mean“. Okno jsem zvolil rovno třem - klouzavým průměr se tedy vždy počítá za 3 měsíce. Jelikož se jedná o necentrovaný klouzavý průměr, tak tato funkce nespočítá hodnoty pro první dva měsíce. Ty jsem dopočítal ručně — využívám toho, že měsíce se v roce opakují. Hodnoty jsou navíc průměry výskytů za jednotlivé měsíce. Pro leden jsem tedy využil hodnot z prosince a listopadu (a ledna). Pro únor hodnoty z prosince a ledna (a února). Data navíc díky předchozímu doplnění vždy budou existovat.

Po výše popsané funkci se volá funkce setTrendsForTrendTable, která již identifikuje trendy a vkládá je do databáze:

```
1  def setTrendsForTrendTable(Position, avg_count_tables,
   ↪  groupedBy, connection, cursor, grouped, trend_table,
   ↪  trend_tables, where_part, subdirectory):
2
3      for i in range(0, grouped.shape[0]):
4          if grouped["mean"].iloc[i] > grouped["3ma"].iloc[i]:
5              p = "Up"
6              Position.append("Up")
7          else:
8              Position.append("Down")
9              p = "Down"
10
11         postgres_insert_query = """ UPDATE """ + trend_table + """
           ↪  SET """ + avg_count_tables[i] + """=%s, """ +
           ↪  trend_tables[i] + """=%s """ + where_part
           ↪  record_to_insert = (str(grouped["mean"].iloc[i]), p,
           ↪  groupedBy)
12         cursor.execute(postgres_insert_query, record_to_insert)
13         connection.commit()
14     grouped["Trend"] = Position
15     ax = plt.gca()
16     grouped["Průměr v daný měsíc"] = grouped["mean"]
17     grouped["Klouzavý průměr za 3 měsíce"] = grouped["3ma"]
18     grouped.plot(kind='line', x='month', y='Průměr v daný měsíc',
       ↪  color='blue', ax=ax, grid=True, xticks=range(1, 13))
19     grouped.plot(kind='line', x='month', y='Klouzavý průměr za 3
       ↪  měsíce', color='red', ax=ax, grid=True)
20
21     plt.savefig('./graphs/means/' + str(subdirectory) + '/' +
       ↪  str(groupedBy) + '.png', bbox_inches='tight')
```

```
22      plt.close()
23      print(grouped)
```

Po definici funkce přichází blok s for cyklusem, který porovnává průměr výskytů kategorire (cenové hladiny) v objednávkách s klouzavým průměrem za tři měsíce. Pokud je první hodnota větší než klouzavý průměr, označí se jako trend „Up“ v opačném případě jako „Down“. Poté se hodnota pro příslušné měsíce vloží do databáze. Řádky počínaje řádkem 15 a konče řádkem 22 vykreslují grafy pro každou kategorii a ukládají je do adresáře na řádku 21. Tyto grafy mi posloužili k ověření správnosti algoritmu. Jako ilustraci příkládám graf 11.2 na straně 92.



Obrázek 11.2: Klouzavý průměr vs. průměr v daný měsíc pro kategorii čokolády

11.5 API

API — Application Programming Interface — je založeno na balíčku express pro Node.js, který je popsán v kapitole 4.4.1 na straně 30. Nachází se v kontejneru, který nese název „partner-api_papi_1“ viz kapitola 8 na straně 43.

Mnou vytvořené endpoint jsou definovány jako všechny ostatní v souboru app.js a jsou definovány následovně:

```
app.use('/stats', statsRouter)
app.use('/export', exportRouter)
```

```
app.use('/predict', predictRouter)
app.use('/usersuit', usersuitRouter)
```

První endpoint obsluhuje zapsání statistik. Druhý slouží pro exportování dat do BI systémů. Endpoint /predict slouží pro určení nejvhodnější kategorie pro reklamu a poslední endpoint /usersuit ohodnocuje vhodnost přání pro uživatele. Každému z endpointů zde věnuji podkapitolu.

11.5.1 Endpoint /stats

Tento endpoint sám o sobě nic nedělá, avšak má pod sebou řadu dalších. Jedná se o endpointy, přidávající data do tabulek ad_stats a user_stats. Tyto statistiky dále využívám pro samotné vybrání vhodné reklamy a ohodnocení vhodnosti produktu:

- /stats/ad
- /stats/click
- /stats/collection
- /stats/impression
- /stats/search

Všechny výše zmíněné endpointy jsou založeny na HTTP metodě POST. Všechny metody vyžadují v těle požadavku parametry **„category“** a **„userid“** Endpoint **/stats/ad** slouží pro vytvoření záznamu v entitě ad_stats s defaultními (nulovými) počty pro konkrétní kombinaci uživatelského id a kategorii.

Bod **/stats/click** přičte ke konkrétnímu uživateli a kategorii kliknutí (atribut clicks), toto analogicky platí pro **/stats/collection** a **/stats/impression**.

V případě bodu **/stats/search** se v případě neexistujícího výchozí záznamu vytvoří záznam s výchozí hodnotou 1 pro uživatele a kategorii, nebo se počet hledání pouze inkrementuje.

11.5.2 Endpoint /export

Tento endpoint nedisponuje žádnými dceřinými endpointy jako tomu bylo v předchozím případě. Očekává se zde HTTP metoda GET s následujícími parametry:

- categories
- page
- itemsperpage

Parametr **categories** určuje zda se mají záznamy jednotlivých výskytů produktových variant ještě dále napojit skrze „INNER JOIN“ s konkrétními kategoriemi ve kterých se vyskytují – očekává se roven „1“ pro spojení.

Parametry **page** a **itemsperpage** jsou použity pro podporu stránkování.

Příklad navrácených hodnot pro **/export?page=1200&itemsperpage=1&categories=1** (hashe byly zkráceny):


```
{"status":"ok","error":null,
"data":[{"id":"199508",
"creationtime":"2017-04-21T00:00:00.000Z",
"paid":1,
"status":"hotovo","billing_zip":"46","customer_group_name":"VIP",
"price":249,
"name":"Vína","brand":"Víno Kadrnka (ČR)",
"type":"product","part_of_month":"3","season":"jaro",
"priceInterval":"<150,500)",
"customer":"877077f7d7224ec130dcc4e0d...2293797a613fee46d8c0268",
"order":"35448c972d7179d9da77298...670c8c86b8709a779f8c39deba6d",
"name_heureka":"Vína","name_zbozi":"Vína","name_google":"Víno"}],
"meta":null}
```


