FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Optimalizace přiřazení klientů k zaměstnancům
Student:	Filip Novák
Vedoucí:	Ing. Jan Motl
Studijní program:	Informatika
Studijní obor:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	Do konce zimního semestru 2019/20

Pokyny pro vypracování

Maximalizujte vážené párování v bipartitním grafu na poskytnutých datech. Protože se ale jedná o data z průmyslu, je třeba se vypořádat s následujícími komplikacemi:
1) Předzpracování dat.
2) Odhad vah v bipartitním grafu pomocí metod učení s učitelem.
3) Vysokou výpočetní složitostí, která může být neslučitelná s požadavkem na rychlost zpracování velkého množství dat při nasazení.
4) Byznysové zadání může vyžadovat modifikaci optimalizační funkce, respektive přidání omezujících podmínek.

Úkoly:
1) Popište byznysové zadání.
2) Vytvořte přehled literatury, zabývající se problematikou.
3) Porovnejte několik různých algoritmů na poskytnutých datech.
4) Vyhodnoťte dosažené výsledky na daných datech na základě očekávaného finančního zisku dle byznysového zadání, výpočetního času a paměťové náročnosti.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Karel Klouda, Ph.D.
vedoucí katedry

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

V Praze dne 2. března 2018

Poděkování

Chtěl bych poděkovat panu Ing. Janu Motlovi za odborné vedení práce a cenné rady při konzultacích, které mi pomohly tuto práci zkompletovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2018 ……

Odkaz na tuto práci

Novák, Filip. *Optimalizace přiřazení klientů k zaměstnancům.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Bakalářská práce se zabývá optimalizací oslovení zákazníka společnosti, která nabízí srovnání dodavatelů energetických komodit.

Zákazníci společnosti jsou nejdříve klasifikováni podle pravděpodobnosti, že se společností uzavřou smlouvu na základě vyplněných hodnot ve webovém formuláři. Následně se práce zabývá optimalizací přiřazení operátora call-centra ke konkrétnímu zákazníkovi.

Práce popisuje byznysový problém a analyzuje možné způsoby řešení klasifikace zákazníků pomocí metody „Gradient Boosting“, umělé neuronové sítě a pomocí logistické regrese. V optimalizačních metodách se práce zaměřuje na analýzu celočíselného lineárního programování, maďarského algoritmu a také hladové *greedy* optimalizace.

Jednotlivé vybrané metody jsou mezi sebou porovnány. Konečné řešení problému, kterým se tato práce zabývá, je demonstrováno na použití *Gradient Boostingu* jako klasifikátoru a celočíselného lineárního programovaní pro optimalizaci přiřazení operátora k zákazníkovi.

Toto zvolené řešení je porovnáno s aktuální metodou, kterou společnost v současnosti používá.

Klíčová slova optimalizace, klasifikace, celočíselné lineární programování, Gradient Boosting, problém přiřazení, call-centrum

Abstract

The aim of the bachelor thesis is the optimization of the assignment problem between an operator and a customer of a call-centre. The company operating call-centre offers comparison of suppliers of energy commodities like natural gas and electricity.

Company's customers are first classified based on the probability that the client will sign a contract with the specific operator. The classification is based on the customer's current supplier and company's historical data. Subsequently, the thesis deals with the optimization of the assignment of the operator to a customer.

The thesis describes a business problem and analyses possible ways of solving customer classifications by Gradient Boosting, Artificial Neural Networks and Logistic Regression. In the optimization methods, the thesis focuses on the analysis of Integer Linear Programming, the Hungarian Algorithm and Greedy Optimization.

The selected methods are compared to each other. The final solution to the problem that this thesis deals with is demonstrated using Gradient Boosting as a classifier and Integer Linear Programming as an optimizer for the assignment problem.

This solution is compared to the current method used by the company.

Keywords optimization, classification, Integer Linear Programming, Gradient Boosting, assignment problem, call-centre

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Vytěžování znalostí z dat a jejich správná analýza dnes vytváří hranici, která rozděluje mnohé společnosti na ty úspěšné a naopak ty neúspěšné. Call-centra patří mezi ty společnosti, které jsou právě těmito analýzami dotčeny nejvíce. Správná metoda oslovení zákazníků tvoří v mnoha případech klíčovou část fungování celého byznysového procesu.

