FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Algoritmy pro odvození vah produktů na základě vah celých objednávek s chybovostí
Student:	Michael Olšavský
Vedoucí:	Ing. Martin Horský
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Cílem práce je vytvoření nástroje, který z dat o vahách objednávek (zjištěných vahami na dopravníku) odhadne předem neznámé váhy jednotlivých produktů v objednávkách obsažených. Je třeba počítat s tím, že obsah některých objednávek je chybný.

1. Formulujte matematicky zadaný problém a proveďte rešerši existujících řešení. Vezměte v úvahu, že výsledky samotných vážení v sobě mohou obsahovat malé odchylky a stejně tak váhy jednotlivých produktů.
2. Implementujte vybrané metody řešení a případně navrhněte vlastní přístup.
3. Navrhněte též metodologii vyhodnocování úspěchu dané implementace.

Elektronicky schválil/a Ing. Karel Klouda, Ph.D. dne 13. ledna 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Algoritmy pro odvození vah produktů na základě vah celých objednávek s chybovostí

Michael Olšavský

Katedra aplikované matematiky
Vedoucí práce: Ing. Martin Horský

11. května 2021

Poděkování

Chtěl bych poděkovat Mgr. Karlu Tesařovi za podnětné konzultace a Ing. Martinu Horskému za vedení práce.

Také bych rád poděkoval své přítelkyni za její neustálou podporu nejen při psaní práce, ale po celé studium.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2021 . . . . . .

Odkaz na tuto práci

Olšavský, Michael. *Algoritmy pro odvození vah produktů na základě vah celých objednávek s chybovostí*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato práce se zabývá výpočtem vah produktů z naměřených vážení celých objednávek, jejichž obsah je předem známý. Úloha je přeformulována na hledání řešení soustavy lineárních rovnic s náhodnými veličinami jakožto proměnnými. Důraz je kladen na navržení robustního řešení schopného vypořádat se s chybami v datech. Vedle metod matematické optimalizace jako lineární programování a metoda nejmenších čtverců navrhujeme i jedno vlastní řešení. Všechny metody jsou porovnány na testovacích datech vygenerovaných na základě reálných měření. Výstupem této práce je prototyp aplikace řešící tuto úlohu s dostatečnou úspěšností pro zavedení nových skladových procesů jako kontrolní vážení na dopravníku zajišťující kontrolu kvality.

Klíčová slova odhad váhy produktů, kontrola váhy, systém lineárních rovnic s chybou, přibližné řešení přeurčeného lineárního systému, skladový systém, wms, optimalizace skladových procesů

Abstract

In our thesis, we explore methods for calculating the weight of individual products from weight measurements of entire orders, of which the content is known. The problem is formulated as finding a solution to a linear system with random variables. Emphasis is placed on designing a robust solution capable of dealing with data errors. We propose our custom solution and several optimization methods, such as linear programming and the least-squares method. All methods are compared on generated data based on real measurements. We present a prototype solving this task with sufficient accuracy for introducing new warehouse processes such as check weigher machines as a part of quality control.

Keywords product weight estimation, check weigher, system of linear equations with error, approximate solution to overdetermined linear system, warehouse management system, wms, warehouse process optimization

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Skladový systém (Warehouse Management System, dále jen WMS) je souhrnný pojem pro software zajišťující běh skladu, a to od evidence a naskladňování zboží, vyřizování objednávek, reporting až po plánování konkrétní práce v jednotlivých částech skladu.

Kvalitní WMS jsou navrhovány s cílem optimalizovat skladové procesy – může jít například o snížení času potřebného ke zpracování objednávky, efektivní rozmístění a uchování zboží, snížení nákladů na zpracování objednávky či zavedení automatizace s cílem snížit počet pracovníků nutných k obstarání stejného objemu zboží.

Nutnou součástí procesu zpracování objednávky je kontrola kvality, tedy kontrola, zda se v objednávce vyskytuje správný typ a kvantita zboží, a zda je zboží v adekvátním stavu.

Jedním z průmyslových standardů kontroly kvality zboží (resp. celých objednávek) je kontrolní váha (tzv. *checkweigher*). Jedná se o zařízení umístěné na dopravním páse, které váží položky na něm umístěné a naměřenou váhu porovnává s očekávanou váhou. Při správném nastavení dalších procesů může takové zařízení razantně snížit čas potřebný ke zpracování objednávky a počet zaměstnanců nutných k vykonávání manuální kontroly kvality.

Pro zavedení takové kontroly je ale nutné mít velmi přesné informace o váze zboží, které se ve skladu vyskytuje. To může být prakticky nezajistitelný předpoklad pro sklady s velmi dynamickým počtem klientů s malou mírou kontroly nad přijímaným zbožím (typicky jde o sklady zajišťující tzv. *e-commerce order fulfillment*, tedy distribuci zboží pro eshopy třetí strany).

Znalost váhy produktů umístěných ve skladu se ale dá využít i v dalších procesech – příkladem může být volba vhodného způsobu dopravy či obalového materiálu.

Tato práce se zabývá extrakcí informace o váhách jednotlivých produktů z historických měření vah objednávek získaných ze zmíněné kontrolní váhy. Následné využití této informace je již mimo rozsah práce.

Kapitola 1

Cíl práce

Cílem teoretické části práce je matematicky definovat problém extrakce informace o váhách jednotlivých produktů z vážení objednávek, jejichž očekávaný obsah známe, a navrhnout metody, které tento problém řeší. Součástí definice problému je také analýza dostupných dat, na základě které bude možné vytvořit hypotézy o očekávané kvalitě navržených metod řešení.

V praktické části je pak cílem implementovat a porovnat navržené metody na vygenerovaných testovacích datových sadách, která svou strukturou odpovídají reálným měřením zaznamenaných kontrolní váhou. Výstupem tohoto porovnání bude nástroj, který na základě vstupních parametrů identifikuje nejvhodnější metodu a vypočítá odhady vah produktů.

V celé práci je třeba pracovat s existencí náhodných, ale i systematických chyb v naměřených datech a s tím, že váhy produktů mají nenulový rozptyl.

Kapitola 2

Analýza problému

Jak bylo řečeno v kapitole 1, problémem, kterým se tato práce zabývá, je odvození vah jednotlivých produktů ze znalosti váhy a obsahu objednávek. Jelikož jde o velmi specifický problém, relevantní literatura prakticky neexistuje. Cílem této kapitoly je tedy celou problematiku formálně definovat a zobecnit.

2.1 Popis problémové domény

Na základě prezentovaného problému v kapitole 1 definujeme konkrétní objekty, které následně využijeme k matematické formulaci úlohy.

2.1.1 Produkt

Produkt P_i je reprezentován pouze svou váhou, tj. náhodnou veličinou příslušící nějakému normálnímu rozdělení $N(\mu_i, \sigma_i^2)$.[1]

p-tici všech produktů P tedy definujeme následovně:

$$i \in \{1, 2, \ldots, p\} : \; P_i \sim N(\mu_i, \sigma_i^2) \tag{2.1}$$

$$P = (P_1, P_2, ..., P_p) \tag{2.2}$$

kde p je celkový počet produktů.

2.1.2 Objednávka

Objednávku O_i popisuje p-rozměrný vektor, jehož j-tá složka reprezentuje počet kusů produktu P_j v dané objednávce. Pro zjednodušení předpokládejme, že objednávky se stejným obsahem jsou reprezentovány stejným objektem nehledě na další reálné proměnné jako cílový zákazník či datum objednání.

[1] Váha produktu samozřejmě může nabývat pouze hodnot z intervalu $(0; \infty)$, nicméně vzhledem k povaze dat, kdy je rozptyl násobně menšího řádu než váha samotná, je pravděpodobnost získání záporné váhy z takového rozdělení prakticky nulová.

$$i \in \{1, 2, ..., o\} : O_i = (q_{i,1}, q_{i,2}, \ldots, q_{i,p}), O_i \in \mathbb{N}_0^p \tag{2.3}$$

kde o je celkový počet objednávek a $q_{i,j}$ tedy značí počet kusů produktu P_j v objednávce O_i.

2.1.3 Váha objednávky

(Teoretická) váha W_i objednávky O_i je součtem vah produktů v ní obsažených.[2] Jedná se tedy o součet nezávislých normálně rozdělených náhodných veličin. Pro n normálně rozdělených nezávislých náhodných veličin $X_1, X_2, \ldots, X_n$ se středními hodnotami $\mu_1, \mu_2, \ldots, \mu_n$ a rozptyly $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$ platí [1]

$$\sum_1^n c_i X_i \sim N(\sum_1^n c_i \mu_i, \sum_1^n c_i^2 \sigma_i^2) \tag{2.4}$$

(Teoretická) váha objednávky je tedy definována jako:

$$\psi_i = O_i \cdot P^T \tag{2.5}$$

Reálná váha objednávky O_i bude zatížena náhodnou chybou měření (způsobenou např. odchylkou měření váhy). Pro zjednodušení předpokládejme, že váha je pravidelně odborně kalibrována a nedochází tak k systematické změně vážení v čase. V takovém případě je chyba měření $\varepsilon \sim N(0, \sigma^2)$, kde σ je střední odchylka váhy udávaná výrobcem.

Očekávaná váha objednávky O_i (dále jen „Váha objednávky“) je pak definována jako:

$$\begin{aligned} W_i &= O_i \cdot P^T + \varepsilon \\ E[\varepsilon] &= 0 \end{aligned}$$

2.1.4 Vážení

Vážení objednávky O_i je konkrétní realizací náhodného výběru z rozdělení W_i. Každá objednávka může být zvážena libovolně krát.

$$Y_{i,j} : j\text{-té vážení objednávky } O_i$$

Vektor Y pak označuje uspořádanou k-tici těchto měření.[3]

[2] Při vážení je objednávka typicky umístěna v nějaké přepravce. Buď se jedná o standardizované přepravky a jejich váhu lze odečíst již při přípravě dat nebo lze přepravku považovat za další produkt objednávky. V našem případě byly přepravky standardizované.

[3] Ačkoliv dolní index naznačuje indexaci ve dvourozměrné matici, jde pouze o předání informace k jaké objednávce měření přísluší. Uspořádaná dvojice (a, b) z dolního indexu pak jednoznačně identifikuje pořadí měření v matici Y.

2.1.5 Úloha

Z naměřených dat máme k dispozici vektor vážení Y a k tomu matici korespondujících objednávek $\boldsymbol{Q}$.

$$\begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,p} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,p} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,p} \\ q_{1,1} & q_{1,2} & \cdots & q_{1,p} \\ q_{3,1} & q_{3,2} & \cdots & q_{3,p} \\ q_{k,1} & q_{k,2} & \cdots & q_{k,p} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{pmatrix} + \begin{pmatrix} \varepsilon_{1,1} \\ \varepsilon_{2,2} \\ \varepsilon_{2,2} \\ \varepsilon_{1,2} \\ \varepsilon_{3,1} \\ \varepsilon_{k,1} \\ \vdots \end{pmatrix} = \begin{pmatrix} Y_{1,1} \\ Y_{2,1} \\ Y_{2,2} \\ Y_{1,2} \\ Y_{3,1} \\ Y_{k,1} \\ \vdots \end{pmatrix} \tag{2.6}$$

Tento systém lineárních rovnic reprezentuje celou řešenou úlohu. Vektor ε v této rovnici reprezentuje jak konkrétní chybu měření způsobenou odchylkou váhy, tak situaci kdy měřená objednávka nemá očekávaný obsah. Stále se v ní však vyskytují náhodné veličiny – váhy produktů. Snahou dále představovaných algoritmů je řešit úlohu příbuznou, ve které se předpokládá, že rozptyl vah produktů je rozumně malý.[4] Hledáme tedy řešení soustavy rovnic 2.7 pro $E[P] \in \mathbb{R}^p$.

$$\begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,p} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,p} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,p} \\ q_{1,1} & q_{1,2} & \cdots & q_{1,p} \\ q_{3,1} & q_{3,2} & \cdots & q_{3,p} \\ q_{k,1} & q_{k,2} & \cdots & q_{k,p} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \begin{pmatrix} E[P_1] \\ E[P_2] \\ \vdots \\ E[P_p] \end{pmatrix} + \begin{pmatrix} \varepsilon_{1,1} \\ \varepsilon_{2,2} \\ \varepsilon_{2,2} \\ \varepsilon_{1,2} \\ \varepsilon_{3,1} \\ \varepsilon_{k,1} \\ \vdots \end{pmatrix} = \begin{pmatrix} Y_{1,1} \\ Y_{2,1} \\ Y_{2,2} \\ Y_{1,2} \\ Y_{3,1} \\ Y_{k,1} \\ \vdots \end{pmatrix} \tag{2.7}$$

Resp. kompaktněji

$$\begin{aligned} \boldsymbol{Q} \cdot E[P] + \boldsymbol{\varepsilon} &= Y \\ \boldsymbol{Q} &\in \mathbb{N}^{m,p} \\ \boldsymbol{\varepsilon}, Y &\in \mathbb{R}^m \end{aligned} \tag{2.8}$$

kde m značí celkový počet provedených měření.

Pokud bychom znali chybu každého měření a rozptyl váhy jednotlivých produktů byl opravdu nulový, vyřešit takovou úlohu by bylo triviální. Tak tomu ale u měření z fyzického světa není.