11.5.3 Endpoint /predict

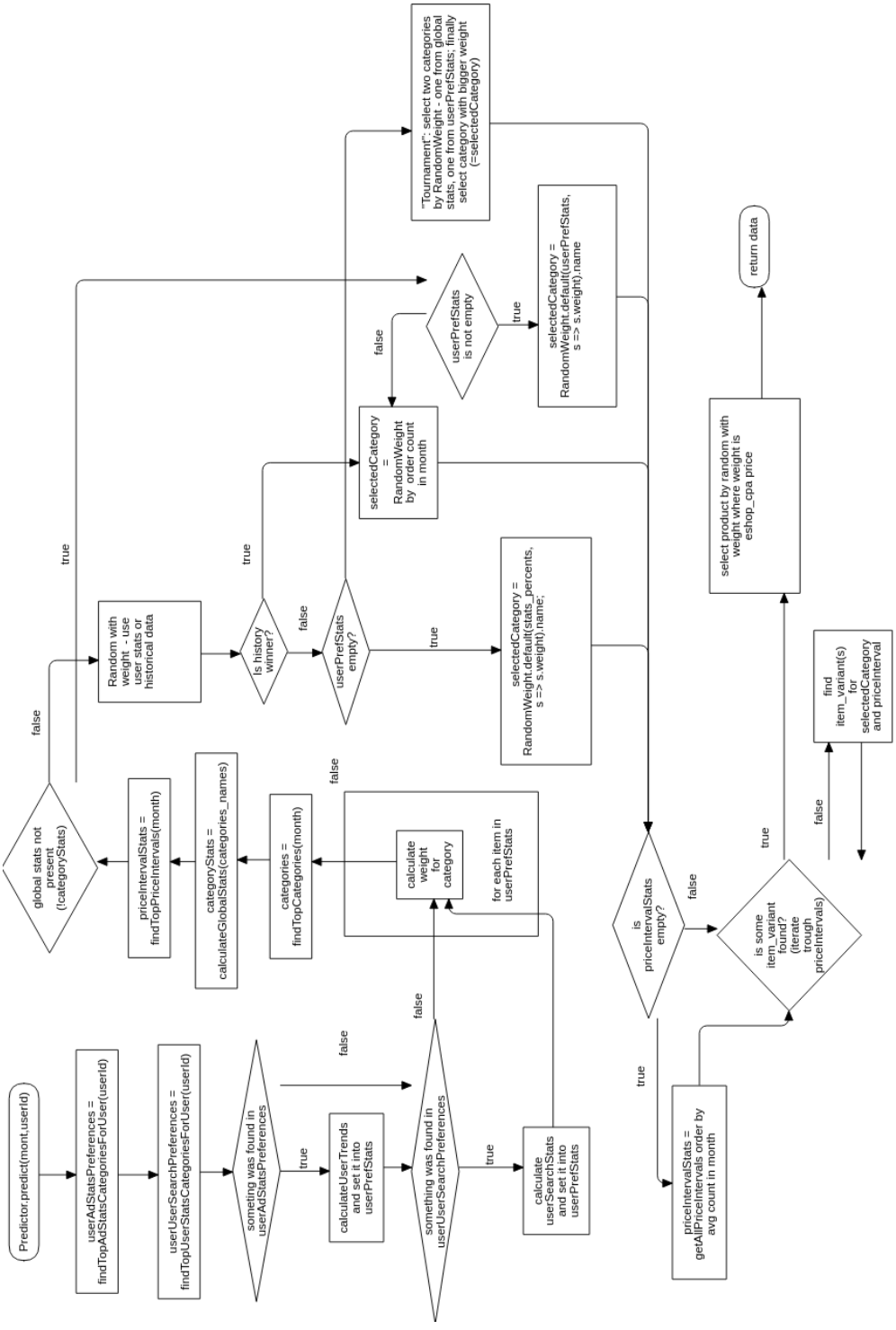
Tento endpoint podporuje metodu HTTP GET a potřebuje, k tomu aby navrátil data, parametry **month** a **userId**. Na základě nich navrátí nejlepší možnou kategorii potažmo produkt pro reklamu.

Flow předpovědi je poměrně složitá, proto zde namísto úryvků z kódu okomentuji UML diagram 11.3, který je k nalazení na straně 95.

Algoritmus předá parametry z HTTP GET požadavku funkci predict v objektu Predictor. Ta začíná tím, že nalezne statistiky pro konkrétního uživatele. Ty jsou dvojího druhu, tzv. „ad_stats“ a „user_search_stats“. První z jmenovaných jsou statistiky vedené k reakcím uživatele na konkrétní reklamu, druhá statistika udržuje informaci o tom, jaké kategorie se týkalo uživatelovo vyhledávání v aplikaci Wowee.

Následuje sestavení objektu userPrefStats, což je objekt typu Map. Ten se skládá z výsledků předchozích dvou volání. Jsou v něm uloženy všechny statistiky a jako klíč je použit název kategorie, ke které reakce patří. Části daného objektu se sestavují za předpokladu, že existují — pokud ne, pokračuje se.

Ve chvíli, kdy je vytvořen objekt userPrefStats, projdou se všechny jeho záznamy a spočítá se pro každý vážený průměr na základě získaných uživa-



Obrázek 11.3: Flowchart predikce

telských statistik. Mnou zvolený vzorec je následovný:

$$weight = \frac{\frac{pocetKliku}{pocetImpresi} + 2 * \frac{pocetVytvorenychKolekci}{pcoetImpresi} + 2 * \frac{pocetHledani}{sumaHledani}}{5}$$

Výsledek se uloží do pole, jako objekt mající vlastnost name — název kategorie a weight — spočítaná váha. Tímto výsledkem se přepíše původní objekt userPrefStats.

Jako další se najde až pět kategorií, které mají stoupající trend (ten je již identifikován a uložen v tabulce category_trends) pro požadovaný měsíc. Tento výsledek se uloží, zpracuje a získá se pole s názvy kategorií, které je následně předáno funkci calculateGlobalStats, která nalezne uživatelské statistiky pro nalezené kategorie. U těchto statistik se jako váha již počítá jen průměr úspěšných prokliků, tedy:

$$weight = \frac{pocetKliku}{pocetImpresi}$$

Tyto statistiky nejsou tak cenné jako předchozí, proto je jejich kalkulace zjednodušena.

Jako další algoritmus najde, podobně jako tomu bylo pro kategorie, cenové intervaly (entita price_interval_trends), které mají stoupající trend v požadovaném měsíci.

Nyní již následuje samotný výběr kategorie.

Pokud neexistují globální uživatelské statistiky a nejsou ani žádné statistiky k požadovanému uživateli, vybere se kategorie náhodně s uvažovanou vahou podle průměrného počtu objednávek v daný měsíc (tento údaj je opět v entitě category_trends). Pokud uživatelské statistiky existují, vybere se kategorie náhodně s vahou podle nich. Pokud existují, algoritmus se náhodně váhově rozhodne jestli dále bude postupovat podle uživatelských statistik, nebo dá příležitost čistě historickým datům. Zde jsem zvolil váhy 60 pro uživatelské statistiky a 40 pro historická data.