Při oslovení zákazníků se nabízejí dvě klíčové otázky:

1. Kteří zákazníci call-centra mají být osloveni?

2. Který operátor má oslovit vybraného zákazníka?

Existuje celá řada metod, jak k problémům vyplývajících z těchto otázek přistoupit. Klasifikační i optimalizační metody jsou známy již několik desítek let, přesto až dnes mnoho společností pochopilo jejich důležitost a potřebu nasadit je do reálného prostředí.

Celý proces správného oslovení zákazníka začíná sběrem dat, ze kterých se dá zákazník poměrně přesně klasifikovat a s určitou pravděpodobností se dá předpovědět, zda tento zákazník se společností smlouvu uzavře, či nikoliv. Samotná klasifikace zákazníků je v případě této bakalářské práce provedena na základě dat získaných z vyplněných webových formulářů. Optimalizační metoda potom již ohodnocené zákazníky přiřadí ke konkrétnímu operátorovi call-centra tak, aby byla pravděpodobnost uzavření smlouvy co nejvyšší.

Tato bakalářská práce si klade za cíl popsat možné způsoby řešení tohoto problému a následně i demonstrovat použitelnost metod, které v analytické části dosáhly nejlepších výsledků, na praktickém příkladu.

Práce je teoreticky-aplikačního charakteru. V teoretické části jsou analyzovány možné způsoby řešení jak klasifikace zákazníků, tak i samotné optimalizace přiřazení operátora k zákazníkovi. V praktické části je potom definován byznysový problém a vstupní data z call-centra. Na těchto datech jsou následně vybrané metody porovnány. Řešení tzv. *assignment problému*, kterým

se tato práce zabývá, je demonstrováno na použití *Gradient Boostingu* jako klasifikátoru a celočíselného lineárního programovaní pro optimalizaci přiřazení. Poslední část bakalářské práce se zabývá porovnáním zvoleného řešení s aktuální metodou oslovování zákazníků společnosti.

Cíle práce

Cílem této práce je analýza řešení efektivního oslovování zákazníků call-centra na základě historických dat z databáze za účelem zvýšení finančního zisku společnosti. Podrobněji jsou jednotlivé cíle práce vytyčeny následovně:

- analýza současných možností řešení klasifikace zákazníků a optimalizace přiřazení operátora a zákazníka;
- rešerše literatury zabývající se klasifikací a optimalizací;
- popsání byznysového problému call-centra;
- porovnání zvolených klasifikačních algoritmů;
- analýza optimalizačních metod;
- demonstrace použitelnosti metod, které dosáhly nejlepších výsledků;
- analýza navrhovaného řešení.

Kapitola 1

Teoretická část

Možností, jak efektivně řešit oslovování zákazníků call-centra existuje celá řada. Tato práce se zaměřuje na využití moderních postupů vytěžování znalostí z dat. Práce se zabývá klasifikačními a optimalizačními metodami.

Cílem této kapitoly je teoretická analýza možností řešení problému klasifikace zákazníků a následné optimalizace přiřazení operátora ke konkrétnímu zákazníkovi.

Celý proces pro správné oslovení začíná sběrem dat, které při klasifikaci využijeme. Vzhledem k různým možnostem, jak data získat a možným chybám, které mohly vzniknout například selháním systému, či překlepem uživatele, je nutné data nejdříve předzpracovat. Této části se věnuje sekce 1.1. Dále jsou v této kapitole v sekci 1.2 a 1.3 popsány možnosti predikování, že zvolený zákazník s call-centrem uzavře smlouvu, respektive možnosti optimalizace přiřazení operátora ke konkrétnímu zákazníkovi.

1.1 Předzpracování dat

K tomu, abychom vůbec mohli provést nějakou klasifikaci, musíme mít nejenom data připravená v určité formě, ale musíme také zajistit, že tyto data budou odpovídat skutečnostem. V případě, že data budou obsahovat mnoho chybných údajů, znehodnotíme celý výsledek klasifikačních a následně i optimalizačních metod. Prvním krokem ke správnému vytěžování znalostí z dat je tedy samotné předzpracování dat.

Předzpracování dat můžeme rozdělit do čtyř kategorií:

1. čištění dat,
2. datová integrace,
3. datová transformace,
4. datová redukce.

Následující kapitoly se věnují jednotlivým kategoriím podrobněji.

1.1.1 Čištění dat

Čištění dat (angl. *Data Cleaning*) bývá často kategorií velmi podceňovanou a přeskakovanou. Přitom se jedná o velmi kritickou část celého vytěžování znalostí z dat. Přeskočení této části může vést k velmi nepřesným výsledkům, které můžou být velmi snadno nesprávně interpretovány.