[4] Pokud by tento předpoklad neplatil, ani při znalosti rozptylu by v takovém případě nebylo možné spolehnout se na výsledek *kontrolního vážení*, jelikož by takový produkt přinášel do kontroly přílišnou míru náhodnosti.

2.2 Analýza dostupných dat

K výběru vhodných metod řešení úlohy definované v kapitole 2.7 bylo nutné se seznámit se strukturou a rozdělením dostupných dat. Výsledky této práce vychází z vybrané podmnožiny měření zaznamenaných na kontrolní váze v jednom ze skladů zadavatele práce v období od 1. 2. 2021 do 6. 4. 2021.

Data byla vyexportována z SQL databáze ve formátu CSV. Z naměřené váhy byla vždy odečtena střední hodnota váhy standardizované přepravky, ve kterém byla objednávka umístěna. Zároveň došlo k odebrání všech takových objednávek, které byly v průběhu průchodu skladem označeny jako problémové či přímo neprošly manuální kontrolou (například z důvodu chybějícího či špatného produktu v objednávce).

Formát dat je následující:

```
measurement_id,weight,product_id,account_id,quantity
00000448-396a-4816-80b6-94df6b9e729b,0.890000,208829,3026,1
00002b9d-a75b-466a-a766-5ee8a8da0c2c,0.660000,159285,3831,1
```

kde (název sloupce, datový typ, popis)

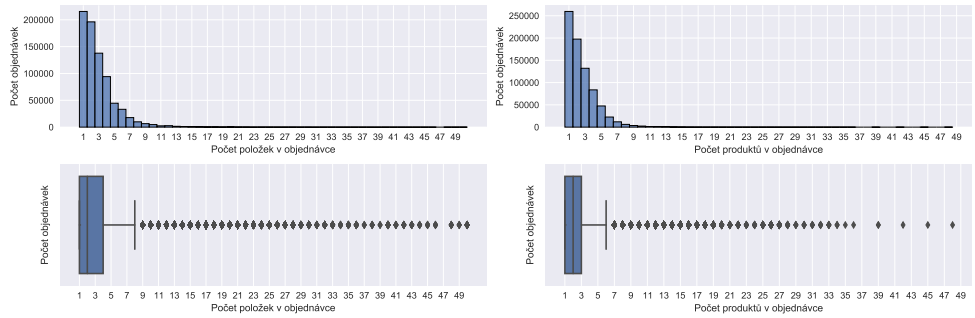
- measurement_id: string - UUIDv4 identifikátor objednávky
- weight: float - změřená váha celé objednávky
- product_id: int - unikátní identifikátor produktu v objednávce
- account_id: int - unikátní identifikátor e-shopu, kterému daná objednávka (i produkt) přísluší
- quantity: int - počet kusů daného produktu v objednávce

Objednávku o k produktech reprezentuje k řádků. Celkově jde o více než 750 tisíc měření, ve kterých se vyskytlo přes 33 000 různých produktů. Přesná čísla jsou k dispozici v tabulce 2.1.

	Celkem
Počet měření	772229
Počet produktů	31113
Počet e-shopů	646

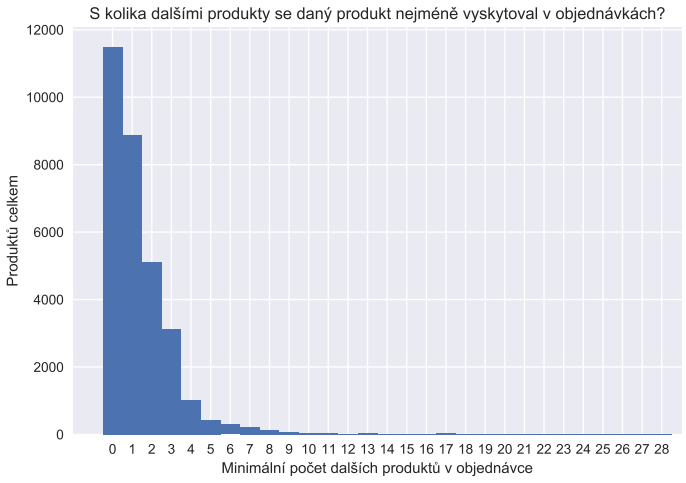
Tabulka 2.1: Dostupná reálná měření

Nejčastější jsou jednokusové objednávky, počet objednávek s vyšším počtem produktů relativně rychle klesá, což znázorňuje graf 2.1.



Obrázek 2.1: Histogram počtu kusů produktu

Z pohledu na takové rozdělení by se mohlo zdát, že pro téměř všechny produkty existuje objednávka, ve které se vyskytoval právě jen konkrétní produkt sám, díky čemuž by bylo základní řešení problému triviální úlohou – stačilo by odstranit odlehlá měření a vypočítat odhad střední hodnoty (např. zprůměrováním). Z grafu 2.2 je ale vidět, že tomu tak není a je tedy třeba zvolit sofistikovanější metody.



Obrázek 2.2: Histogram počtu položek na objednávku

Vzhledem k tomu, že **produkty a objednávky jednotlivých e-shopů jsou navzájem disjunktní**, nemá význam provádět výpočet nad celou datovou sadou, ale pouze na jednotlivých podmnožinách reprezentujících měření daných e-shopů.

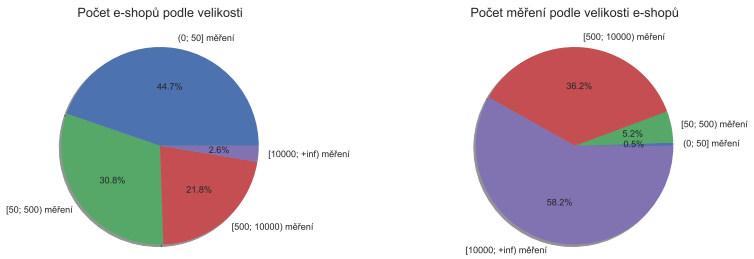
Charakteristiku e-shopů reprezentují tabulka 2.2 a graf 2.3. Jak je vidět, majoritu objednávek reprezentuje pouze 2% největších e-shopů, přičemž nejvíce

měření na e-shop je 59646 a produktů maximálně 3027. To je stále v rozumných mezích i pro algoritmy s kvadratickou časovou i prostorovou složitostí, obzvlášť pokud výpočet může běžet jednotky hodin.

Pokud by ani po rozdělení datové sady na disjunktní podmnožiny dle e-shopů byly jednotlivé podmnožiny na výpočet příliš velké, jistě by bylo možné (alespoň nějakou heuristickou metodou) problém rozdělit na disjunktní podmnožiny podle vybraných produktů.

	Počet produktů	Počet měření	Medián produktů na objednávku	Počet měření : produktů
Průměr	48.16	1195.40	2.32	40.99
Std	182.58	4926.42	3.15	102.67
Min	1	1	1	0.095
25%	2.24	11	1	2
50%	7	73	2	7
75%	25	488.75	2	31.77
Max	3027	59646	43	1039.59

Tabulka 2.2: Statistika dle eshopů

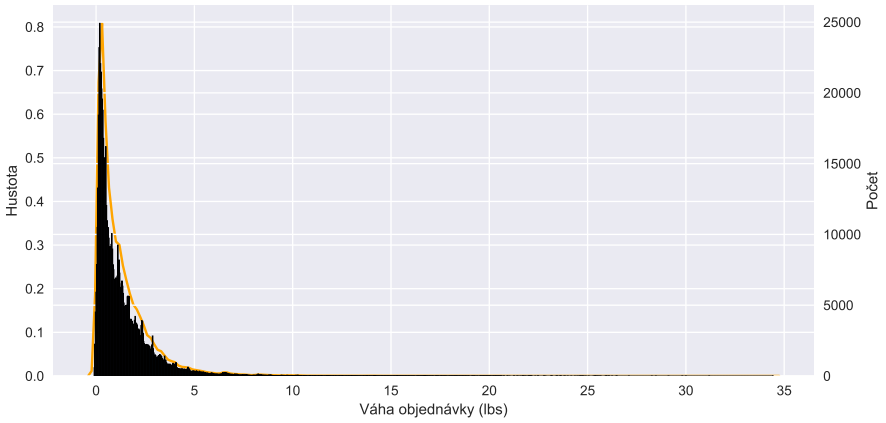


Obrázek 2.3: Charakteristika e-shopů

Na závěr analýzy je třeba se seznámit s distribucí vážení objednávek, jelikož u objemnějšího zboží můžeme očekávávat výrazně vyšší rozptyl, což by znehodnocovalo odhady vah menších produktů.

Na grafu 2.4 a tabulce 2.3 je vidět, že ve většině případů jde o objednávky menšího rozměru od 0.1 lb do 3 lb a téměř zanedbatelné procento tvoří objednávky nad 5 lb.[5]

[5] 1 kg ≐ 2,205 lb. Všechna měření byla provedena v jednotkách lb.



Obrázek 2.4: Distribuce vážení objednávek

	Počet produktů
Průměr	1.285
Std	1.515
Minimum	0.010
25%	0.320
50%	0.805
75%	1.720
Maximum	34.440

Tabulka 2.3: Statistika vážení objednávek

Kapitola 3

Metody řešení

V této kapitole je představeno několik modelů řešících úlohu popsanou v kapitole 2.1.5 s přihlédnutím na strukturu dostupných dat analyzovanou v kapitole 2.2. Relevantní studie zabývajících se konkrétně touto problematikou nebyly nalezeny, nicméně z povahy úlohy se nabízí tři základní přístupy, na které se dá problém převést.

Prvním přístupem je hledání **řešení soustavy rovnic** ve stylu Gaussovy eliminační metody (dále „GEM“), přičemž je třeba předpokládat, že systém rovnic je přeurčený a do výsledků se nutně bude propagovat nenulová chyba v důsledku chyb měření. Tento přístup je reprezentován *Naivní metodou* (3.1).

Problém lze také chápat jako **optimalizační úlohu**, ve které minimalizujeme $|w - E[P]|$.[6] Do této skupny zahrnuji *regresní modely* (3.2) a *lineární programování* (3.3).

Alternativou je také pravděpodobnostní pohled na problém, při kterém se neprovádí odhad střední hodnoty $E[P]$, ale rovnou distribuce rozdělení všech vah produktů P pomocí metody MLE (*Maximum Likelihood Estimation*). Tuto metodu popsal Toshiro Tango [2] a následně ji využil pro řešení podobného problému. Ve svém článku však uvádí jako nutný předpoklad konstantní rozptyl odhadovaných vah napříč pozorovanými skupinami a zároveň pracuje s řádově nižším počtem proměnných na výrazně větším objemu dat, což neodpovídá naší úloze.

[6] Kde $||$ reprezentuje libovolnou metriku.

3.1 Naivní metoda

Naivní metoda je založena na rekurentním dosazování odhadů vah jednotlivých produktů „zdola nahoru“.

Na vstupu algoritmus přijímá matici objednávek $\boldsymbol{Q}$ a vektor vážení objednávek Y (viz. kapitola 2.1.5). V každé iteraci pak hledá takové řádky, ve kterých se vyskytuje právě 1 produkt. Řádky se stejným produktem seskupí a následně z nich provede odhad váhy produktu. Pro zajištění reprezentativního vzorku je požadováno minimálně k měření daného produktu. Zároveň je před provedením odhadu nutné odstranit odlehlá měření (např. pomocí mezikvartilového rozpětí či na základě z-skóre). Celý pseudokód algoritmu je popsán níže.

Algoritmus 1: Naivní metoda

```
Data: Q - matice objednávek, y = vektor vážení objednávek
Result: Result - Váhy produktů
end ⟵ 0
Result ⟵ ∅
while True do
    updated ⟵ 0
    /* R = množina dvojic (objednávka = řádek Q, zvážená
       hodnota)                                          */
    R ⟵ FindSingleProductMeasurements(Q, y)
    /* P = množina dvojic (index sloupce v Q, pole z R
       seskupené podle produktu)                         */
    P ⟵ GroupByProduct(R)
    for p ∈ P do
        w ⟵ EstimateWeightFromSample(p)
        /* Může nastat v případě nedostatečně velkého vzorku
           pro provedení odhadu                          */
        if w is NULL then
            continue
        end
        y ⟵ SubstractEstimatedWeight(y, p, w)
        Q ⟵ ResetQuantityForProduct(p)
        Result ⟵ Result ∪ (getColumnId(p), w)
        updated ⟵ 1
    end
    /* Nebyl provede žádný odhad v této iteraci          */
    if updated = 0 then
        return
    end
end
```

V každé iteraci se tedy sníží počet produktů v objednávkách, případně algoritmus skončí, pokud není z čeho dále provádět odhady.

Funkce *EstimateWeightFromSample* může být implementována libovolně, v této práci byl využit medián/průměr s odstraněním odlehlých dat pomocí filtrace na základě z-skóre. Funkce *SubstractEstimatedWeight* projde všechny řádky vektoru y a z každého odečte $q_{i;p} * \hat{w}_p$, tedy odhadnovanou váhu vynásobenou počtem kusů produktu v objednávce. Funkce *ResetQuantityForProduct* pak vynuluje počet kusů již „využitého“ produktu, tak aby bylo možné provést odhady v další iteraci.