V případě, že „vyhrají“ historická data, vybere se opět náhodným výběrem podle vah, které představují průměrné výskyty v daný měsíc.

V případě, že se dál počítá s uživatelskými statistikami a neexistují data k tomuto konkrétnímu uživateli, použijí se statistiky uživatelsky globální.

Pokud existují uživatelská data k tomuto uživateli, pokračuje se dál na „turnaj“. Zde se naleznou dvě kategorie pomocí náhodného výběru s vahou z uživatelských statistik a globálních statistik. Vyhraje ta kategorie, která má větší váhu.

Ve chvíli, když již algoritmus nalezl kategorii, zkontroluje, zda-li v předchozích krocích našel nějaké cenové hladiny. Pokud ne, získá všechny, a to v pořadí podle průměrného výskytu v měsíci.

Poté v cyklu algoritmus hledá konkrétní záznamy v entitě item_variant, které vyhovují nalezené kategorii a cenové hladině.

Ve chvíli, kdy je něco nalezeno, vybere se z výsledku náhodným výběrem s vahou konečný produkt resp. jeho guid. Váha v tomto případě představuje cenu za akci (jedná se o CPA viz 1.1.1 na straně 5), kterou platí e-shop Wowee (a tedy větší šanci mají ti, co platí více).

Příklad s testovacími daty

Pro HTTP požadavek:
/predict?userid=61&month=11
API navrací výsledek:


```
{"status":"ok","error":null,
   "data":{
      "category":"Káva",
         "product":{
            "name_heureka":"Káva",
            "guid":"adf4a424-6ce6-11e8-8216-002590dad85e",
            "name":"Káva",
            "eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
            "price":124.35,"price_per_action":0}},"meta":null}
```


11.5.4 Endpoint /usersuit

Tento endpoint předpokládá HTTP GET požadavek s parametry:

- month
- itemguid
- userid
- price

Tyto parametry pro mě představují parametry při vytváření přání. Na jejich základech endpoint ohodnotí vhodnost přání pro daného uživatele. Algoritmus samotný využívá již dříve napsaných funkcí pro předchozí endpoint. Jeho jádro je následující:

```
1 UserSuit.isItemTrendyForMonth(month,itemguid).then((prediction)=>{
2    let categories = prediction[1];
3
     ↪  UserSuit.isItemTrendForUserAd(userid,categories).then((adbool)=>{
4
        ↪  UserSuit.isItemTrendForUserSearch(userid,categories).then((ussbool)=>{
5
           ↪  UserSuit.isPriceInPriceIntervalUp(price,month).then((pibool)=>{
6          res.status(200).send(
```

```
7               getSuccessResponse(
8                   [pibool+ussbool+adbool+prediction[0],
9                   [pibool,ussbool,adbool,prediction[0]],
10                  prediction[1]])
11          );
12  }
```

První řádek zjistí, zda kategorie pro daný produkt má stoupající trend pro daný měsíc, třetí řádek zjistí, zda-li uživatel někdy reagoval na reklamu, následující řádek zjistí jestli ho někdy vyhledával na Wowee. A konečně poslední řádek zjistí, zda-li je cena pro dané přání ve stoupajícím trendu, či nikoliv.

Jako výsledek endpoint vrátí součet pozitivních hodnot (tedy ohodnocení 0-4), pole s bool hodnotami jako odpovědi na otázky výše. Jako poslední součást se vrátí kategorie, ve kterých se produkt vyskytuje.

Příklad

Pro HTTP dotaz **/usersuit?userid=61&month=11&itemguid=adf4a424-6ce6-11e8-8216-002590dad85e&price=10**

navrátí nedpoint následující výsledek

```
{"status":"ok","error":null,
   "data":[2,[true,false,false,true],["Káva"]],
   "meta":null}
```

Kapitola 12

Celková rekapitulace flow

Za předpokladu, že se nacházím v adresáři s kompletním projektem Wowee Partner API:

1. **docker-compose build**
   (vytvoření kontejnerů)
2. **docker-compose up**
   (spuštění kontejnerů)
3. **docker exec -it partner-api_papi_1**
   node /app/src/cron/predictionDataDownload.js
   (spuštění migrace mezi Shoptet API a mojí databází)
4. **docker exec partner-api_python_1 python3 dpTool.py –analysis**
   (spuštění analýzy dat)
5. **docker exec partner-api_python_1 python3 dpTool.py –init**
   inicilizace nových entit v Partner API databázi
6. **docker exec partner-api_python_1 python3 dpTool.py –sync**
   (synchronizace dat mezi databází Partner API a databází pro data z Shoptet API)
7. **docker exec partner-api_python_1 python3 dpTool.py –trends**
   (napočítání trendů do trendových tabulek a vytvoření grafů)

Python skript akceptuje ještě parametr –**delete**, který sloužil spíše při vývoji ke smazání tabulek, které předtím skript vytvořil. Body 3, 6, 7 je vhodné pouštět periodicky (cron).

Výsledkem je pak funkční API s nálsedujícími endpointy:

- **/stats/ad?category={category_name}&userid={userid}**
  (přidání nového záznamu do ad_stats s nulovými hodnotami)

- **/stats/click?category={category_name}&userid={userid}**
  (přidání kliknutí do statistik)
  (dále analogicky /stats/impression a /stats/collection)
- **/predict?month={month_number}&userid={userid}**
  (nalezení vhodné kategorie a produktu pro reklamu)
- **/usersuit?month={mont_number}**
  &itemguid={item_guid}&userid={userId}&price={price}
  (stanovení vhodnosti sbírky)

Kapitola 13

Testování a vyhodnocení algoritmu

Pro ověření funkčnosti algoritmu jsem provedl následující: Vždy jsem nasimuloval uživatelské statistiky a sledoval, zda algoritmus navrací očekávané výsledky:

1. UC prázdné statistiky
2. UC kategorie vína: 10 impresí a 5 kliků
3. UC kategorie vína: 10 impresí a 2 kolekce
4. UC kategorie vína a káva: 10 impresí a 2 kolekce

Tyto testy byly provedeny pro konkrétního uživatele — **user id 61** a v konkrétní měsíc — **leden (= 1)**. Tedy pro endpoint **/predict?month=1&userid=61** se jedná o následující výsledky za výše zmíněných podmínek: Výsledek **pro první use-case**:


```
{"category":"Káva","product":{"name_heureka":"Káva",
"guid":"58c04c67-5bcb-11e9-beb1-002590dad85e",
"name":"Zrnková káva",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":126.09,"price_per_action":0}}
```