Při procesu získávání a skladování dat se velmi často stane, že se některé údaje, ať už vlivem lidského faktoru (chybně zadaný údaj, překlep, …), či technické chybě, stanou nevalidní, zkreslené. S těmito daty je potom potřeba naložit takovým způsobem, abychom z nich důležitou informaci neztratili, ale zároveň nesmíme výsledek ovlivnit právě těmito chybami.

Hledání chyb v již uložených datech bývá často velmi pracná záležitost, a proto je mnohem lepší, když veškerá data získávána přímo od uživatelů prochází kontrolou systému, který nevalidní data nepřijme. V případě, že již tuto možnost nemůžeme využít, existují metody mimo manuálního prohledávání, jak chyby v datech (tzv. *outliery*) identifikovat. Podle V. Chandoly můžeme jednotlivé metody hledání outlierů rozdělit podle použití:[1]

- klasifikátoru,
- shlukové analýzy,
- algoritmu k-nejbližších sousedů,
- statistických metod.

Druhým problémem čištění dat nastává v případě, kdy mnoho hodnot u vybraného příznaků nebylo zadáno, případně naměřeno. S tímto problémem se můžeme vypořádat následovně:[2]

- chybějící data vložíme ručně;
- úplně odstraníme instance s chybějícími hodnotami;
- hodnotu příznaku nahradíme všemi možnými hodnotami (*multiple imputation*);
- hodnotu příznaku nahradíme průměrem/modem;
- hodnotu příznaku nahradíme průměrem/modem k-nejbližších instancí;
- k doplnění chybějící hodnoty použijeme klasifikační model.

1.1.2 Datová integrace

Data jsou extrahována z databází, datových skladů, souborů a různých jiných zdrojů, které nám přináší jakoukoliv věcnou informaci o zkoumaném problému. Pokud chceme provést klasifikaci, tak musíme data vhodně agregovat do datasetu, který je pro zvolený model vhodný. Vzhledem k tomu, že data můžou být ve strukturované i nestrukturované formě, je někdy skoro nemožné data v přípustném čase propojit do jednoho uceleného celku. Pro ulehčení datové integrace (angl. *Data Integration*) vzniklo několik softwarů jako například Informatica PowerCenter[1] nebo Talend Data Management Platform[2].

1.1.3 Datová transformace

Při datové transformaci (angl. *Data Transformation*) jsou jednotlivé hodnoty atributů transformovány do formy vhodné pro konkrétní klasifikátory. Data můžeme převést do jiné podoby využitím následujících metod:

- normalizace dat,
- diskretizace,
- konverze typu atributu,
- agregace dat,
- zobecnění dat (např. převedení názvu ulice a města na krajinu).

Takto převedená data mohou být klasifikátor snadněji pochopeny, což často zapříčiní zvýšení přesnosti klasifikátoru.

1.1.4 Datová redukce

Dat můžeme mít hodně jak z hlediska celkového objemu (počtu instancí), tak i z hlediska velikosti dimenze (počtu příznaků). Provádění klasifikace na velmi objemných datasetech může trvat neúnosně dlouhou dobu. Zároveň velikost dat dělají model méně čitelný, ale hlavně zbytečné příznaky můžou mít i negativní vliv na přesnost klasifikace. [3] Proto je v případě velkých datasetů nutné provést datovou redukci (angl. *Data Reduction*).

Počet instancí můžeme redukovat pomocí výběru reprezentativního vzorku. Složitější část tvoří výběr příznaků, na kterých budeme klasifikátor trénovat a následně i provádět klasifikaci. K tomu nám mohou pomoci heuristické metody, evoluční metody, nebo jiné statistické metody.[3]

[1] `https://www.informatica.com/products/data-integration/powercenter.html`
[2] `https://www.talend.com/products/data-integration/data-management-platform/`

1.2 Přehled klasifikačních metod

Tato sekce popisuje možné metody, které lze aplikovat na klasifikaci zákazníků. Klasifikační problémy řeší přiřazení pozorování do správné kategorie na základě trénovacích dat, na kterých jsme klasifikátor naučili. Při těchto problémech máme k dispozici data se správně klasifikovanými případy (v případě našeho problému to jsou zákazníci, kteří se společností uzavřeli smlouvu) a z těchto dat se snažíme předpovědět budoucí situace – jedná se tedy o tzv. metodu učení s učitelem.

Klasifikaci můžeme rozdělit na základě počtu kategorií, do kterých pozorování může patřit na:

- binární – pozorování patří pouze do jedné ze dvou tříd;
- diskrétní – počet tříd, do kterých může pozorování patřit je větší nebo rovno dvěma.