3.1.1 Nedostatky naivní metody

Metoda je označena jako „naivní“ z toho důvodu, že při provádění odhadu nebere v potaz všechna měření, ve kterých se daný prvek vyskytuje, ale pracuje pouze s takovou podmnožinou objednávek, které v dané iteraci zbývá právě 1 produkt. Pokud algoritmus odhadne váhu konkrétního produktu z takové podmnožiny, je tento odhad zpropagován do všech dalších měření, ve kterých se produkt vyskytuje, a měření s více produkty nemají šanci tento odhad v případě, že je chybný, napravit.

Naivní metoda také neumí využít informace o váhách objednávek, které nelze odvodit prostým dosazením váhy jednoho produktu po druhém, ale jsou lineární kombinací jiných objednávek. Příklad takové situace je uveden níže (3.1).

$$O_1 = (1, 1, 1, 1, 1) \qquad O_2 = (1, 0, 0, 1, 0) \qquad O_3 = (0, 1, 1, 0, 0) \tag{3.1}$$

Za předpokladu, že máme k dispozici dostatečný počet měření objednávek O_1, O_2, O_3, můžeme odvodit váhu 5. produktu, jelikož

$$O_1 = O_2 + O_3 + (0, 0, 0, 0, 1)$$

Naivní metoda ale takové dosazení neprovede.

Problém jistě lze řešit, ale hledání všech takových lineárních kombinací v matici $\boldsymbol{Q}$ by bylo výpočetně velmi náročné. Navíc, jak vyplývá z analýzy dat v kapitole 2.2, jedno či dvou produktových objednávek je významně nejvíce a s vyšším počtem produktů je počet objednávek klesající. Ačkoliv se tak využije jen zlomek informace z dostupných dat, výsledky by měly být postačující. Nicméně, všechny dále navrhované robustnější metody oba zmíněné problémy řeší.

3.1.2 Časová složitost

Při odhadu časové složitosti algoritmu vycházíme z následující analýzy 2.[7]

Algoritmus 2: Naivní metoda

```
⋮
/* Za předpokladu, že každou iteraci je možné odhadnout
   právě p produktů, se provede ⌈n/p⌉ iterací                */
while True do
    ⋮
    /* Projde všechny prvky, tedy m * n. Alternativou by
       bylo předpočítat počet produktů na objednávku před
       začátkem algoritmu (mn) a následně vyhledávat se
       složitostí O(m)                                       */
    R ⟵ FindSingleProductMeasurements(Q, y, i)
    /* Záleží na počtu řádků, ve kterých se vyskytuje p,
       maximálně však m                                      */
    P ⟵ GroupByProduct(R)
    /* P-krát                                                */
    for p ∈ P do
        /* Záleží na počtu řádků, ve kterých se vyskytuje p,
           maximálně však m                                  */
        w ⟵ EstimateWeightFromSample(p)
        ⋮
        /* Záleží na počtu řádků, ve kterých se vyskytuje p,
           maximálně však m                                  */
        SubstractEstimatedWeight(y, p, w)
        /* Záleží na počtu řádků, ve kterých se vyskytuje p,
           maximálně však m                                  */
        ResetQuantityForProduct(p)
        ⋮
    end
end
```

Horní hranice složitosti je tedy maximálně

$$\mathcal{O}(\frac{m}{p}(mn + m + 3mp)) = \mathcal{O}(\frac{m^2}{p}(n + p)) \tag{3.2}$$

což (vzhledem k $p \in [1; n]$), odpovídá $\mathcal{O}(m^2 n)$.[8]

[7] V analýze se pro ulehčení předpokládá, že každou iteraci nalezneme právě p produktů, což je není nutně pravdivý předpoklad, ale pro nalezení horní asymptotické složitosti postačuje.

[8] Pro případ $p = 0$ je časová složitost $\mathcal{O}(m * n)$.

Pokud by se algoritmus upravil tak, aby před začátkem běhu předpočítal počet produktů na objednávku ($\mathcal{O}(mn)$), metoda *FindSingleProductMeasurements* by měla složitost $\mathcal{O}(m)$ a celková horní hranice složitosti by se snížila na $\mathcal{O}(\max(m^2, mn))$.

3.2 Lineární regresní modely

Ačkoliv tuto kapitolu nazývám „regresními modely“, fakticky řeším pouze podproblém regresní analýzy – nesnažím se predikovat hodnotu závislé veličiny na základě nezávislých proměnných (příznaků), ale jen o odhad parametrů lineárního modelu, což je problém ekvivalentní s hledáním $E[P]$.

Regresní model tedy stručně představím, jelikož v praktické části využívám implementace celého modelu, primárně se ale budu zabývat použitými metodami pro odhad parametrů modelu.

3.2.1 Lineární regrese

Model lineární regrese je statistická diskriminativní metoda predikují hodnotu nějaké spojité náhodné veličiny Y ($E(Y|\boldsymbol{X} = x)$), přičemž předpokládáme, že tato náhodná veličina je lineárně závislá na příznacích $x_1, x_2, \cdots, x_p$, u kterých jsme při měření schopni zajistit zanedbatelnou chybu.

$$
\begin{aligned}
Y &= w_0 + \sum_{1}^{p} w_i x_i + \varepsilon = w^T x + \varepsilon \qquad (3.3)\\
E\varepsilon &= 0\\
w &= (w_0, w_1, \cdots, w_p)^T \in \mathbb{R}^{p+1}\\
x &= (1, x_1, x_2, \cdots, x_p)^T \in \mathbb{R}^{p+1}
\end{aligned}
$$

kde x je vektor příznaků, a w parametry modelu (tzv. *váhy*). Náhodná veličina ε v sobě skrývá náhodnost veličiny Y.

Predikcí modelu pak rozumíme:

$$\hat{Y} = \hat{w}^T x \qquad (3.4)$$

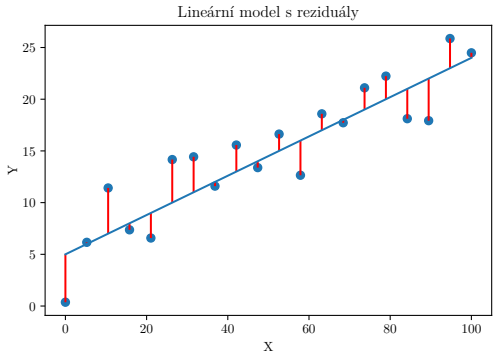
kde $\hat{w}$ je konkrétní odhad parametrů modelu. Z předpokladu $E\varepsilon = 0$ vyplývá, že predikce $\hat{Y}$ je bodovým odhadem EY v bodě x.

Pro rozumný odhad parametrů modelu je třeba specifikovat, jak změřit chybovost modelu. Chybu konkrétního odhadu měříme pomocí tvz. *ztrátové funkce* L, typicky $L : \mathbb{R}^2 \to [0; \infty)$. Celkovou chybou modelu pak rozumíme:

$$Err(w) = \sum_{i=1}^{N} L(Y_i, \hat{w}^T x_i) \qquad (3.5)$$

kde Y_i je realizace náhodné veličiny Y (bod trénovací množiny). Nejlepším odhadem $\hat{w}$ je pak ten minimalizující celkovou chybu modelu. V závislosti na volbě *ztrátové funkce* rozlišujeme různé metody pro odhad parametrů.

Jelikož na základě modelu neprovádíme predikce, není třeba řešit problém přeučení modelu (např. regularizací) a na odhadování parametrů je možné použít celý dataset.



Obrázek 3.1: Lineární model $Y = ax + b$

3.2.2 Metoda nejmenších čtverců (Ordinary Least Squares)

Metoda nejmenších čtverců (dále *OLS*) využívá *kvadratickou ztrátovou funkci.*

$$L(Y_i, \hat{Y}_i) = (Y_i - \hat{Y}_i)^2 \tag{3.6}$$

Jelikož je *kvadratická ztrátová funkce* konvexní a diferencovatelná na celém oboru hodnot, je i funkce $Err(w)$ konvexní a diferencovatelná [3]. Globální minimum $Err(w)$ tedy nalezneme vždy a to jak gradientním sestupem, tak analyticky (OLS).

Mějme N párů měření (Y_i, x_i), kde $Y_i = w^T x_i + \varepsilon$.

$$\boldsymbol{Y} = (Y_1, Y_2, \cdots, Y_N)^T \tag{3.7}$$

$$\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_N)^T \tag{3.8}$$

$$\boldsymbol{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} 1 & x_{1;1} & x_{1;2} & \cdots & x_{1;p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N;1} & x_{N;2} & \cdots & x_{N;p} \end{pmatrix} \tag{3.9}$$

$$\boldsymbol{Y} = \boldsymbol{X}w + \boldsymbol{\varepsilon} \tag{3.10}$$

V bodech, kde funkce Err nabývá minima musí mít nutně i nulovou první derivaci. Body *podezřelé z extrému* jsou tedy body splňující rovnici 3.11.

$$\nabla Err = -\sum_{i=1}^{N} 2(Y_i - w^T x_i)x_i = -2\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}w) = \mathbf{0} \tag{3.11}$$

kde $\nabla Err(w)$ značí gradient funkce $Err(w)$. Ověřit, zda nalezené body jsou opravdu extrémy funkce lze z Hessovy matice (matice druhých parciálních derivací $Err(w)$). Při použití *kvadratické ztrátové funkce* je tomu tak vždy [4].

Předchozí rovnici $\nabla Err = \mathbf{0}$ lze dále přeskládat na tzv. *normální rovnici*

$$\boldsymbol{X}^T\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{X}w = \mathbf{0} \tag{3.12}$$

Za předpokladu, že matice $\boldsymbol{X}$ je regulární, je nejlepším odhadem $\hat{w}$:

$$\hat{w}_{OLS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \tag{3.13}$$

Matice $\boldsymbol{X}^T\boldsymbol{X}$ je vždy čtvercová, ale nemusí být regulární v případě, že sloupce matice $\boldsymbol{X}$ nejsou lineárně nezávislé.

Pro výpočet $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ se proto používá *Moore-Penrose inverze* (neboli *pseudoinverze*), která vrátí nejlepší aproximaci řešení. Typická implementace využívá pro nalezení *pseudoinverze* singulární rozklad matice [5]. Knihovna LAPACK nespecifikuje časovou složitost této operace, nicméně z rozkladu SVD matice lze očekávat $\mathcal{O}(m * n^2 + n^3)$ pro matici o rozměrech $m \times n$ [6].

3.2.3 Metoda nejmenších čtverců omezená na pozitivní řešení (Non-negative Least Squares)

Metoda OLS vrací optimální řešení nehledě na další omezení, která jsou v reálných datech přítomna. Reálná váha produktů nikdy nemůže být záporná. Pokud tedy OLS vrací záporný koeficient, nejen, že odhad váhy daného produktu nelze použít, ale pravděpodobně tento parametr negativně ovlivnil i ostatní odhady.

Nabízí se tedy řešit problém:

$$\arg\min_{w} (\boldsymbol{Y} - \boldsymbol{X}w)^2 \text{ za podmínky } w \geq 0 \tag{3.14}$$

Jelikož je ztrátová funkce konvexní, globální minimum existuje. Úlohy kvadratického programování lze řešit mnoha metodami, já využívám metodu active-set [7]. Matematika použitá na řešení kvadratické optimalizace je nad rámec této bakalářské práce.[9]

Od této metody lze očekávat lepší výsledky za cenu delšího běhu výpočtu – obecná kvadratická optimalizace je NP-těžký problém [8] a ačkoliv pro některé vstupy je známý polynomiální algoritmus, je téměř jisté, že silně optimalizovaná implementace hledání pseudoinverze při řešení OLS bude několikanásobně rychlejší.

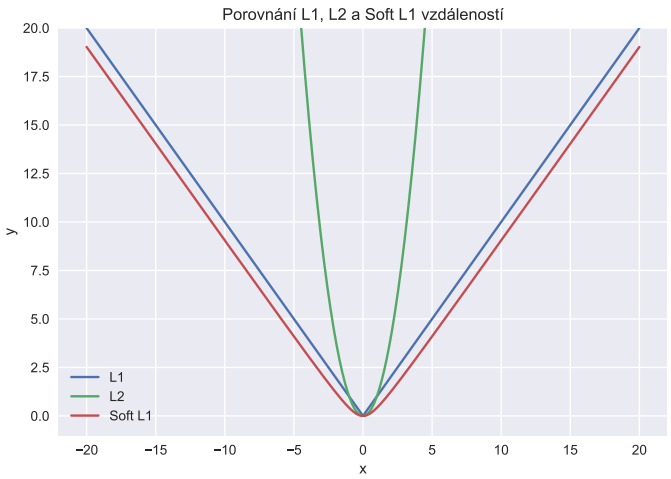
[9] Metody používané pro konvexní optimalizaci popsal například Philip E. Gill a Elizabeth Wong ve článku „Methods for Convex and General Quadratic Programming“, 2014 [7].