UC1 ještě jednou (proveden znovu HTTP požadavek):


```
{"category":"Káva","product":{"name_heureka":"Káva",
"guid":"ad430c0c-6ce6-11e8-8216-002590dad85e",
"name":"Káva",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":127.83,"price_per_action":0}}
```


Výsledek pro **UC2**:

{"category":"Vína","product":{"name_heureka":"Vína",
"guid":"76e8e97c-b4d5-11e7-ae76-0cc47a6c92bc",
"name":"Vína",
"eshop_id":"bc915f31-7ee7-4abe-a826-be19be8879fb",
"price":149,"price_per_action":0}}

Výsledek pro **UC3**:

{"category":"Vína","product":{"name_heureka":"Vína",
"guid":"76e8e97c-b4d5-11e7-ae76-0cc47a6c92bc",
"name":"Víno Kadrnka (ČR)",
"eshop_id":"bc915f31-7ee7-4abe-a826-be19be8879fb",
"price":149,"price_per_action":0}}

Výsledky pro tři volání **UC4**:

{"category":"Káva","product":{"name_heureka":"Káva",
"guid":"6db2b2dd-82ac-11e8-8216-002590dad85e",
"name":"Instantní káva",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":7.83,"price_per_action":0}}

{"category":"Káva","product":{"name_heureka":"Káva",
"guid":"4c2f6004-5bc9-11e9-beb1-002590dad85e",
"name":"Káva",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":126.09,"price_per_action":0}}

{"category":"Čokolády","product":{"name_heureka":"Čokolády",
"guid":"95e4bba9-822b-11e8-8216-002590dad85e",
"name":"Čokoláda",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":66.09,"price_per_action":0}}

Z výše uvedených výsledků je vidět, že se v nich odráží trendy (zejména pokud nejsou ještě dostupná žádná data), které jsou popsány na grafu 10.9 na straně 69. Dále algoritmus vybírá kategorii, na kterou uživatel někdy v minulosti reagoval. V případě, že je dostupných více kategorií, resp. uživatel má nějakou interakci s reklamou a existují ještě další interakce s kategoriemi, které mají stoupající trend tento měsíc má algoritmus šanci vybrat kategorii náhodným výběrem s vahou (**UC4**). Tyto výsledky, tedy dle mého názoru, splňují očekávání.

Pro stejné případy užití jako jsou popsány výše, jsem ověřil funkcionalitu druhého API endpointu pro ohodnocení vhodnosti sbírky. A proto jsou použity guid produktů z předchozího testu. Test tohoto endpointu byl proveden hned po provedení testu daného UC pro endpoint /predict. V případě

prvního UC
(na HTPP požadavek
/usersuit?itemguid=58c04c67-5bcb-11e9-beb1-002590dad85e&month=1&userid=61&price=150)
algoritmus vrátil výsledek:

```
[1,[false,false,false,true],["Káva"]]
```

Pro **UC2**
(/usersuit?itemguid=76e8e97c-b4d5-11e7-ae76-0cc47a6c92bc&month=1&userid=61&price=150):

```
[1,[false,false,true,false],["Vína","Vína","Vína"]]
```

Pro **UC3**
(/usersuit?itemguid=76e8e97c-b4d5-11e7-ae76-0cc47a6c92bc&month=1&userid=61&price=150):

```
[1,[false,false,true,false],["Vína","Vína","Vína"]]
```

Algoritmus tedy v tomto případě měl vždy jednu hodnotu „true“. V prvním případě je kategorie ve stoupajícím trendu pro daný měsíc a v ostatních jde o výskyt v uživatelských statistikách. Vždy jde jen o pouze jednu hodnotu, součet tedy odpovídá (je roven jedné). V případě UC2 a UC3 je navráceno pole s více hodnotami „Vína“. To je zapříčiněno tím, že daný produkt byl v e-shopu zařazen do více kategorií, ty však na Heurece odpovídají právě jedné kategorii, a tou je kategorie „Vína“.

Další use-case je následující:

5. UC 10 impresí, vytvoření 2 kolekce Káva (user id 62) a 10 impresí a 2 kolekce Vína (user id 61) – vždy jde o **/predict?month=1** a **&userid=61** nebo **&userid=62** ...

Tyto podmínky mají za následek následující:
Výsledek pro dvě volání s user id 61:


```
{"category":"Vína",
"product":{"name_heureka":"Vína",
"guid":"142e8af7-bacc-11e6-968a-0cc47a6c92bc",
"name":"Vína",
"eshop_id":"bc915f31-7ee7-4abe-a826-be19be8879fb",
"price":134,"price_per_action":0}}

{"category":"Vína",
"product":{"name_heureka":"Vína",
"guid":"142e8af7-bacc-11e6-968a-0cc47a6c92bc",
"name":"Vína",
"eshop_id":"bc915f31-7ee7-4abe-a826-be19be8879fb",
"price":134,"price_per_action":0}}
```

Výsledek pro tři volání s user id 62:


```
{"category":"Čokolády",
"product":{"name_heureka":"Čokolády",
"guid":"bb73ceeb-c7b0-11e8-a8f6-002590dad85e",
"name":"Vánoční fair trade zboží",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":126.09,"price_per_action":0}}

{"category":"Káva",
"product":{"name_heureka":"Káva",
"guid":"b9b44398-5bcc-11e9-beb1-002590dad85e",
"name":"Káva",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":128.7,"price_per_action":0}}

{"category":"Káva",
"product":{"name_heureka":"Káva",
"guid":"b9b44398-5bcc-11e9-beb1-002590dad85e",
"name":"Káva",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":128.7,"price_per_action":0}}
```


V datech uvedených výše je vidět, že algoritmus vrací skutečně přizpůsobené výsledky podle toho, na co uživatel zareagoval. Vína „přebijí“ kategorii Káva, která je jednak obsažena v globálních uživatelských statistikách, ale zároveň je i nejčastější kategorií pro měsíc leden - viz graf 10.9 na straně 69.

6. UC třetí uživatel s podmínkami z UC5


```
{"category":"Káva",
"product":{"name_heureka":"Káva",
"guid":"af40f3da-6ce6-11e8-8216-002590dad85e",
"name":"Káva bez kofeinu",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":141.74,"price_per_action":0}}

{"category":"Příslušenství ke společenským hrám",
"product":{"name_heureka":"Příslušenství ke společenským hrám",
"guid":"9d93342e-338e-11ea-aa82-ecf4bbd76e52",
"name":"Doplňky","eshop_id":"5b46fc12-2799-46e1-9eda-366327a998aa",
"price":89,"price_per_action":0}}

{"category":"Káva",
"product":{"name_heureka":"Káva",
```

```
"guid":"28526029-5bcc-11e9-beb1-002590dad85e",
"name":"Zrnková káva",
"eshop_id":"3da626b3-8d51-4226-8ea5-aa3e21e829ab",
"price":126.09,"price_per_action":0}}
```


Algoritmus nyní ví, že nemá k dispozici žádná uživatelskou statistiku, ale existují nějaké globální statistiky (některý z uživatelů na ní reagoval) ke kategorii Káva, která má stoupající trend tento měsíc — viz grafy 10.9 na straně 69 a 10.13 na straně 72.