Tato bakalářská práce se zabývá binární klasifikací – zákazník smlouvu podepíše či nikoliv. Neexistuje klasifikační metoda, která by byla nejlepší pro každý problém (tzv. *no free lunch theorem*), což jako první popsal David H. Wolpert spolu s Williamem G. Macreadym [4]. Z tohoto důvodu je potřeba jednotlivé klasifikační metody mezi sebou spravedlivě porovnávat.

Základní metody pro porovnání jednotlivých klasifikátorů můžeme rozdělit podle způsobu provádění ohodnocení klasifikátoru na dvě skupiny a to:

- podle matice záměn,
- na základě měřených křivek.

Matice záměn binárního klasifikátoru je zobrazena v tabulce 1.1.

		skutečnost	
		ANO	NE
klasifikace	ANO	TP	FP
	NE	FN	TN

Tabulka 1.1: Matice záměn

V této matici jsou obsaženy informace týkající se počtu správně/nesprávně klasifikovaných objektů, přičemž:

- *true positives* (TP) – klasifikováno správně jako ANO;
- *true negatives*(TN) – klasifikováno správně jako NE;
- *false positives* (FP) – klasifikováno jako ANO, ve skutečnosti NE;
- *false negatives* (FN) – klasifikováno jako NE, ve skutečnosti ANO.

Na základě těchto definovaných pojmů, můžeme u klasifikátoru měřit následující skóre:

- *true positive rate* (*recall*, *sensitivity*): $RC = \frac{TP}{TP + FN}$;
- *true negative rate* (*specificity*): $TNr = \frac{TN}{TN + FP}$;
- *false positive rate* (*fall-out*): $FPr = \frac{FP}{FP + TN}$;
- *false negative rate* (*miss rate*): $FNr = \frac{FN}{FN + TP}$;
- *accuracy*: $AC = \frac{TP + TN}{TP + TN + FP + FN}$;
- *precision*: $PC = \frac{TP}{TP + FP}$;
- *F-Measure* (F_1 *score*): $F_1 = 2 \cdot \frac{PR \cdot RC}{PR + RC}$.

Druhou skupinu, podle které lze porovnávat jednotlivé klasifikátory, jsou křivky, které ukazují vybrané skóre v závislosti na prahu. Nejznámější křivkou je tzv. *Receiver Operating Characteristic Curve*, neboli ROC křivka. Tato křivka ukazuje vztah mezi *false positive rate* a *true positive rate*. V ideálním případě bude křivka stoupat svisle vzhůru. Naopak v případě, kdy klasifikátor jednotlivé skupiny klasifikuje zcela náhodně, bude křivka stoupat po uhlopříčce.

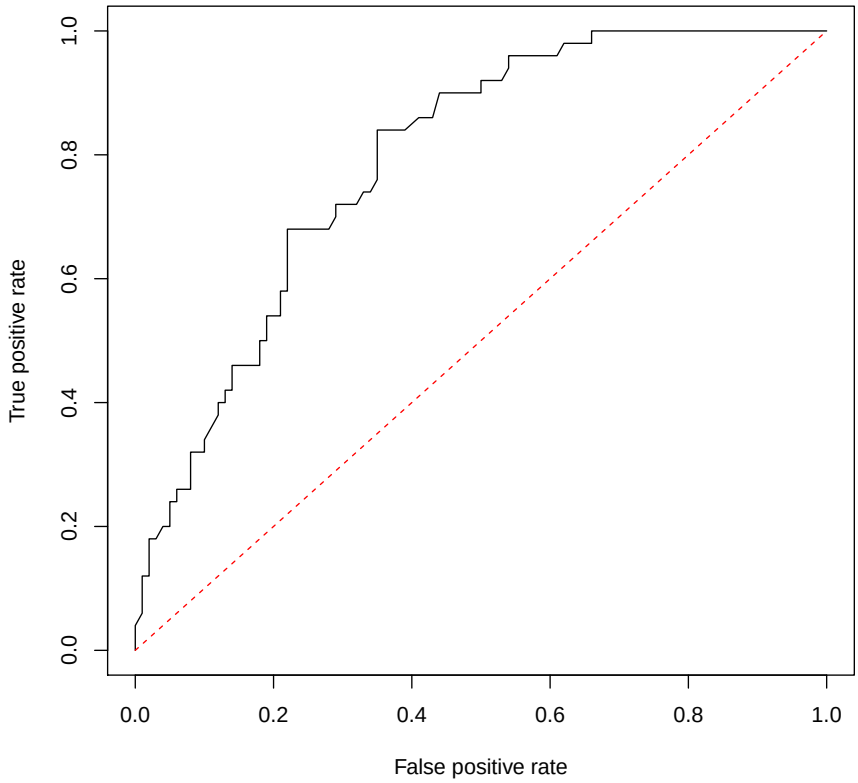
Ukázka ROC křivky je zobrazena na grafu 1.1. Na tomto grafu je černou barvou zobrazena ROC křivka modelu logistické regrese, který klasifikuje květinu *Iris versicolor*. Červenou přerušovanou čárou je na tomto grafu zobrazen náhodný klasifikátor. V případě ROC křivky nás může zajímat plocha pod křivkou *Area Under Curve* (AUC). Obecně platí, čím větší je AUC, tím lepších výsledků model dosahuje.