3.2.4 Soft L1 NNLS

Principiálně jde znovu o metodu nejmenších čtverců omezenou na pozitivní řešení, avšak kvadratickou ztrátovou funkci nahrazuje funkce Soft L1 3.15.

$$L(Y_i, \hat{Y}_i) = \sqrt{1 + (Y_i - \hat{Y}_i)^2} - 1 \tag{3.15}$$

Jak už z názvu vyplývá, jde o hladkou aproximaci absolutní vzdálenosti, resp. MAE, při zachování diferencovatelnosti a konvexitě na celém definičním oboru. Chování této funkce reprezentuje graf 3.2.[10]



Obrázek 3.2: Porovnání L1, L2 a Soft L1 vzdáleností

Stejně jako u metody NNLS (viz. 3.2.3) je pro nalezení parametrů modelu minimalizující tuto chybu nutné použít techniky spojité konvexní optimalizace, které typicky umožňují pracovat i s lineárními omezeními jednotlivých proměnných [9].

Srovnáním MAE a MSE se zabývá kapitola 3.3.1. Předpokládáme, že tento model bude dosahovat robustnějších výsledků za cenu ještě delšího běhu výpočtu než tomu je u NNLS.

[10] Alternativou Soft L1 s prakticky stejným chováním je ztrátová funkce Huber Loss, resp. Huber Loss regression

3.3 Lineární programování

Podobně jako u regresních modelů jde o řešení optimalizační úlohy, ale namísto MSE se minimalizuje MAE (*mean absolute error*). Díky tomu by měl být model robustnější, tedy nemělo by být tak jednoduché rozhodit odhad malým počtem výrazně chybných měření.

Optimalizační úlohu lze formulovat následovně:

$$\text{minimize } |\sum_{i=1}^{m} e_i| \tag{3.16}$$

$$\begin{aligned} \text{subject to: } i \in \{1, 2, \ldots, n\} : x_i > 0 \\ \boldsymbol{Q} \cdot x + e = \boldsymbol{Y} \end{aligned} \tag{3.17}$$

kde $x \in \mathbb{R}^n$ je vektor hledaných vah, $e \in \mathbb{R}^m$ je vektor reprezentující chybovost (i-tý prvek je hodnotou chyby v i-té rovnici), $\boldsymbol{Q}$ je matice měření objednávek (2.1.5).

Dále je třeba zbavit se absolutní hodnoty, což lze provést zavedením dalších proměnných a omezení [10]:

$$\text{minimize } \sum_{i=1}^{m}(e_{\alpha;i} + e_{\beta;i}) \tag{3.18}$$

$$\begin{aligned} \text{subject to: } i \in \{1, 2, \ldots, n\} : x_i > 0 \\ i \in \{1, 2, \ldots, m\} : e_{\alpha;i} \geq 0 \\ i \in \{1, 2, \ldots, m\} : e_{\beta;i} \geq 0 \\ \boldsymbol{Q} \cdot x + e_\alpha - e_\beta = \boldsymbol{Y} \\ e_\alpha, e_\beta \in \mathbb{R}^m \end{aligned} \tag{3.19}$$

Standardní úloha LP pak navíc povoluje omezení pouze ve formě nerovností $x \leq y$, nicméně prostými algebraickými operacemi lze každou lineární (ne)rovnost do kýženého tvaru převést [10]. Většina LP řešičů navíc tento přepis zvládá automaticky.

Rozdělení proměnné reprezentující chybovost každé rovnice na dvě (kladnou a zápornou část) za účelem odstranění absolutní hodnoty v užitkové funkci nicméně není bez následků. Simplexový algoritmus, který implementuje většina LP řešičů, má pro některé vstupy až exponenciální časovou náročnost, i když se průměrná složitost ukazuje být polynomiální [11].[11] V každém případě lze předpokládat, že s každým dalším omezením či proměnnou složitost stoupá polynomiálně.

Od této metody očekávám nejpřesnější výsledky (na základě robustnějšího chování vůči odlehlým datům) za cenu delšího běhu výpočtu způsobeného velkým počtem proměnných.

[11] Je dokázáno, že LP problémy lze řešit v polynomiálním čase pomocí elipsoidní metody. V praxi je však tato metoda pro většinu vstupů pomalejší než Simplexový algoritmus. [12]

3.3.1 Porovnání MAE a MSE

Výše navržené modely řešící *optimalizační úlohu* se (při opomenutí vnitřní implementace) primárně liší v použité užitkové funkci – zatímco regresní modely minimalizují *MSE*, lineární programování minimalizuje *MAE*.

Ilustrovat rozdílné chování při volbě parametrů lze například na jednoduchém lineárním modelu $Y = c$.

Za pomocí matematické analýzy lze ukázat, že nejlepším odhadem parametru c při použití metriky *MSE* je výběrový **průměr** (3.20).

$$\forall i \in \mathbb{N} : \hat{Y}_i = c \tag{3.20}$$

$$\mathrm{MSE}(Y, \hat{Y}) = (Y - \hat{Y}) = \sum_{i=1}^{N} (X_i - c)^2 = \mathrm{MSE}(Y, c) \tag{3.21}$$

$$\frac{\partial \mathrm{MSE}}{\partial c} = \sum_{i=1}^{N} 2(X_i - c)$$

Funkce *MSE* je pro parametr c diferencovatelná a konvexní na celém oboru hodnot [3], minimum tedy nabývá v bodě, ve kterém je první derivace nulová.

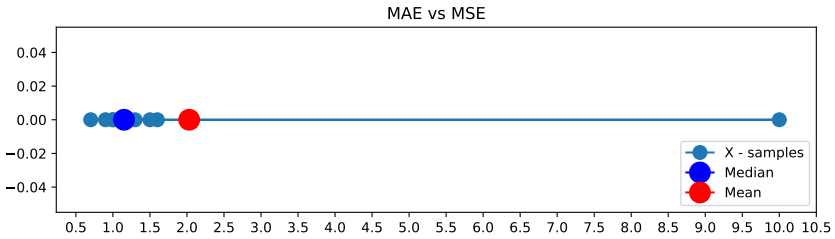
$$\frac{\partial \mathrm{MSE}}{\partial c} = 0 \tag{3.22}$$

$$\sum_{i=1}^{N} 2(X_i - c) = 0 \tag{3.23}$$

$$\sum_{i=1}^{N} 2X_i - 2cN = 0 \tag{3.24}$$

$$c = \frac{\sum_{i=1}^{N} X_i}{N} = \mathrm{mean}(X) \tag{3.25}$$

Pro odvození nejlepšího odhadu při použití metriky MAE je třeba být trochu kreativnější, jelikož absolutní funkce není v bodě, kde nabývá minima diferencovatelná, každopádně nejlepším odhadem parametru c je **medián**. [13].



Obrázek 3.3: Ukázka rozdílu mezi *MAE* a *MSE* na 1D datech

Jak je vidět na grafu 3.3, odhad parametru reprezentovaný mediánem je výrazně lepším odhadem střední hodnoty náhodné veličiny X, jelikož metrika MSE dává větší důraz odlehlým bodům – zatímco u predikčního modelu je toto žádané chování, pro náš problém je to přesně naopak. Lze tedy říct, že modely minimalizující *MAE* pro odhad parametrů jsou robustnější.

Na základě tohoto zjištění předpokládáme, že *LP model* (3.3) bude vracet nejlepší výsledky a to zejména proto, že v reálných datech se budou vyskytovat i velké chyby měření způsobené např. neočekávaným obsahem objednávky, kdy je jeden produkt zaměněn za jiný.[12]

3.4 Nedostatky optimalizačních modelů oproti naivní metodě

Jak bylo již zmíněno v kapitole 3.1, optimalizační metody oproti naivní metodě využívají veškerá dostupná data pro určení výsledného odhadu. Naivní metoda má ovšem oproti ostatním modelům jednu velmi důležitou vlastnost – díky principu postupného dosazování zdola nahoru a požadavku na minimální počet měření dané varianty objednávky pro provedení odhadu lze jednoduše kontrolovat, jak je odhad důvěryhodný. Situaci lze pěkně reprezentovat na příkladě 3.26.

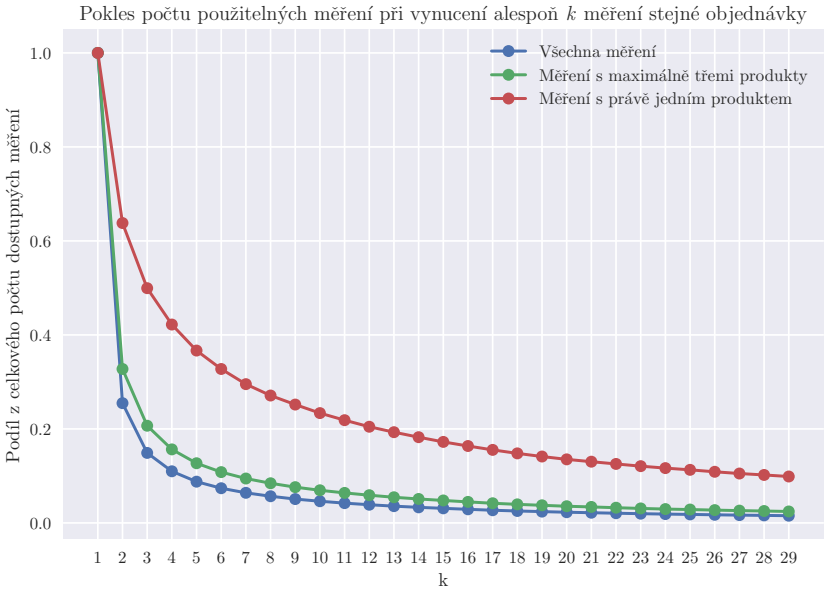
$$O_1 = (1, 0, 0, 0) \qquad O_2 = (0, 1, 0, 0) \qquad O_3 = (0, 1, 1, 0) \tag{3.26}$$

Mějme k dispozici právě 10 měření objednávek O_1, O_3 a 9 měření O_2. Pokud je minimální počet měření k jedné objednávky pro provedení odhadu 10, naivní model provede odhad pouze pro produkt 1, jelikož vychází pouze z objednávky O_1.

Optimalizační metody oproti tomu odhadnou váhu každého produktu, nehledě na počet dostupných měření či fakt, že na provedení odhadu 3. a 4. produktu nemáme dostatek informací (součet vah těchto dvou produktů se může libovolně rozdělit mezi tyto dva produkty s nulovou výslednou chybou v užitkové funkci).

Pokud bychom chtěli podobného výsledku dosáhnout už při předzpracování dat – tedy vyfiltrovat veškerá měření takových variant objednávek, které nebyla provedeny alespoň k-krát, přišli bychom o majoritu měření jak ukazuje graf 3.4.

[12] Do jisté míry se tomuto problému vyhýbám odstraněním odlehlých dat již při předzpracování, nicméně tam pracuji pouze na úrovni jednotlivých variant objednávek, nikoliv všech měření, ve kterých se produkt vyskytuje.



Obrázek 3.4: Podíl dat z celkového počtu měření při vynucení alespoň k měření dané varianty objednávky

Částečnou odpovědí by mohla být regresní analýza koeficientů u regresních modelů, avšak při experimentálním testování této metody nebyly nijak vypovídající.

3.5 Metodika hodnocení kvality metod

Klasický způsob hodnocení kvality regresních modelů je porovnání predikcí oproti naměřeným hodnotám na nepoužité části datasetu (testovací množina). Nejčastěji používanými metrikami jsou pak (R)MSE, MAE, R2 skóre či RMSLE [14], přičemž každá z metrik na data pohlíží jinak a výsledky je třeba specificky interpretovat.

Jak už ale bylo zmíněno v kapitole 3.2, cílem této práce není provádět odhad váhy objednávky, ale získat co nejpřesnější odhady $E[P]$. Kvalitu našeho modelu tedy nebude udávat rozdíl predikce od naměřených hodnot, ale rozdíl odhadu vah jednotlivých produktů od reálných vah produktů.

$$S(E[P], \hat{E}[P]) = M(E[P] - \hat{E}[P]) \tag{3.27}$$

kde S je výsledné skóre modelu, $E[P]$ jsou reálné střední hodnoty vah produktů a $\hat{E}[P]$ jsou odhady vah produktů.[13] Funkce M reprezentuje libovolnou metriku. V této práci byla zvolena metrika RMSE (*Root Mean Squared Error*, def. 3.28).

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum(E[P] - \hat{E}[P])^2} \tag{3.28}$$

Zatímco při výpočtu odhadovaných vah bylo vhodné volit robustnější model kvůli chybám v datech (s čímž se váže metrika MAE), při porovnání jednotlivých metod už chceme více penalizovat velké chyby. Hlavní rozdíl oproti MSE je, že výsledná hodnota je reprezentována v původních jednotkách, díky čemuž je výsledek o něco snazší interpretovat.

[13] Získáním dat vhodných k hodnocení kvality metod se zabývá kapitola 4.2

Kapitola 4

Realizace

4.1 Výběr nástrojů pro implementaci

Zdrojový kód práce je psán v programovacím jazyce *Python*. Jazyk byl zvolen kvůli existenci kvalitních knihoven pro konkrétní doménu (operace s maticemi, optimalizace), ale i vzhledem k popularitě jazyka *Python* v odvětví Data Science [15].

4.1.1 Použité knihovny

Pro zpracování naměřených dat ve formátu CSV byla využita knihovna *Pandas*, která vytváří vrstvu abstrakce nad tabulárními daty a zjednodušuje práci s nimi. V našem případě je použita pro načtení a následnou transformaci vstupních dat do maticové podoby potřebné pro běh výpočetních modelů. Knihovna interně reprezentuje data za pomocí níže zmíněné knihovny *NumPy*.