7. UC predikce pro třetího usera pro měsíc srpen

Pro měsíc srpen a třetího uživatele při třech voláních jsem dostal následující výsledek:


```
{"category":"Stolní hry",
"product":{"name_heureka":"Stolní hry",
"guid":"7ce1637e-c00a-11e9-beb1-002590dad85e",
"name":"Deskové hry",
"eshop_id":"5b46fc12-2799-46e1-9eda-366327a998aa",
"price":199,"price_per_action":0}}

{"category":"Ostatní fitness nářadí",
"product":{"name_heureka":"Ostatní fitness nářadí",
"guid":"3e7ec34b-95fa-11e5-b6d1-ac162d8a2454",
"name":"Kotevní materiál",
"eshop_id":"86b7a15e-1fe3-4ea0-b575-bfd33510ac6a",
"price":198,"price_per_action":0}}

{"category":"Stolní hry",
"product":{"name_heureka":"Stolní hry",
"guid":"ef35c289-ff1a-11e9-beb1-002590dad85e",
"name":"Deskové hry",
"eshop_id":"5b46fc12-2799-46e1-9eda-366327a998aa",
"price":229,"price_per_action":0}}
```


V případě změny měsíce z ledna na srpen algoritmus silně využívá data pro třetího uživatele, který nemá statistiky a zároveň ani neexistují statistiky globální – tedy žádná interakce s kategoriemi, které mají stoupající trend v tento měsíc. Algoritmus v tomto případě použije historická data — náhodný výběr s vahou podle průměrného počtu výskytů v objednávkách — viz opět graf 10.9 na straně 69.

Chování predikcí je tedy přesně podle popsaného algoritmu zobrazeného na diagramu 11.3 na straně 95.

Poslední testy, které uvedu jsou pro endpoint /usersuit – konkrétně šlo o postupně měnící se podmínky pro následující HTTP požadavek:

/usersuit?itemguid=ad430c0c-6ce6-11e8-8216-002590dad85e &month=1&userid=61&price=1

Ohodnocení a postupně měnící se podmínky jsou uvedeny v tabulce 13.1 na straně 106.

Podmínky	Výsledek
ad_stats,user_search_stats, interval trend up, kategorie trend up	[4,[true,true,true,true],[”Káva”]]
user_search_stats, interval trend up, kategorie trend up	[3,[true,true,false,true],[”Káva”]]
interval trend up, kategorie trend up	[2,[true,false,false,true],[”Káva”]]
cena 100000000	[1,[false,false,false,true],[”Káva”]]

Tabulka 13.1: Test endpointu /usersuit

První řádek tabulky představuje podmínky, kdy uživatel má záznam v ad_stats, user_search_stats, cena je v cenovém intervalu, který má stoupající trend, totéž kategorie produktu.

V případě druhého řádku již neexistuje záznam v entitě ad_stats.

V dalším případdě neexistuje záznam ani v entitě user_search_stats.

A konečně poslední řádek, kdy již ani cena není v cenové hladině, která by měla stoupající trend. Poslední ilustrací (testem) buď následující případ, kdy pro HTTP požadavek nevyhovuje nic — příkladem buď request : **/usersuit?itemguid=d7ae4fd0-5987-11ea-8cfc-0cc47a6c9c84 &month=1&userid=61&price=100000000**
s následujícím výsledkem:

```
[0,[false,false,false,false],["Tvrzená skla pro mobilní telefony"]]
```

I tento endpoint tedy správně reflektuje algoritmus popsaný v kapitole 11.5.4 na straně 97.

Kapitola 14

Náklady a přínosy

Náklady spojené s popsaným řešením v této práci jsou minimální. Implementace využívá databázového systému, který již v celé aplikaci existuje — stačí do něj přidat další databázi. Obdobně je tomu v algoritmické části. Práce pracuje s Docker kontejnery, které opět již existovaly před touto prací — až na jeden, ve kterém běží Python s příslušnými rozšiřujícími balíčky. Výkonnostně je Python samozřejmě pomalejší nežli například C++ a celkově potřebuje více HW zdrojů. Nicméně vzhledem k tomu, že aplikace již využívá PostgreSQL a Node.js předpokládám, že tyto technické nároky na výkon jsou irelevantní a především již předem splněné.

14.1 Teoretická cena práce

Mezi náklady patří i cena vývoje této práce. Cena je v tomto případě pouze teoretická, ale může ukazovat kolik financí projekt ušetřil na vývoji. Na práci jsem pracoval cca od poloviny ledna 2020 a předpokládám práci dokončit cca v půlce května. V tomto období je **84 pracovních dnů**. V případě, že budu uvažovat plat JavaScript vývojáře – ten se pohybuje dle platy.cz kolem 50 000,- Kč měsíčně, pokud budu uvažovat 20 pracovních dnů, vychází cena ze jeden man day 2 500,- Kč. **Výsledná hrubá mzda by tedy teoreticky podle dat výše byla 210 000,- Kč**. **Pokud připočítáme náklady zaměstnavatele**, tak reálná cena, kterou by zaplatilo Wowee, by se počítala dle následujícího: Z hrubého platu zaměstnance se stane 66 900,- Kč, tudíž za man day by se jednalo o 3 345,- Kč, **za 84 dní by celková suma tedy byla 280 980,- Kč**[42].

14.2 Přínosy a další využití

Přínosy této práce spočívají zejména ve finančním zisku pro Wowee z uživatelských akcí skrze zobrazenou reklamu. Záleží tedy na konkrétním e-shopu,

jakou částku je ochoten za dané akce zaplatit. Vzhledem k aktuální povaze e-shopů, kde se jedná spíše o malé, či začínající hráče na trhu, dle mého názoru nelze očekávat nějaké velké sumy. Může však jít o příjem, kterým pokryje alespoň provoz služeb.

Tato situace by se radikálně změnila v případě velkých obchodů, jako jsou Alza.cz nebo CZC.cz. V případě, že by se Wowee podařilo získat jako partnera takovýto e-shop, částky by mohly být už skutečně zajímavé[43]. Na druhou stranu by bylo třeba upravit algoritmy tak, aby uvažovaly data správně ze všech zdrojů a „nepřebili“ jej jen velcí hráči. Toto se týká jak uvažování nad časovými řadami, tak v případě samotného algoritmu pro vybrání reklamy.