Alternativou k ROC křivce je tzv. *Precision Recall Curve* - PR křivka. Tato křivka, jak její název napovídá, ukazuje vztah mezi *recall* a *precision*. Plocha pod PR křivkou se potom nazývá *Area Under Precision Recall Curve* (AUPRC).

Dostupných klasifikátorů a jejich různých modifikací existuje celá řada, v následujících sekcích jsou uvedeny pouze ty vybrané.

1.2.1 Rozhodovací stromy

Rozhodovací stromy patří mezi jednu z nejoblíbenějších metod pro klasifikaci a to hlavně kvůli své interpretovatelnosti. Zároveň jsou rozhodovací stromy při

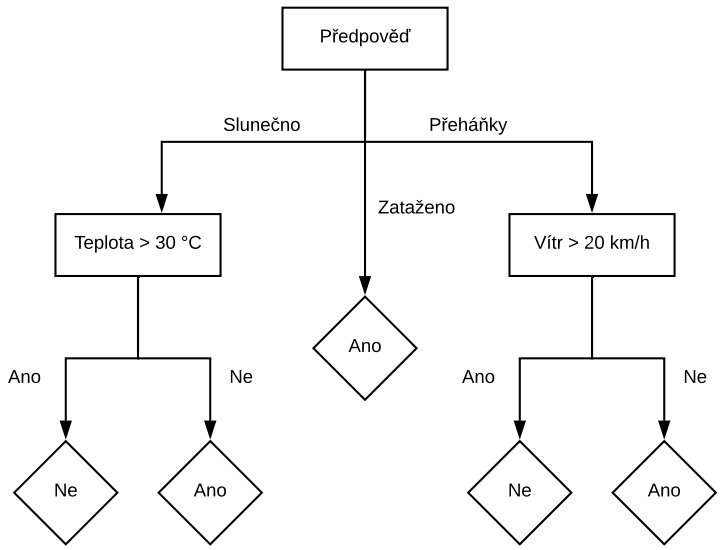


Obrázek 1.1: ROC křivka logistické regrese klasifikující květinu *Iris versicolor*

klasifikaci poměrně odolné vůči nedokonalým datům a nevyžadují tak pracné předzpracování dat.

Rozhodovací strom je kořenový strom, kde uzly tohoto stromu představují jednotlivé atributy. Z každého uzlu vede několik hran. Tyto hrany udávají jednotlivé hodnoty, kterých atribut nabývá (v případě spojitých dat můžeme volit *treshold*, podle kterého hodnoty dělíme do skupin). Listy tohoto stromu jsou potom ohodnoceny výsledným *labelem*. Cesta od kořene k listům tvoří soubor rozhodovacích pravidel. Příklad jednoduchého rozhodovacího stromu je uveden na obrázku 1.2.

Metod pro sestavování rozhodovacích stromů existuje několik, nejčastěji se však pořadí jednotlivých atributů vybírá podle míry informačního zisku. Atribut, který má informační zisk nejvyšší je vybrán jako kořen. Z kořene vedeme hrany tak, abychom co nejvíce rozdělily instance na jednotlivé kategorie. Tyto hrany vytvoří podmnožiny trénovacího datasetu. Rekurzivně zjistíme informační zisk pro každou nově vytvořenou skupinu. Rekurze je ukončena pokud



Obrázek 1.2: Rozhodovací strom, který určuje, zda se fotbalové utkání odehraje

jsou naplněný následující podmínky:

1. ve skupině máme všechna data z jedné třídy;
2. neexistují další atributy, podle kterých můžeme data dále rozdělit;
3. skupina je prázdná.