Matice a operace s nimi v celém projektu zajišťuje knihovna *NumPy*. Knihovna je napsaná v jazyce *C*, díky čemuž obchází výkonnostní limitace interpretovaného jazyka *Python*, ale zároveň vystavuje uživatelsky pohodlné API pro jazyk *Python*.

Knihovna *SciPy* pak poskytuje složitější matematické operace jako SVD rozklad matice, QR dekompozici, metodu OLS a další. Knihovna sama předpokládá použití *NumPy* pro reprezentaci matic a ve většině případů pouze převolává silně optimalizované algoritmy z knihoven *LAPACK* a *BLAS*.

K vizualizaci dat byla použita populární knihovna *Matplotlib*. Práci s touto knihovnou v mnoha případech ulehčila knihovna *Seaborn* jakožto nadstavba knihovny *Matplotlib*, nabízející předpřipravené vizualizace pro „standardně“ formátovaná data.

4.2 Získání dat pro měření kvality modelů

Vzhledem k neexistenci referenčních vah produktů z produkčních dat bylo třeba vygenerovat nové datové sady, které budou svou charakteristikou co nejvíce odpovídat reálným datům. Generování zároveň umožňuje kontrolu nad jednotlivými parametry, jako míra chybovosti či průměrná odchylka vah produktů a sledovat vliv těchto parametrů na jednotlivé modely.

Generátor dat je reprezentován modulem *app.sample_weight_generator*. Na vstupu předpokládá následující parametry:

- PRODUCT_CONFIGURATION: ProductConfiguration - definuje typ a parametry rozdělení, ze kterého se budou generovat váhy jednotlivých produktů (tedy $E[P_i]$)
- ORDER_CONFIGURATION: OrderConfiguration - definuje diskrétní rozdělení počtu různých produktů v objednávce a počtu kusů těchto produktů v objednávce.
- NUMBER_OF_PRODUCTS: integer - Počet produktů v datové sadě
- NUMBER_OF_ORDERS: integer - Celkový počet měření v datové sadě
- SCALE_STDEV: float - Standardní odchylka váhy (zařízení) v librách
- HARD_ERR_RATIO: float - Poměr hrubých chyb měření
- PRODUCT_WEIGHT_STDEV_RATION: float - Průměrná odchylka $\sigma_{\%}$ od této hodnoty pro jednotlivé instance produktu v procentech – $P_i = N(E[P_i], \sigma_{\%}E[P_i])$

Hodnoty těchto parametrů vychází z analýzy dat v kapitole 2.2 a dodatečné analýzy dostupné v jupyter notebooku *data_analysis/Production Data Analysis.ipynb*. Standardní odchylka váhy je fixně nastavena na 0.03 lbs, což odpovídá standardní odchylce uváděné výrobcem váhy použité k měření produkčních dat.

Rozdělení váhy produktů bylo odvozeno z podmnožiny produktů z produkčních dat, které byly zvážené manuálně. Tyto váhy byly následně napasovány na různá rozdělení, která byla porovnána na pravděpodobnostním grafu [16].

S cílem zachytit ostatní parametry co nejpřesněji byly e-shopy na základě grafu 2.3 rozděleny do 3 kategorií – do 500 měření, $[500; 10000)$ měření a $[10000; +\infty)$ měření za poslední tři měsíce.

Pro tyto tři skupiny byly následně spočítané statistiky poměru počtu produktů a objednávek, rozdělení počtu produktů a počtu kusů těchto produktů v objednávkách. Základní hodnoty ukazuje tabulka 4.1.

	Počet produktů	Počet měření	Počet produktů : počet měření
Malé eshopy	20	200	1:10
Střední eshopy	30	1500	1:50
Velké eshopy	300	45000	1:150

Tabulka 4.1: Základní konfigurace generátoru dle skupin

Parametry rozdělení pro jednotlivé skupiny jsou dále uloženy v souboru *src/app/sample_weight_generator/data_configuration.py*. Produkty byly do jednotlivých objednávek navíc vybírány s pravděpodobností vytvořenou z exponenciálního rozdělení ($\lambda = 1$), díky čemuž data lépe reprezentují reálnou situaci, kdy některé produkty jsou více oblíbené než jiné.

Od každé skupiny bylo náhodně vygenerováno 15 datových sad ve výstupním formátu csv definovaném produkčními daty v kapitole 2.2, přičemž ke každé datové sadě byly uloženy i příslušící referenční váhy produktů. To samé bylo provedeno pro všechny řádky z tabulky 4.2, které určují konfiguraci míry chybovosti ve generovaných datech. Vzniklo tedy 3x4 konfigurací, každá o 15 datových sadách.

	Průměrná odchylka váhy produktu v procentech	Poměr chybných měření
Ideální data	5%	0.1%
Očekávaná data	5%	1%
Očekávaná data - více chyb	10%	5%
Chybná data	15%	10%

Tabulka 4.2: Očekávaná chybovost v datech

4.3 Implementace metod

Každá z implementací splňuje velmi jednoduché rozhraní definované třídou *Model* (*app.weight_derivation.models.model*).

Běh algoritmu pro odhad vah produktů spouští metoda *model.estimate*, přijímající na vstupu datovou strukturu *WeightData*, která obsahuje dříve definovanou matici $\boldsymbol{Q}$ a vektor w popsané v kapitole 2.1.5. Výstupem této metody je vektor $E[P]$ korespondující s produkty ve sloupcích matice $\boldsymbol{Q}$.

Metody třídy *Model* se typicky neprovolávají napřímo, ale prostřednictvím třídy *ModelConfig*, která definuje konfiguraci modelu včetně názvu a metod předzpracování dat, a zajišťuje získání skóre modelu (třída *ModelAccuracy*) a běhové informace jako celková doba běhu výpočtu.

V následujících podkapitolách jsou uvedeny implementace představených metod v kapitole 3, porovnání implementací je pak následuje v kapitole 4.4.

Všechny implementace jsou součástí balíčku *app.weight_derivation.models*.

4.3.1 Naivní model

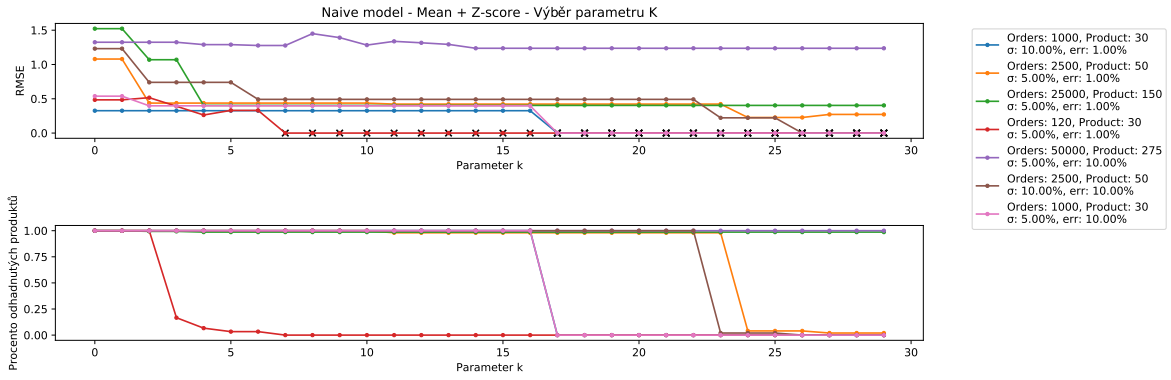
Zdrojový kód naivního modelu se nachází ve třídě *models.Naive* a jedná se o v podstatě doslovný přepis pseudokódu 1 s použitím knihovny *NumPy*. Korektní chování algoritmu je otestováno pomocí unit testů v souboru *tests/app/weight_derivation/models/test_naive.py*.

Konstruktor modelu přijímá následující parametry:

- get_expected_weight_from_samples: Callable[List[float], float] - Reprezentuje funkci *EstimateWeightFromSample* z pseudokódu (1), tedy určuje, jakým způsobem bude z nalezených měření daného produktu odvozena výsledná váha.

- min_number_of_measurements_required: int - Určuje minimální počet měření daného produktu pro provedení odhadu, jde tedy o parametr k popsaný v kapitole 3.1.

Graf 4.1 reprezentuje dopad volby parametru k na výslednou chybu (horní část grafu) a procento odhadnutých produktů (spodní část grafu) na striktnějším modelu s filtrací na základě z-skóre. Ideální hodnota parametru k by měla minimalizovala *RMSE* bez propadu procenta odhadnutých produktů. Optimální stav v tomto případě nelze objektivně definovat, jelikož metriky mezi sebou nelze porovnávat, nicméně rozumnou hodnotou parametru se zdá být $k = 5$.



Obrázek 4.1: Naivní metoda - průměr + z-skóre: Volba parametru k na různých variantách datových sad

Filtrací na základě z-skóre ($\alpha\sigma$) je myšleno odstranění takových měření předaných do metody *EstimateWeightFromSample*, pro která platí:

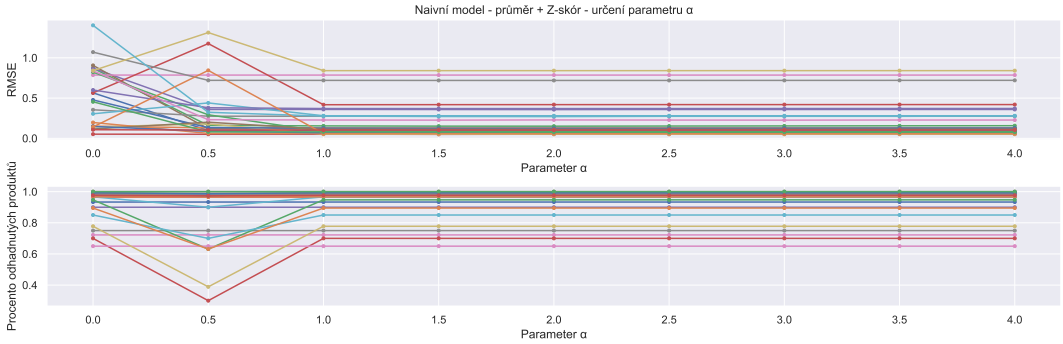
$$\left|\frac{X_i - \mu}{\sigma}\right| > \alpha \tag{4.1}$$

kde μ je očekávaná střední hodnota výběru a σ směrodatná odchylka výběru.[14] Pro menší výkyvy v případě extrémních hodnota na malém počtu měření je pro výpočet μ použit medián namísto výběrového průměru.

Na náhodném výběru 24 datových sad s rozdílnou konfigurací znázorňuje graf 4.2 vliv parametru α. Vrchní část grafu reprezentuje celkové RMSE skóre odhadu, spodní graf pak procento odhadnutých produktů.[15] Hodnota $\alpha = 0$ reprezentuje model bez filtrace.

Graf na testovacích datech ukazuje, že použití filtrace je zlepšuje odhady prakticky bez ohledu na zvolenou hodnotu parametru α až na hraniční hodnoty jako $\alpha = 0.5$ kdy je filtrováno až příliš mnoho dat a výrazně se snižuje celkový počet odhadnutých produktů.

Na základě tohoto porovnání byla zvolena hodnota parametry $\alpha = 2$, která by teoreticky měla pokrýt 95% interval vah produktů jakožto normálně rozdělených náhodných veličin.



Obrázek 4.2: Naivní metoda - průměr + z-skóre: Volba parametru α na různých variantách datových sad

Kvalita modelu byla měřena na následujících konfiguracích:

- Medián ($k = 5$)
- Medián + z-skóre (2σ) ($k = 5$)
- Průměr ($k = 5$)
- Průměr + z-skóre (2σ) ($k = 5$)

[14] Jde tedy o standardizaci na rozdělení $N(0, 1)$.

[15] Lze předpokládat, že procento odhadnutých produktů klesne, jelikož jde o metodu filtrace vstupních dat.

4.3.2 OLS (NNLS)

Implementaci OLS a NNLS metod zajišťuje třída *models.OLS*, která interně využívá *sklearn.linear_model.LinearRegression* knihovny *scikit-learn*.

Konstruktor modelu přijímá parametr *positive_only*, který určuje, zda se provede obvyklá metoda nejmenších čtverců (OLS) či varianta omezená na řešení z množiny $\mathbb{R}_0^+$ (NNLS). Lineární regresní model je nakonfigurován následujícím způsobem:

- normalize: bool = False - Normalizace dat nemá význam, jelikož všechny dimenze (resp. váhy produktů) jsou dodány ve stejných jednotkách.
- fit_intercept: bool = False - Nepřidáváme intercept, jelikož z definice problému v kapitole 2.1.5 vyplývá, že musí být nulový.
- n_jobs: int = 1 - Paralelizace na této úrovni nemá žádný význam, jelikož probíhá na úrovni sloupců parametru y reprezentující naměřené hodnoty (v našem případě jde vždy o **vektor** w)

4.3.3 Soft L1 - Least Squares

Metoda nejmenších čtverců se ztrátovou funkcí Soft L1 není dostupná v žádném předem připraveném regresním modelu a pro její implementaci bylo nutné využít obecnější metodu *optimize.least_squares* knihovny 'scipy', která umožňuje nalézt řešení nejmenších čtverců pro libovolnou nelineární funkci pro parametry s konstatními omezeními. Principiálně jde o hill-climbing algoritmus, ve kterém se posun rozhoduje na základě gradientu optimalizované funkce.