Další rozšíření, mnou implementovaných algoritmů, by mohla být detekce anomálií v časových řadách, které by mohly naznačovat propagování produktu e-shopem nějakou formou reklamy. Tato data by byla užitečná v exportu pro BI systémy a pro ostatní e-shopy, aby například mohly analyzovat periodičnost a adekvátně reagovat na toto chování.

Mimo využívání nasbíraných interakcí v rámci aplikace Wowee a analýzy časových řad by jako další zdroj dat pro predikci a hodnocení uživatelů mohlo být propojení se sociálními sítěmi. Toto spojení lze ostatně nalézt i v některých známých problémech — viz kapitola 1.3 na straně 7.

Další rozšíření by byl jakýsi administrační panel pro partnery Wowee, ve kterém by mohl být přístup k mnou vytvořeným grafům, nebo by se k tvorbě grafů mohl využít jen implementovaný endpoint pro BI systémy. Jelikož jde již o poměrně vzácná data, je zde příležitost pro další zpoplatnění. Otázkou je, zda-li Wowee chce být takovým nástrojem, nebo se chce specializovat především na svůj primární účel, a to je vybírání příspěvků na sbírky resp. dárky.

Závěr

Cílem práce bylo stanovit algoritmus vybírající vhodný produkt k reklamě pro projekt Wowee a v případě vytvoření sbírky ve stejnojmenném projektu ohodnotit její vhodnost. K těmto účelům bylo zadáno využít data z Shoptet API, přičemž dalším cílem bylo jejich zpracovanou formu nabídnout jako export pro business inteligence systémy.

Teoretická část rozebírá samotné chování zákazníka (resp. uživatele) pro jeho lepší pochopení. Dále obsahuje kapitoly 1.3 a 2, které se zabývají podobností s existujícími problémy a možnostmi řešení.

Praktická část, věnující se analýze, zkoumá získaná data se zjištěními, že řada z možných atributů není e-shopy využívána a z dalších získaných informací usuzuje jako nejlepší využít časové řady.

Algoritmus pro vybírání inzerce reflektuje chování uživatele v aplikaci Wowee dle nasbíraných statistik, které se zapisují díky navrženému API. Podle chování uživatele se tedy mění i to, co vrací mnou navržený algoritmus. Vzdáleně toto řešení tak může připomínat funkci neuronových sítí. Funkce, které byly v předchozím případě použity, se využívají i pro ohodnocení samotných sbírek. Jde de facto o stejné informace, které jsou jen jinak interpretované. Z pohledu praktické části tedy tento bod byl naplněn.

Jedním z dalších požadavků bylo vytvoření exportu pro business inteligence sytémy. Tento požadavek je vyřešen opět skrze příslušný API endpoint. Vše je navrženo s podporu stránkování. Uživatel má tedy poměrně pohodlný přístup k datům a jejich dalšímu zpracování.

Výše popsaný algoritmus vznikl v rámci této práce, využívá historických dat z e-shopů. Data z nich v době psaní práce byla však ovlivněna skladbou e-shopů, které využívají Wowee doplněk pro Shoptet. Často se jedná o malé e-shopy s malými objemy dat, nebo o e-shopy, které jsou na trhu velmi krátce. Podobně jsou na tom některé získané atributy produktů a objednávek, Shoptet jich nabízí celou řadu, ale nezdá se, že by je provozovatelé minimálně těchto e-shopů aktivně využívali.

Pro projekt Wowee a celé navržené řešení by bylo dobré získat některé

velikostně „střední“ popř. velké e-shopy. To by projektu pomohlo mít jednak větší objem dat a především více relevantních výsledků pro stanovení reklamy a i lepší ohodnocené konkrétní sbírky. Algoritmus by se v případě větších e-shopů musel nejspíše z části přizpůsobit, a to tak, aby dostaly příležitost i e-shopy menšího rázu. Nemyslím si, že by se jednalo o nějaké velké změny a velká část principů bude stejná, znovu aplikovatelná. Totéž se týká navržené databáze a analytického nástroje.

Jedna z možností rozšíření je vytvoření prostředí v rámci Wowee Partner API, které by se více chovalo jako analytický nástroj podobný business inteligence systémům. To by otevřelo brány Wowee do úplně nového segmentu.

Literatura

[1] Šanda, M.: *Analýza dopadů GDPR v oblasti webových služeb*. Bakalářská práce, České vysoké učení technické, Fakulta Informačních Technologií, Praha, 2018.

[2] Balajka, R.: Exkluzivní report o vyhledávání z 500 ecommerce projektů. [cit. 27.2.2020]. Dostupné z: `https://www.ecommercebridge.cz/exkluzivni-report-o-vyhledavani-z-500-ecommerce-projektu/`

[3] Vysekalová, J.: *Chování zákazníka: jak odkrýt tajemství "černé skříňky"*. Praha: Grada, první vydání, 2011, ISBN 9788024735283.

[4] Hroch, M.; Cach, P.: Business intelligence staví na datovém skladu. [cit.23.3.2020]. Dostupné z: `https://www.systemonline.cz/business-intelligence/business-intelligence-stavi-na-datovem-skladu.htm`

[5] Hajn, P.: Business intelligence jako rozšíření ERP systémů stavebních firem. [cit. 11.3.2020]. Dostupné z: `https://www.systemonline.cz/business-intelligence/business-intelligence-jako-rozsireni-erp-systemu-stavebnich-firem.htm`

[6] Pilous, J.: Trendy business intelligence. [cit. 23.3.2020]. Dostupné z: `https://www.systemonline.cz/business-intelligence/trendy-business-intelligence.htm`

[7] Krstanov, Z.: Bez investice až do USA. Česká Keboola teď dobývá svět firemních dat z Chicaga. Dostupné z: `https://www.forbes.cz/bez-investice-az-do-usa-ceska-keboola-chce-dobyt-svet-firemnich-dat-z-chicaga/`

[8] Keboola: Product. [cit. 25.3.2020]. Dostupné z: `https://www.keboola.com/product/`

[9] Cibulka, J.: Stát se k informacím o pohybu lidí nedostane, říká jeden z tvůrců chytré karantény. https://bit.ly/3eBhPkE, [cit. 21.4.2020].

[10] Špaček, M.: Vyjádření k chytré karanténě. [cit. 21.4.2020]. Dostupné z: https://www.facebook.com/spaze/posts/10220860780645895

[11] Microsoft: Microsoft PowerBI. [cit. 25.3.2020]. Dostupné z: https://powerbi.microsoft.com/cs-cz/

[12] Microsoft: Power Apps. [cit. 25.3.2020]. Dostupné z: https://powerapps.microsoft.com/cs-cz/

[13] Microsoft: Power Automate. [cit. 25.3.2020]. Dostupné z: https://flow.microsoft.com/cs-cz/

[14] Sadollah, A.; Gao, K.; Barzegar, A.; aj.: Improved model of combinatorial Internet shopping optimization problem using evolutionary algorithms. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Nov 2016, ISSN null, s. 1–5, doi: 10.1109/ICARCV.2016.7838660.