V případě podmínky 2 a 3 je skupina (uzel) převedena na list, jehož hodnotou je majoritní třída. [5]

Vzhledem ke konečnému množství hran, které můžou z jednotlivých uzlů vést mají rozhodovací stromy problém se spojitými daty, u kterých nemusí být zřejmé, jak je distribuovat do jednotlivých kategorií. Další nevýhodou rozhodovacích stromů je náchylnost k velké změně celého modelu i při malé změně dat. Tento problém se úspěšně daří eliminovat použitím regresních lesů nebo obecných metod *bagging*, či *boosting*, které ale zpravidla využívají rozhodovacích stromů. Princip těchto metod je založený na shlukování jednotlivých stromů, které sami o sobě tvoří nespolehlivý model. Kombinace těchto stromů však dohromady vytváří model velmi robustní.

1.2.2 Lineární regrese

Lineární regrese je matematická metoda, která modeluje vztah mezi závislou proměnnou $\mathbf{y}$ a jednou nebo více nezávislými proměnnými $\mathbf{x}$. Tento vztah můžeme popsat rovnicí:

$$y_i = \beta_0 + \beta_1 \cdot x_i + \varepsilon_i \quad \text{pro všechna } i \in \{0, ..., n\}$$

kde β_0 a β_1 jsou parametry ε je reziduální odchylka a n je počet vzorků. Tuto rovnici potom nazýváme regresním modelem, případně regresní rovnicí. Odhady parametrů β_0 a β_1 provádíme pomocí metody nejmenších čtverců v případě, že jsou splněny následující podmínky:

$$\begin{aligned}
&E(\varepsilon_i) = 0 && \text{pro všechna } i \in \{0, ..., n\} \\
&var(\varepsilon_i) = \sigma^2 && \text{pro všechna } i \in \{0, ..., n\} \\
&cov(\varepsilon_i, \varepsilon_j) = 0 && \text{pro všechna } i \neq j, \text{ kde } i \in \{0, ..., n\}, j \in \{0, ..., n\} \\
&\varepsilon_i \sim N(0, \sigma^2) && \text{pro všechna } i \in \{0, ..., n\}
\end{aligned}$$

V případě, že máme pouze jednu nezávislou proměnnou x, mluvíme o jednoduché regresi. V případě více proměnných mluvíme o vícenásobné regresi. Lineární regrese, a obecně regresní modely, predikují spojitou proměnnou. V případě klasifikace však můžou regresní modely předpovídat pravděpodobnost, že určitá situace nastane. Vzhledem k tomu, že proměnná $\mathbf{y}$ v lineární regresi může nabývat záporných hodnot, nebo hodnot větších než 1, tak byla zavedena logistická regrese. Proměnná $\mathbf{y}$ v logistické regresi nabývá hodnot mezi 0 a 1. Proto se pro klasifikační problémy používá zpravidla logistická regrese.

Ukázka jednoduché lineární regrese je na obrázku 1.3. Červenou barvou jsou na tomto obrázku vyznačeny reziduální odchylky. Na lineární regresi můžeme nahlížet jako na proložení bodového diagramu přímkou.

1.2.3 Umělá neuronová síť

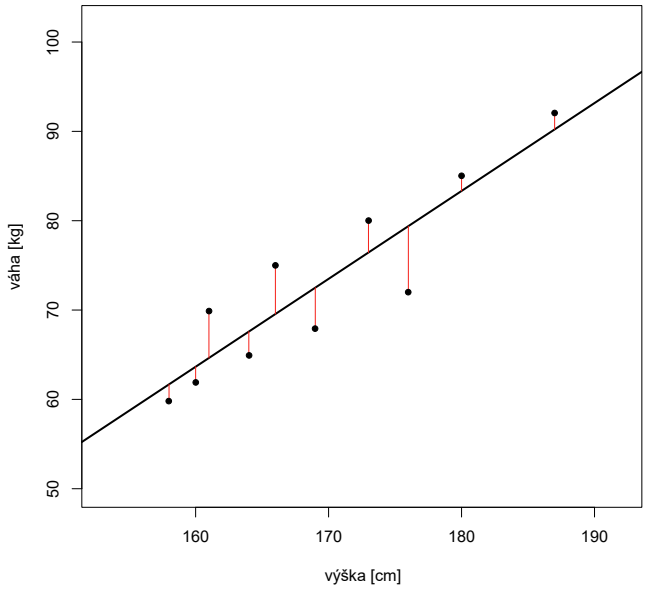
Umělá neuronová síť je matematický model, který vychází z biologické předlohy neuronové sítě nacházející se v mozku živočichů.