Implementace využívá metodu Trust Region Reflective, která je kombinací tabu prohledávání a gradientního sestupu. [17]

Funkce je volána s následujícími parametry:

- fun: Callable[[x: np.ndarray], y: np.ndarray] - Funkce, která vrací vektor reziduí, což je v našem případě $\mathbb{Q} \cdot x - w$.
- x_0: np.ndarray - Iniciální vektor vah w
- bounds: Tuple[np.ndarray, np.ndarray] - Spodní a horní omezení oboru hodnot x, resp. w
- method: str - *trf* (Trust Region Reflective algorithm)
- loss: str - Definuje zvolenou ztrátovou funkci, v našem případě tedy *soft_l1*

4.3.4 Lineární programování

Model řešící LP problém definovaný v kapitole 3.3 je implementován třídou *models.LinearProgramming*. Definici LP problému zajišťuje knihovna PuLP, která poskytuje vrstvu abstrakce nad nejpopulárnějšími open-source i komerčními řešiči LP a (M)ILP [18].

Na základě srovnání dostupných řešičů a experimentálních měření byl vybrán řešič COIN-OR CBC [19], resp. jeho část COIN-OR/Clp řešící LP problémy. Komerční řešiče jako Gurobi či CPLEX jsou dle srovnání rychlejší, ale licence jsou relativně drahé [20]. COIN-OR CBC je nejlepší open-source alternativou, a z experimentálního měření jsme ověřili, že dokáže v rozumném čase nalézt řešení i pro problémy velikosti 100 000 měření a 3 000 produktů.

Jak už bylo navíc řečeno, knihovna PuLP umožňuje výměnu CBC řešiče za např. Gurobi velmi jednoduše změnou konfigurace.

4.4 Porovnání implementací

V této kapitole jsou detailně porovnány všechny implementace na vygenerovaných datových sadách popsaných v kapitole 4.2.[16]

Změřena byla celková chyba RMSE, procento odhadnutých produktů, doba běhu a tzv. „relativní RMSE vůči naivnímu modelu“ (dále pouze „relativní RMSE“), která značí RMSE implementace pouze na podmnožině produktů odhadnutých naivním modelem s konfigurací [průměr/medián, $k = 5, \sigma = 2$].[17]

Před prezentací výsledků srovnání je třeba upozornit, že horizontální osa dále uváděných grafů reprezentuje míru „entropie“ (příp. konfiguraci náhodnosti) v generovaných datech a ačkoliv je seřazená, rozhodně nejde o spojitou lineární projekci. Typ grafu byl zvolen pouze za účelem vizualizace trendu.

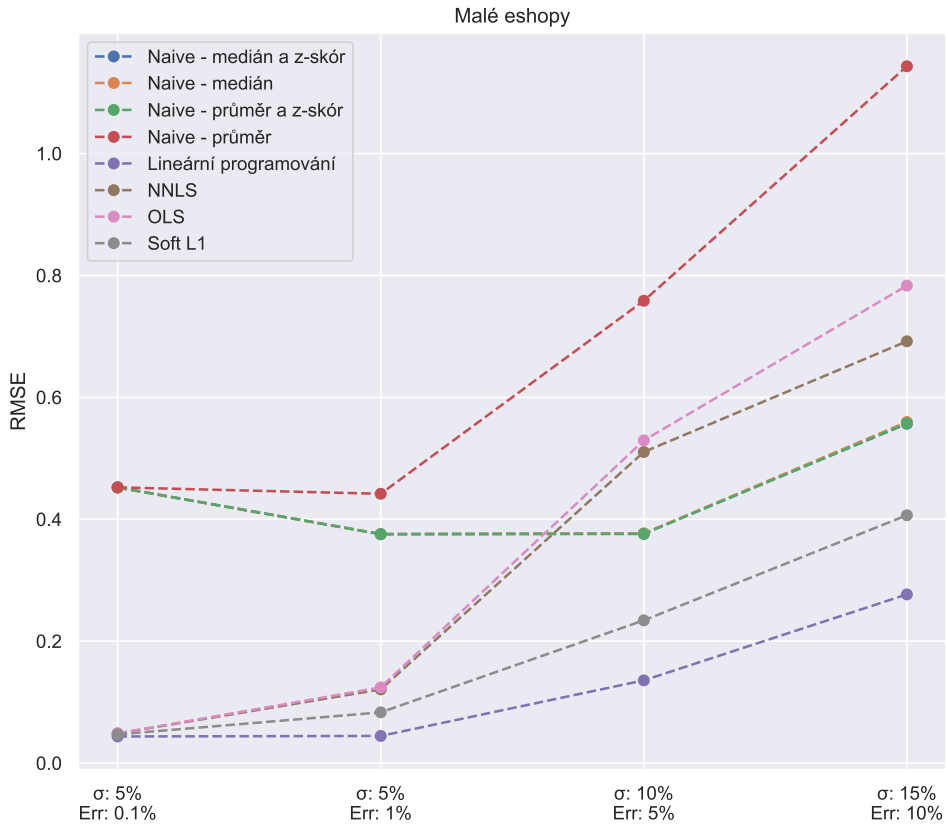
4.4.1 Malé e-shopy

Malé e-shopy jsou specifické menším počtem produktů a výrazně menším objemem objednávek. Poměr počtu objednávek : počet produktů se typicky pohybuje od 1:1 do 50:1.

Graf 4.3 porovnává průměrnou chybovost (RMSE) jednotlivých metod na vygenerovaných datech specifických pro menší e-shopy (20 produktů, 200 objednávek). Každý datový bod reprezentuje průměr z 15 datových sad pro danou konfiguraci (specifikovanou na horizontální ose).

[16] Celkem bylo připraveno 12 konfigurací, od každé 15 datových sad.

[17] Na volbě funkce *EstimateWeightFromSample* nezáleží, jelikož nijak nefiltruje data



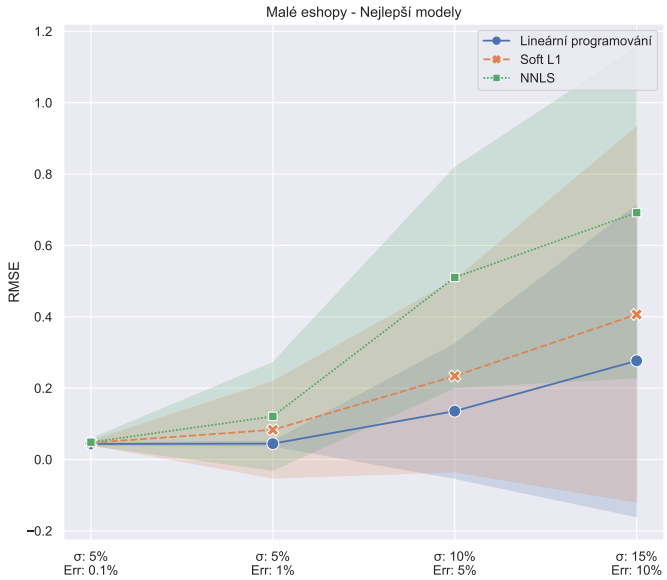
Obrázek 4.3: Porovnání implementací - Malé e-shopy

Ze srovnání vychází jednoznačně nejlépe *Lineární programování* (3.3), následované *Soft L1* metodou (4.3.3), tedy metody optimalizující ztrátovou funkci MAE.[18] Spolu s narůstající entropií datasetů se navíc rozdíl zvětšuje, což odpovídá hypotéze 3.3.1. Nejhůře se pak naopak umístil Naivní model [průměr, $k = 5$] bez z-skóre filtrace, který se nezvládl vypořádat s chybnými měřeními.

Implementace naivního modelu se v tomto srovnání umístily na posledním místě, jelikož neměly k dispozici dostatek dat k odhadnutí vah všech produktů. **Naivní model** s filtrací 2σ zvládl v průměru odhadnout váhy **80% produktů**.

Na grafu 4.4 je pak detailnější srovnání tří nejlepších modelů pro tuto kategorii dat. Poloprůhledné pásy vyznačují standardní odchylku jednotlivých implementací. Jak je vidět, metoda LP dává nejkonzistentnější výsledky.

[18] Resp. její aproximaci v případě Soft L1

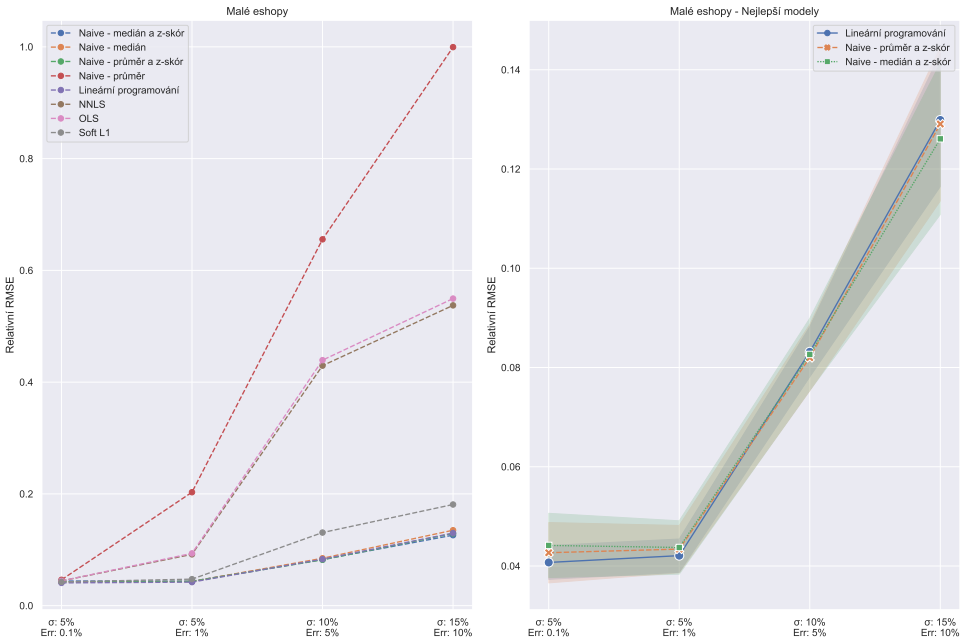


Obrázek 4.4: Porovnání nejlepších implementací - Malé e-shopy

Pokud bychom ale z nějakých důvodů (například kvůli vyžádání reprezentativního počtu měření každé varianty objednávky popsaném v kapitole 3.4) požadovali pouze odhady produktů, které jsou vypočítatelné naivním modelem, budou výsledky srovnání velmi odlišné.

Graf 4.5 porovnává relativní RMSE vůči naivnímu modelu, tedy počítá RMSE pouze z produktů, u kterých naivní model zvládl provést odhad (v tomto případě se podařilo odhadnout v průměru 80% produktů). Nejlepší výsledky vrací hned tři metody současně – Lineární programování, Naivní model [průměr, $k = 5, \sigma = 2$] a Naivní model [medián, $k = 5, \sigma = 2$]. Rozdíly ve střední hodnotě nejsou statisticky významné.[19]

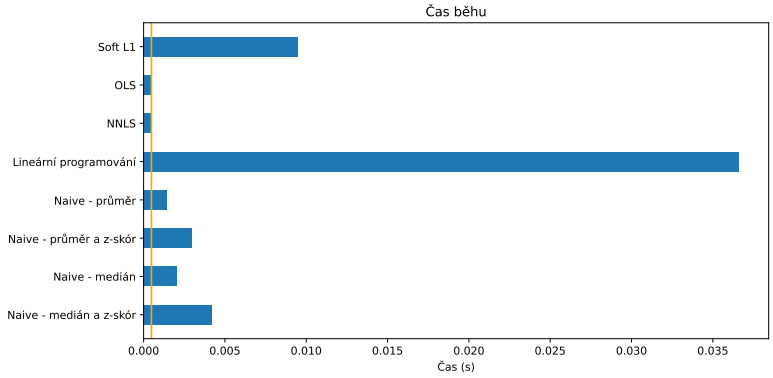
[19]Testováno dvouvýběrovým nepárovým t-testem na hladině významnosti 5% pro každou konfiguraci a kombinaci těchto tří metod.



Obrázek 4.5: Porovnání implementací - Malé e-shopy (Relativní RMSE)

Detailnější srovnání nejlepších metod spolu se standardní odchylkou jednotlivých měření reprezentuje graf 4.5 vpravo. Metoda LP je opět nejkonzistentější.

Průměrnou dobu běhu jednotlivých implementací znázorňuje graf 4.6.[20] Vzhledem k předpokladu, že výpočty poběží offline na pozadí, jsou rozdíly zanedbatelné.