[15] Musial, J.; Pecero, J.; López Locés, M.; aj.: Algorithms solving the Internet shopping optimization problem with price discounts. *Bulletin of the Polish Academy of Sciences Technical Sciences*, ročník 64, 09 2016, doi:10.1515/bpasts-2016-0056.

[16] Salehi, M.; Kamalabadi, I. N.: A Hybrid Recommendation Approach Based on Attributes of Products Using Genetic Algorithm and Naive Bayes Classifier. *Int. J. Bus. Inf. Syst.*, ročník 13, č. 4, Červenec 2013: str. 381–399, ISSN 1746-0972, doi:10.1504/IJBIS.2013.055297. Dostupné z: https://doi.org/10.1504/IJBIS.2013.055297

[17] Jonák, Z.: Heuristika. Dostupné z: https://aleph.nkp.cz/F/?func=direct&doc_number=000002767&local_base=KTD

[18] Quora: Heuristika. [cit. 10.3.2020]. Dostupné z: https://stats.stackexchange.com/questions/300350/is-machine-learning-an-heuristic-method

[19] Klouda, K.; Vašata, D.; Kovalenko, A.: MI-PDM přednáška 2 - Rozhodovací stromy. [cit. 15.4.2020]. Dostupné z: https://courses.fit.cvut.cz/MI-PDM/lectures/files/MI-PDM-02-cs-slides.pdf

[20] Yadav, M.: Understanding Data Attribute Types | Qualitative and Quantitative. [cit. 16.3.2020]. Dostupné z: https://www.geeksforgeeks.org/understanding-data-attribute-types-qualitative-and-quantitative/

[21] Demel, J.: *Grafy a jejich aplikace*. Libčice nad Vltavou: J. Demel, vyd. 2., (vlastním nákladem 1.) vydání, 2015, ISBN 8026076842;9788026076841;.

[22] scikit-learn developers: Decision Trees. [cit. 27.3.2020]. Dostupné z: https://scikit-learn.org/stable/modules/tree.html

[23] Korotyaev, D.: What are the differences between ID3, C4.5 and CART? [cit. 8.3.2020]. Dostupné z: https://www.quora.com/What-are-the-differences-between-ID3-C4-5-and-CART

[24] scikit-learn developers: Ensemble methods. [cit. 20.3.2020]. Dostupné z: https://scikit-learn.org/stable/modules/ensemble.html

[25] Klouda, K.; Vašata, D.; Kovalenko, A.: MI-PDM přednáška 3 - Ensemble metody. [cit. 3.3.2020]. Dostupné z: https://courses.fit.cvut.cz/MI-PDM/lectures/files/MI-PDM-03-cs-slides.pdf

[26] Wrhel, V.: *Rozpoznávání vzorů v obraze pomocí AdaBoost*. Diplomová práce, Vysoké učení technické v Brně, Fakulta Informačních Technologií, Brno, 2010.

[27] Šochman, J.: Cvičení z RPZ – AdaBoost. [cit. 21.4.2020]. Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/recognition/Labs/adaboost/adaboost.pdf

[28] Zelinka, I.: *Umělá inteligence v problémech globální optimalizace*. Praha: BEN - technická literatura, první vydání, 2002, ISBN 8073000695;9788073000691;.

[29] Pošík, P.: Genetické algoritmy. [cit. 20.2.2000]. Dostupné z: http://labe.felk.cvut.cz/~posik/pga/theory/ga-theory.htm

[30] Litschmannová, M.: Úvod do analýzy časových řad. [cit. 27.3.2020]. Dostupné z: https://homel.vsb.cz/~lit40/SMAD/Casove_rady.pdf

[31] Czibor, F.: *E-shop pro prodej stavebních materiálů*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta Informačních Technologií, Brno, 2007. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=115778

[32] Šmíd, V.: Životní cyklus informačního systému. [cit. 9.5.2020]. Dostupné z: https://www.fi.muni.cz/~smid/mis-zivcyk.htm

[33] Pittet, S.: What are the differences between continuous integration, continuous delivery, and continuous deployment? [cit. 24.4.2020]. Dostupné z: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

[34] Vitvar, T.: Lecture 3: Representational State Transfer. [cit. 27.4.2020]. Dostupné z: http://mdw.vitvar.com/pdf/lecture3-1p.pdf

[35] Docker, Inc.: Reference documentation. [cit. 17.4.2020]. Dostupné z: https://docs.docker.com/reference/

[36] OpenJS Foundation: About Node.js. https://nodejs.org/en/about/, [cit. 24.4.2020].

[37] StrongLoop, IBM, and other expressjs.com contributors: Hello world example. [cit. 17.4.2020]. Dostupné z: https://expressjs.com/en/starter/hello-world.html

[38] Shoptet: Import produktů. [cit. 25.3.2020]. Dostupné z: https://podpora.shoptet.cz/hc/cs/articles/360003141512-Import-produktů

[39] Shoptet: Nová políčka v detailu objednávky a sekce na vyžádání. [cit. 1.4.2020]. Dostupné z: https://developers.shoptet.cz/nova-policka-v-detailu-objednavky-a-sekce-na-vyzadani/

[40] Matlis, J.: Ajax. [cit. 17.4.2020]. Dostupné z: https://computerworld.cz/archiv/ajax-24537

[41] Shoptet: Základy práce s API. [cit. 16.3.2020]. Dostupné z: https://shoptet.docs.apiary.io/introduction/zaklady-prace-s-api/jak-volat-api

[42] KELOC CS, s.r.o.: Mzdová kalkulačka pro rok 2020. [cit. 5.5.2020]. Dostupné z: https://www.keloc-software.cz/mzdova-kalkulacka/

[43] Shewan, D.: The Comprehensive Guide to Online Advertising Costs. [cit. 5.5.2020]. Dostupné z: https://www.wordstream.com/blog/ws/2017/07/05/online-advertising-costs