Stejně jako biologický vzor tvoří základ umělé neuronové sítě neurony. Výběžky neuronů tvoří dendrity, které přijímají vstupní informace a naopak axony, které pomocí synapsí informace předávají dál. Analogicky také umělé neurony (také nazývány perceptrony) přijímají jeden nebo více vstupů, které jsou transformovány na jeden výstup. Modelů popisující umělý neuron existuje více, následující rovnice popisuje výstup perceptronu Y s n vstupy popsaného McCullochem a Pittsem [6].

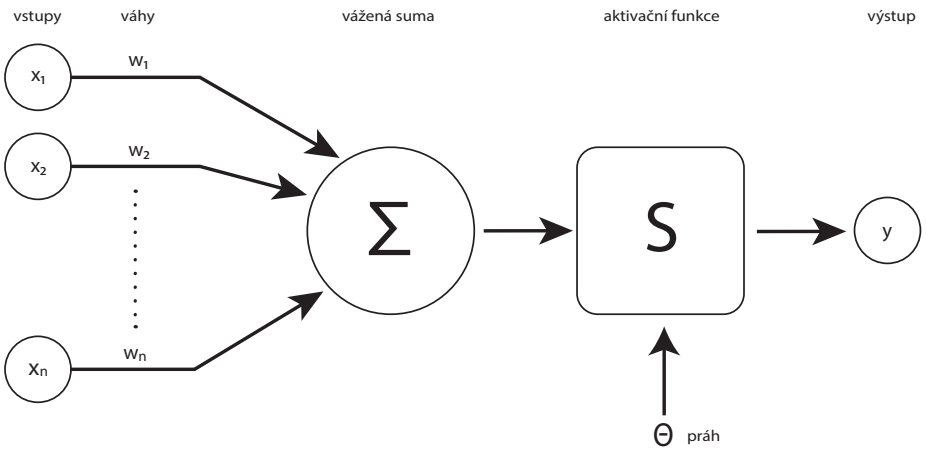
$$Y = S(\sum_{i=1}^{n}(w_i x_i) + \Theta)$$

kde $S(x)$ je aktivační funkce, x_i jsou vstupy, w_i jsou váhy a Θ značí práh perceptronu. Grafické znázornění perceptronu je na obrázku 1.4

Podle řešeného problému je nutné zvolit vhodnou aktivační funkci. Mezi nejpoužívanější aktivační funkce patří skoková aktivační funkce, sigmoidální



Obrázek 1.3: Použití lineární regrese na změřené hodnoty výšky a váhy



Obrázek 1.4: Grafické znázornění umělého neuronu (perceptronu) - *vlastní tvorba*

aktivační funkce nebo například často využívaný tzv. *RELU* (*Rectifier)* [7]. Dalším důležitým krokem k správnému fungování umělé neuronové sítě je nastavení vah a nastavení prahu. Toto nastavení můžeme provést jak pomocí

metody s učitelem, tak i pomocí metody bez učitele.

Skupina navzájem propojených umělých neuronů tvoří umělou neuronovou síť. Jednotlivých typů neuronových sítí je celá řada. Mezi ty nejznámější patří Vícevrstvé neuronové sítě, Samoorganizující se sítě nebo například Hopfieldovy sítě. Neuronové sítě se v dnešní době využívají zejména k rozpoznávání obrazů či zvuků.

1.3 Přehled optimalizačních metod

Cílem této sekce je popsat metody, které je možné použít pro řešení optimalizace přiřazení operátora ke konkrétnímu klientovi. Jedná se o klasický optimalizační problém - tzv. *assignment problem*, který neformálně můžeme definovat jako:

Předpokládejme, že máme n zaměstnanců a n pracovních úkolů. Pro každý pár (zaměstnanec, pracovní úkol) známe plat, který musíme zaměstnanci zaplatit, aby úkol provedl. Cílem je dokončit veškeré úkoly tak, abychom minimalizovali náklady. Podmínkou je, že každý zaměstnanec může provést pouze jeden úkol a každý úkol může být proveden maximálně jedním zaměstnancem.