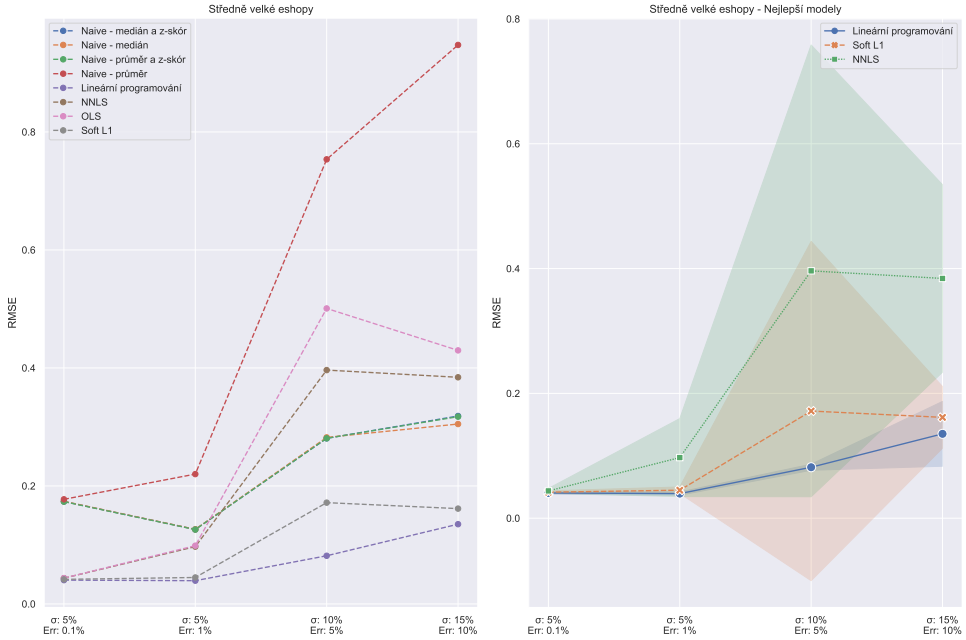
Obrázek 4.6: Doba běhu - Malé e-shopy

[20] Všechna měření byla provedena na sestavě Intel Core i9-9980HK 2.4GHz, 32GB DDR4 RAM. Testy byly spouštěny sekvenčně.

4.4.2 Středně velké e-shopy

Středně velké e-shopy mají typicky stále menší počet produktů, ale výrazně větší objem objednávek. V testovaných datech se předpokládá pro výpočet pesimističtější varianta, kdy je poměr počtu objednávek : počet produktů 50:1.

Graf 4.7 (vlevo) ukazuje průměrnou chybovost (RMSE) jednotlivých metod na vygenerovaných datech specifických pro e-shopy střední velikosti (30 produktů, 1500 objednávek).

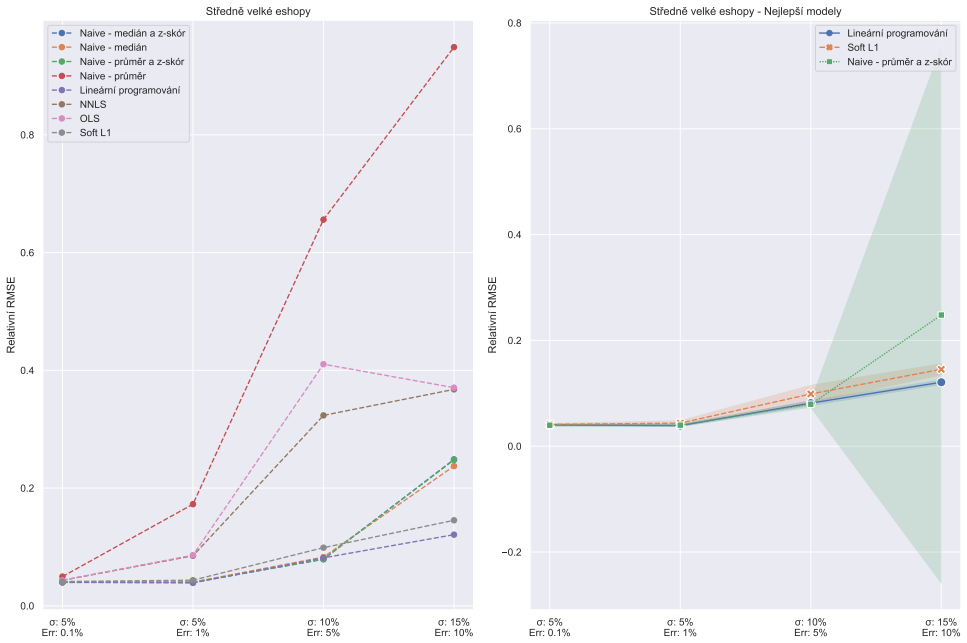


Obrázek 4.7: Porovnání implementací - Středně velké e-shopy

Oproti výsledkům na datech malých e-shopů podávají implementace naivních modelů výrazně lepší výkon, jelikož se jim díky většímu množství změřených objednávek podařilo odhadnout v průměru přes 96% produktů. Ve srovnání stále vede metoda *Lineárního programování* (3.3).

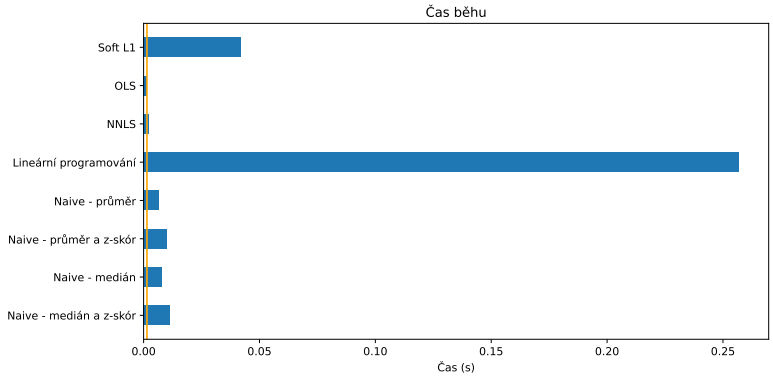
Vpravo na grafu 4.7 je opět detailnější srovnání tří nejlepších modelů pro tuto kategorii. Poloprůhledné pásy vyznačují standardní odchylku jednotlivých implementací. LP dává nejkonzistentnější výsledky.

Při porovnání „relativního RMSE“ se implementace naivního modelu opět vyrovnávají ostatním metodám. Metody LP a Soft L1, Naivní model [medián/průměr, $k = 5, \sigma = 2$] vrací v tomto pořadí nejlepší výsledky. Detail včetně standardní odchylky modelů je vidět na grafu 4.8. Metoda LP má rozptyl významně nejnižší.



Obrázek 4.8: Porovnání implementací - Středně velké e-shopy (Relativní RMSE)

Průměrnou dobu běhu jednotlivých implementací znázorňuje graf 4.9. Stále jde i v nejhorších případech o časy pod 1s, nicméně se zde začíná projevovat horší asymptotická složitost komplexnějších optimalizačních algoritmů použitých při výpočtech Soft L1 a LP metod.

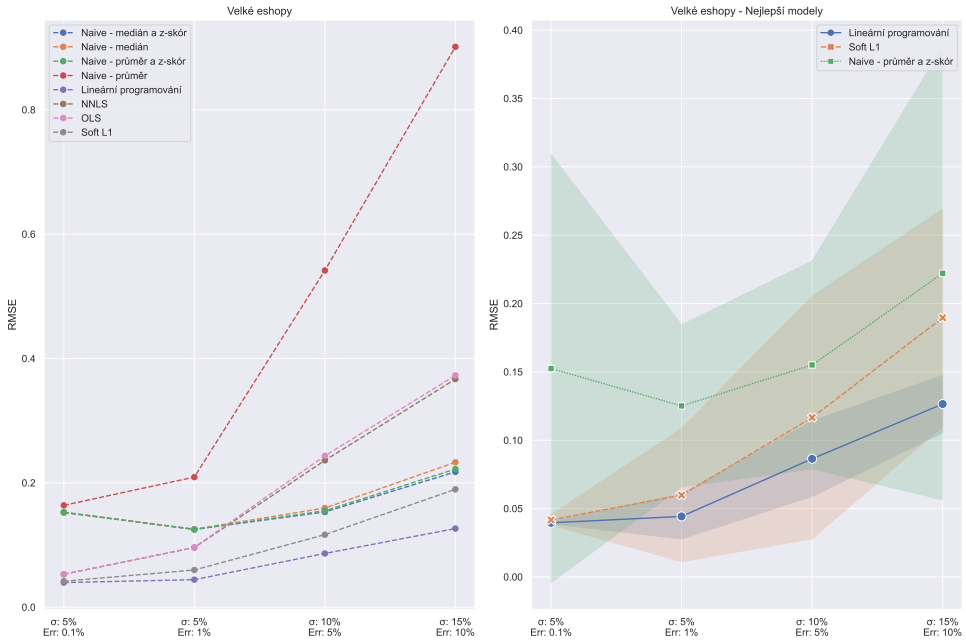


Obrázek 4.9: Doba běhu - Středně velké e-shopy

4.4.3 Velké e-shopy

Datasety velkých e-shopů se vyznačují vysokým počtem produktů, ale i řádově větším počtem objednávek. V testovacích datech byl tento poměr zvolen 1:150 (300 produktů, 45 000), znovu šlo o pesimistický odhad, s narůstajícím počtem objednávek lze očekávat pouze zpřesnění odhadů díky většímu počtu dostupných měření.

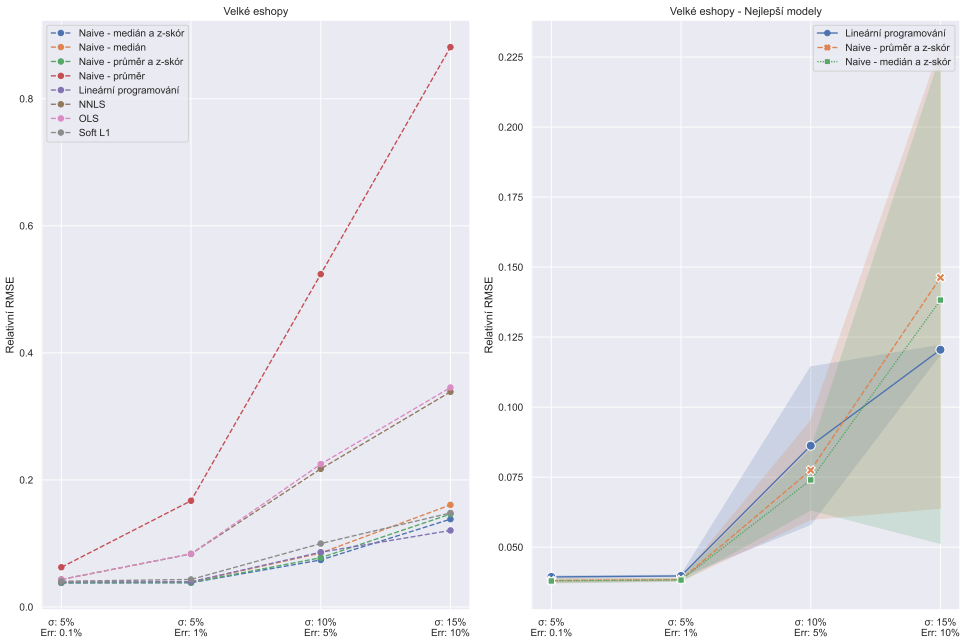
Graf 4.10 ukazuje průměrnou chybovost (RMSE) jednotlivých metod na vygenerovaných datech. Detail srovnání 3 nejlepších metod je vidět na grafu 4.10 vpravo. Pásy okolo jednotlivých metod reprezentují standardní odchylku.



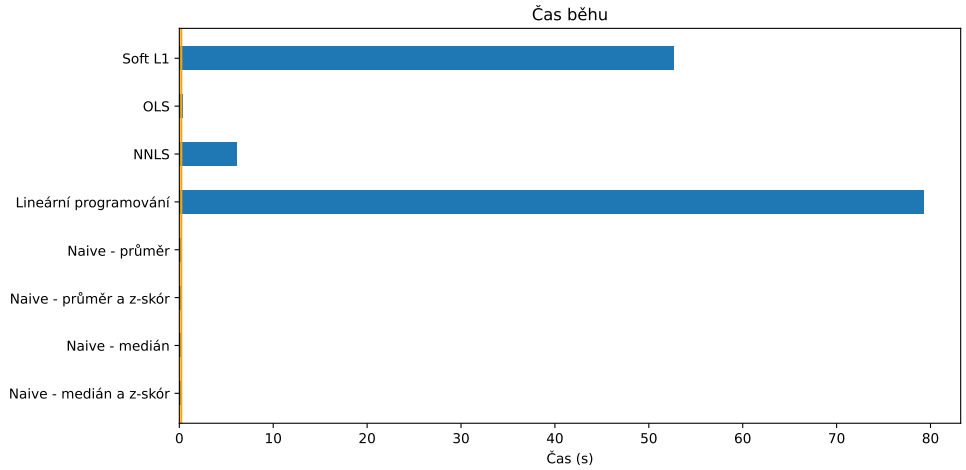
Obrázek 4.10: Porovnání implementací - Velké e-shopy

Výsledky jsou velmi podobné měřením na datech středních e-shopů. Všechny modely zaznamenaly sníženou chybovost díky většímu počtu měření na vstupu, naivní modely byly schopné odhadnout přes 98% produktů. Nejlepších výsledků dosahuje i v tomto případě metoda *Lineárního programování* (3.3).