Příloha A

Seznam použitých zkratek

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

BI Business Intelligence

BI-SI1 předmět Softwarové inženýrství 1 bakalářského programu

BI-SP1/2 předmět Softwarový projekt 1/2 bakalářského programu

CART Classification and regression tree

CD Continous Delivery

CI Continous Integration

CPA Cost Per Action

CRUD Create,Remove,Update,Delete

ČR Česká republika

DWH Data Warehouse

EAN European Article Number

ETL Extract Transform Load

GDPR General Data Protection Regulation

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

ID3 Iterative Dichotomiser 3

ISBN International Standard Book Number

IT Informační technologie

JSON JavaScript Object Notation

KPI Key Performance Indicators

MI-MDW předmět Webové služby a middleware magisterského programu

ORM Object-relational-mapping

OS Operační systém

PPC Pay Per Click

PSČ Poštovní směrovací číslo

REST Representational State Transfer

SDLC Systems development life cycle

SLA Service Level Agreement

UC Use Case

UML Unified Modeling Language

URI Uniform Resource Identifier

VC Voting Clasifier

XML Extensible Markup Language

Příloha B

Přehled četností pro kategorie

List Data

Přehled četnosti výskytů kategorií v objednávkách

Legenda:

...velmi malý počet dat nebo žádná

...malý počet dat

...dostatečný počet dat

pro české eshopy

sloupec cnt = celkový počet výskytu katgorie,jak pro slovenské tak české eshopy

name_heureka	cnt	Dny	Měsíce
Access pointy, routery	2		
Aminokyseliny	2		
Aromaterapie	118		
Autokosmetika	101		
Baterie do e-cigaret	12		
Baterie primární	23		
Baterie pro mobilní telefony - neoriginální	12		
Batohy	26		
Bicykle	47		
Blatníky	267		
Bovdeny a lanka	136		
Brzdy	401		
Bylinné čaje	165		
Cestovná batožina	2		
Cyklistické brašny	43		
Cyklistické brýle	2		
Cyklistické bundy a vesty	15		
Cyklistické dresy	33		
Cyklistické helmy	12		
Cyklistické kalhoty	35		
Cyklistické príslušenstvo	52		
Cyklistické příslušenství	563		
Cyklistické rukavice	44		
Cyklistické tašky	74		
Cyklistické tretry	16		
Cyklistické zvonky	8		
Čaje	174		
Čelenky	3		
Čelovky	6		
Čistící tablety do kávovarů	60		

Stránka 1

List Data

Čokoládové tyčinky	742
Čokolády	1963
Dámská body	11
Dámská obuv	114
Dámská trička	10
Dámské bundy a kabáty	1
Dámské kalhoty	5
Dámské ponožky	43
Dámské šortky	53
Dekorační polštáře	1
Dětská kosmetika	36
Dětská obuv	60
Dětské bundy a kabáty	137
Dětské deky	2
Dětské kalhoty	125
Dětské ponožky	46
Dětské povlečení	5
Dětské rukavice	22
Dětské spodní prádlo	50
Dojčenské fľaše	22
Doplňky stravy	348
Držiaky na mobil	15
Duše	609
Gripy a omotávky	136
Hlavolamy	13
Hlavové zloženia	49
Hodinky	20
Hodiny	3
Hustilky	116
Inteligentné hodinky	21
Interaktivní hračky	4
Iontové nápoje	4
Káva	8347
Kazety	21
Kľuky	38
Knihy	228
Kojenecké čepice, rukavice a šály	40
Kojící polštáře	11
Košíky na fľaše	222
Košíky na lahve	26
Kravaty a motýlky	368
Kryty na mobilné telefóny	4457
Kuchyňské oleje	8
Masážní pomůcky	1023
Masážní přípravky	16
Mikiny a svetry	315
Mýdla	4
Nabíjačky pre mobilné telefóny	30
Náboje	80
Náhrdelníky	200
Nákupní tašky a košíky	14
Náradie	185
Náramky	27

List Data

Nářadí	62
Náušnice	64
Návleky	13
Nosiče detí	38
Nosiče na bicykel	132
Nosiče na kolo	82
Nože	36
NULL	0
Nutriční doplňky	22
Obrazy	59
Oleje, vazelíny, čističe	83
Osiva a semínka	3
Ostatné stavebnice	16
Ostatní fitness nářadí	1023
Ostatní společenské hry	3
Outdoor láhve	12
Ozdobné prvky	1
Pánská obuv	48
Pánská trička	19
Pánské bundy a kabáty	1
Pánské kalhoty	6
Pánské mikiny	146
Pánské ponožky	167
Pánské spodky	17
Pánské šortky	53
Parfémy	9
Pedále	83
Pedály	78
Pláště	253
Pláště	19
Pláštěnky	5
Pleny	7
Pleťová séra a emulze	2
Pleťové krémy	7
Pleťové oleje	5
Pouzdra na mobilní telefony	18
Powerbanky	1
Predstavce	6
Príbory	1
Prívesky	5
Pro nejmenší	1
Prostěradla	5
Proteiny	41
Prstene	12
Přehazovačky	6
Přípravky do myčky	1
Přípravky na čištění pleti	5
Přípravky na vrásky a stárnoucí pleť	11
Příslušenství ke společenským hrám	1296
Příslušenství pro e-cigarety	16
Puzdrá na mobilné telefóny	62
Ráfky	1
Rámy	5

Stránka 3

List Data

Reflexní pásky a klipy	9
Rohy	3
Ručníky	7
Řetězy	21
Řídítka	5
Sedačky a vozíky	8
Sedlovky	118
Slúchadlá	27
Speciální péče o pleť	29
Sprchové gely	8
Stimulanty a energizéry	14
Stojany na jízdní kola	8
Stolní hry	10367
Stredové osy a zloženie	58
Svetlá na bicykel	228
Světla na kolo	45
Šátky	3
Športtestery a computery	14
Tělové balzámy	24
Tělové oleje	29
Tělové peelingy	5
Tlmiče	1
Trička a košile	4
Tvrdené sklá pre mobilné telefóny	94
Tvrzená skla pro mobilní telefony	48
Údržba a čištění obuvi	3
Univerzální čistící prostředky	4
USB káble	49
Vánoční dekorace	3
Vidlice	15
Vína	8241
Vitamíny a minerály	160
Vonné oleje	101
Zámky na bicykel	112
Zámky na kolo	4
Zapaľovače	43
Zapletené kolesá	74
Zavinovačky	38
Zimní čepice	12
Žehličky vlasů	1

Příloha C

Obsah přiloženého CD

```
CD
├── readme.txt ..... stručný popis obsahu CD
├── text ..... adresář s prací ve formátu PDF
│   └── DP_Šanda_Michal_2020.pdf
└── src
    ├── impl ..... zdrojové kódy implementace
    │   ├── python-dpTool ..... analytický nástroj v Pythonu
    │   │   └── ...
    │   └── papi_modules
    │       ├── PredictionDataDownloader . node.js modul pro migraci dat
    │       │   └── ...
    │       └── Predictor ..... node.js modul pro predikci a ohodnocení
    │           └── ...
    └── thesis ..... Zdrojová forma práce ve formátu LaTeX
```