Při porovnání „relativního RMSE“ z grafu 4.11 je vidět, že výsledky LP metody a naivních metod [průměr/medián, $k = 5, \sigma = 2$] jsou prakticky stejné, v některých případech dokonce dosahují implementace naivního modelu lepších výsledků. Vzhledem obrovskému nárůstu času nutnému na na vyřešení úlohy pomocí LP (70–80s) a Soft L1 (50–60s) metod (porovnání je znázorněno na grafu 4.12) je jasnou volbou některá z konfigurací naivního modelu.



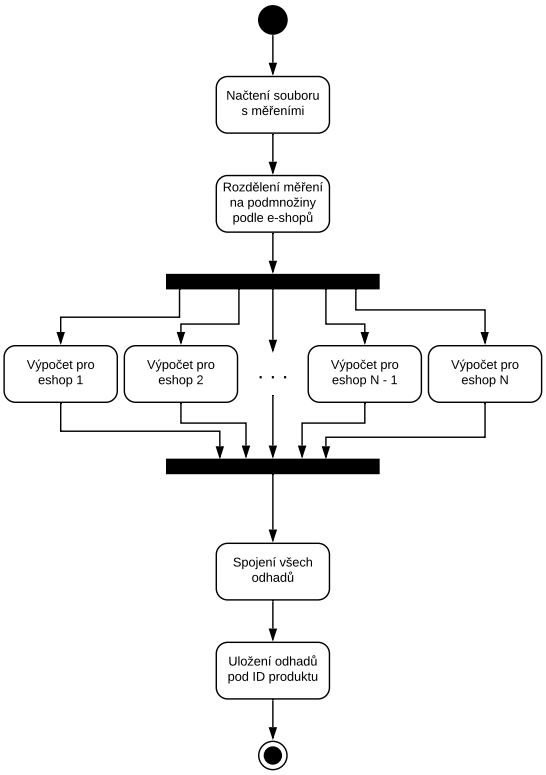
Obrázek 4.11: Porovnání implementací - Velké e-shopy (Relativní RMSE)



Obrázek 4.12: Doba běhu - Velké e-shopy

4.5 Výsledný návrh

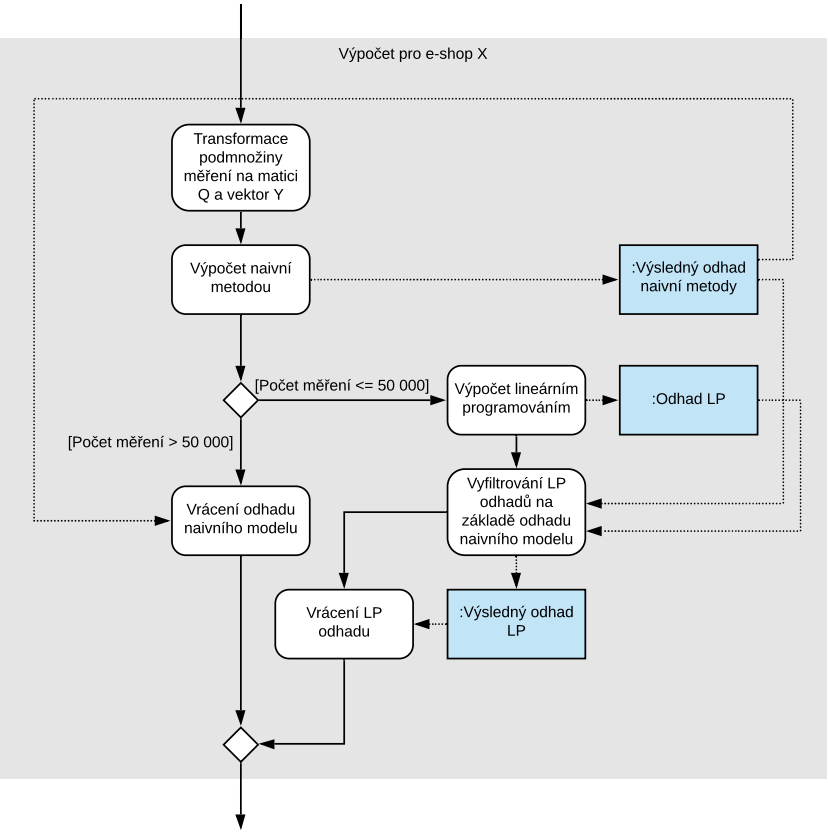
Na základě porovnání implementací byl navržen a implementován výsledný prototyp aplikace odhadující váhy produktů. Diagram 4.13 znázorňuje průchod aplikací. Vstupem je soubor s váženími popsaný v kapitole 2.2. Data jsou následně rozdělena do skupin podle e-shopů a samotné výpočty vah jsou pak paralelizovány na úrovni e-shopu dle počtu dostupných vláken CPU.



Obrázek 4.13: Průchod aplikací

Detail výpočtu popisuje diagram 4.14. Naivní model (4.3.1) v konfiguraci [Průměr, $k = 5, \sigma = 2$] provádí odhad vždy. Pro e-shopy s méně než 50 000 měřeními je následně spuštěn LP model (4.3.4), který dodává přesnější odhady za cenu delšího běhu výpočtu.

Pro zajištění existence reprezentativního vzorku měření (viz. kapitola 3.4) se následně z odhadů naivního modelu vytvoří bitová maska, která se aplikuje na odhady LP modelu. Přijdeme tak o odhady produktů, které naivní model nebyl schopný dopočítat, na druhou stranu se ale zvýší důvěryhodnost odhadů.



Obrázek 4.14: Průchod aplikací - výpočet

Závěr

Hlavním cílem této práce bylo vytvoření nástroje, který z dat o vahách objednávek (zjištěných váhami na dopravníku) odhadne předem neznámé váhy jednotlivých produktů v objednávkách obsažených. Jedním z požadavků na řešení byla schopnost vypořádat se s chybnými měřeními vyskytujícími se v datech. V rámci práce byl problém nejprve zformalizován, následně byly navrženy jednotlivé přístupy a konkrétní metody řešení problému na základě rešerše. V průběhu celého procesu byl brána ohled i na konkrétní strukturu dat dodaných k zadání této práce.

S cílem umožnit porovnání jednotlivých metod byl vyvinut konfigurovatelný generátor testovacích datových sad. Všechny navržené metody řešení byly implementovány a otestovány na sadě takto vygenerovaných dat s rozdílnými charakteristikami. Na základě tohoto srovnání byl implementován již zmíněný nástroj odhadující váhy produktů, který kombinuje dvě metody pro své rozdílné charakteristiky, a to Naivní metodu a Lineární programování.

Cíle práce byly naplněny. Při použití odhadů vah produktů získaných finálním nástrojem bylo dosaženo 61.5% úspěšnosti při vážení objednávek na kontrolní váze (bez změny konfigurace), což je oproti původnímu manuálnímu vážení produktů nárůst o 10%.[21] Při doplnění odhadů o odhady z manuálního vážení úspěšnost vzrostla na 67.5%.

Navržené řešení předpokládá pravidelnou rekalibraci měřících vah a nebere tedy v potaz chybu způsobenou fyzickými změnami váhy v čase. Korekce této chyby je podnětem k dalšímu zkoumání.[22]

[21] Výsledek byl změřen na datech z období 1. 3. 2021 - 20. 4. 2021

[22] S krátkodobými výkyvy se navrhované řešení vypořádat dokáže, se systematickou změnou v procesu vážení nikoliv. V takovém případě je doporučeno sesbírat nová data.

Literatura

[1] Sheffield, S.: Probability and Random Variables 18.600: Lecture 22 [online]. 2020, [cit. 4.3.2021]. Dostupné z: http://math.mit.edu/~sheffield/2018600/Lecture22.pdf

[2] Tango, T.: Linear equations with random variables. *Statistics in Medicine*, 2004. Dostupné z: https://doi.org/10.1002/sim.1969

[3] Loh, P.: Convexity [online]. 2013, [cit. 21.3.2021]. Dostupné z: https://www.math.cmu.edu/~ploh/docs/math/mop2013/convexity-soln.pdf

[4] Klouda, K.; Vašata, D.: BI-VZD – Vytěžování znalostí z dat - Lineární regrese (přednáška 7) [online]. 2020, [cit. 2.2.2021], K dispozici po přihlášení do sítě ČVUT. Dostupné z: https://courses.fit.cvut.cz/BI-VZD/@B201/lectures/files/BI-VZD-07-cs-handout.pdf

[5] Intel: *LAPACK gelsd function [online]*. 2013, [cit. 19.2.2021]. Dostupné z: https://software.intel.com/sites/products/documentation/doclib/mkl_sa/11/mkl_lapack_examples/_gelsd.htm

[6] Golub, G. H.; Van Loan, C. F.: *Matrix computations*. The Johns Hopkins University Press, fourth edition vydání, 2013, ISBN 978-1-4214-0794-4.

[7] Gill, P. E.; Wong, E.: Methods for Convex and General Quadratic Programming. *Math. Prog. Comp. 7*, 2014. Dostupné z: https://doi.org/10.1007/s12532-014-0075-x

[8] Vavasis, S. A.: Quadratic programming is in NP. *Information Processing Letters*, ročník 36, 1990: s. 73–77, ISSN 0020-0190. Dostupné z: https://www.sciencedirect.com/science/article/pii/002001909090100C

[9] SciPy community: scipy.optimize.least_squares [online]. 2021, [cit. 25.3.2021]. Dostupné z: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html

[10] Jiří, R.: Lineární programování (stručný učební text) [online]. 2002, [cit. 10.1.2021]. Dostupné z: http://uivtx.cs.cas.cz/~rohn/publist/skripta.pdf

[11] Megiddo, N.: On the complexity of linear programming. In *Advances in economic theory: Fifth world congress*, Cambridge University Press London, 1987, s. 225–268.

[12] Khachiyan, L.: Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, ročník 20, 1980: s. 53–72, ISSN 0041-5553. Dostupné z: https://www.sciencedirect.com/science/article/pii/0041555380900610

[13] Schwertman, N. C.; aj.: A Simple Noncalculus Proof That the Median Minimizes the Sum of the Absolute Deviations. *The American Statistician*, ročník 44, 1990: s. 38–39. Dostupné z: https://amstat.tandfonline.com/doi/abs/10.1080/00031305.1990.10475690

[14] Botchkarev, A.: A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, ročník 14, 2019, ISSN 1555-1237. Dostupné z: http://dx.doi.org/10.28945/4184

[15] Piatetsky, G.: Python leads the 11 top Data Science, Machine Learning platforms: Trends and Analysis [online]. 2019, [cit. 3.4.2021]. Dostupné z: https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html

[16] Budíková, M.; aj.: Statistika II: Kapitola 3, Diagnostické grafy a testy normality dat [online]. 2005, [cit. 1.4.2021]. Dostupné z: https://is.muni.cz/el/1456/podzim2005/CN_KMSTII/um/kapitola3.pdf

[17] Huang, Q.: Lecture 7: CS395T Numerical Optimization for Graphics and AI - Trust Region Methods [online]. [cit. 10.5.2021]. Dostupné z: https://www.cs.utexas.edu/~huangqx/2017_CS395T_Numerical_Optimization_Lecture_7.pdf

[18] Mitchell, S.; aj.: PuLP: A Linear Programming Toolkit for Python [online]. 2011, [cit. 29.4.2021]. Dostupné z: http://www.optimization-online.org/DB_FILE/2011/09/3178.pdf

[19] John Forrest, R. L.-H.: *CBC User Guide [online]*. 2014, [cit. 27.3.2021]. Dostupné z: https://doi.org/10.1287/educ.1053.0020

[20] Gearhart, J. L.; aj.: Comparison of open-source linear programming solvers [online]. 2013, [cit. 4.5.2021]. Dostupné z: https://www.osti.gov/biblio/1104761

Příloha **A**

Seznam použitých zkratek

LP Lineární programování

MAE Střední absolutní chyba (Mean Absolute Error)

(M)ILP (Smíšené) celočíselné programování

MSE Střední kvadratická chyba (Mean Squared Error)

NNLS Metoda nejmenších čtverců omezená na interval $[0; \infty)$ (Non-negative Least Squares)

OLS Metoda nejmenších čtverců (Ordinary Least Squares)

RMSE Odmocnina ze střední kvadratické chyby (Root Mean Squared Error)

UUIDv4 Universally Unique Identifier verze 4

Příloha **B**

Obsah přiloženého média

- `README.md` ... návod ke spuštění projektu
- `data` ... datové sady
  - `generated` ... vygenerované datové sady
  - `measurements` ... výsledky jednotlivých metod na různých datech
  - `weight_data_2021_02_01_2021_04_06.csv` ... produkční data
- `data_analysis` ... analýza produkčních dat a výsledků
  - `graphs` ... vygenerované grafy z analýzy
  - `Model Results Comparison.ipynb` ... analýza výsledků
  - `Production Data Analysis.ipynb` ... analýza produkčních dat
- `src` ... zdrojové kódy
  - `app`
    - `sample_weight_generator` ... zdrojový kód generátoru dat
      - `__main__.py` ... vstupní bod generátoru
    - `weight_derivation` ... zdrojový kód nástroje
      - `notebooks` ... Jupyter notebooky s porovnáním metod
      - `__main__.py` ... vstupní bod nástroje
- `tests` ... zdrojové kódy testů
- `text` ... text práce
  - `thesis` ... zdrojová forma práce ve formátu LaTeX
  - `thesis.pdf` ... text práce ve formátu PDF