Formální, matematická, definice *assignment problemu* popsaného výše vypadá následovně:

$$\text{minimize } \mathbf{Z} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

kde zároveň platí následující podmínky:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{pro všechna } i \in \{1, ..., n\} \tag{1.1}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{pro všechna } j \in \{1, ..., n\} \tag{1.2}$$

$$x_{ij} \in \{0, 1\} \qquad \text{pro všechna } i, j \in \{1, ..., n\}$$

za předpokladu, že:

- prvek c_{ij} cenové matice $\boldsymbol{C}$ o rozměru $n \times n$ určuje plat i-tého zaměstnance za provedení j-tého úkolu;
- prvek x_{ij} binární matice $\boldsymbol{X}$ o rozměru $n \times n$ určuje zda i-tý zaměstnanec provede j-tý úkol (1), či nikoliv (0). [8]

Podmínka 1.1 znamená, že každý zaměstnanec může provést právě jeden úkol, podmínka 1.2 potom omezuje, že jeden úkol může být vykonán právě jedním zaměstnancem.

Assignment problem v případě call-centra znamená, že v cenové matici $\boldsymbol{C}$ nejsou uchovány informace ohledně výdělku zaměstnance (v tomto případě operátora) za určitou práci, ale jeho koeficient s konkrétním zákazníkem. Tento koeficient je napočítaný pomocí klasifikátoru, který dvojice operátor-klient ohodnotil a určuje pravděpodobnost, že klient s vybraným operátorem uzavře smlouvu. Problém je tedy nutné vhodně otočit a sumu maximalizovat. Tato definice formuluje problém přiřazení jako celočíselný problém.

Řešení problému přiřazení hrubou silou problému přiřazení má složitost $\mathcal{O}(n!)$. Algoritmické řešení lze rozdělit na dvě kategorie:

- přesné - vždy vrátí optimální řešení
- aproximační - nemusí najít optimum

Vzhledem k tomu, že *assignment problem* je NP-težký problém [9], algoritmicky řešitelný se složitostí $\mathcal{O}(n^3)$ [10], tak aproximační metody bývají často využívány vzhledem k mnohem rychlejšímu výpočtu.

Následující text popisuje základní metody řešení tohoto problému.

1.3.1 Celočíselné lineární programování

Vhledem k tomu, že *Assignment problem* je lineární problém, je možné jej řešit lineárním programováním. Celočíselné lineární programování (anglicky *Integer Linear Programming*) je podmnožinou celočíselného programování. Jedná se o optimalizační metodu, která řeší problém ve tvaru:

$$\text{maximize} \quad \mathbf{c}^T\mathbf{x}$$

Množinu řešení potom popisují rovnice ve tvaru:

$$\boldsymbol{A}\mathbf{x} \leq \mathbf{b}$$
$$\mathbf{x} \geq \mathbf{0}$$
$$\mathbf{x} \in \mathbb{Z}^n$$

kde $\boldsymbol{A}$ je matice rozměru $m \times n$ s celočíselnými prvky, $\mathbf{b}$ je m-rozměrný vektor a $\mathbf{c}$, $\mathbf{x}$ jsou n-rozměrné vektory. Speciálním typem celočíselného programování je tzv. „0-1 programování“, kde navíc platí podmínka:

$$\mathbf{x} \in \{0, 1\}^n$$

Celočíselné programování se řadí mezi NP-těžký problém [11]. Mezi nejznámější NP-těžké problémy, které lze převést na celočíselné programování patří například „problém obchodního cestujícího“, „problém splnitelnosti“ (*SAT Problem*) nebo například „problém batohu“ (*Knapsack problem*).

K řešení celočíselného problému můžeme použít několik metod. Mezi nejpoužívanější však patří „metoda sečných nadrovin“ (*Cutting-plane method*) [12] a potom „metody *branch and bound*“ [13], které jsou založené na stromových strukturách.

1.3.2 Maďarský algoritmus

Maďarský algoritmus (angl. *Hungarian algorithm* nebo také *Hungarian method*) je optimalizační metoda, která řeší problém přiřazení v polynomiálním čase [14]. Je zaručené, že výsledkem maďarské metody bude vždy optimální řešení.

Pro použití maďarské metody je nutné, aby byl rozměr matice $n \times n$. Zároveň musí být její prvky nezáporné. V případě, že počet zaměstnanců není roven počtu úkolů, je potřeba matici vhodně rozšířit - při maximalizačním problému nulou, při minimalizačním problému doplníme matici hodnotou vyšší než je nejvyšší hodnota z matice tak, abychom nenarušili optimální výsledek.

Maďarský algoritmus obsahuje pět základních kroků: [14]

1. redukce řádku - od každého prvku v řádku odečteme nejnižší hodnotu tohoto řádku;

2. redukce sloupce - od každého prvku v sloupci odečteme nejnižší hodnotu tohoto sloupce;

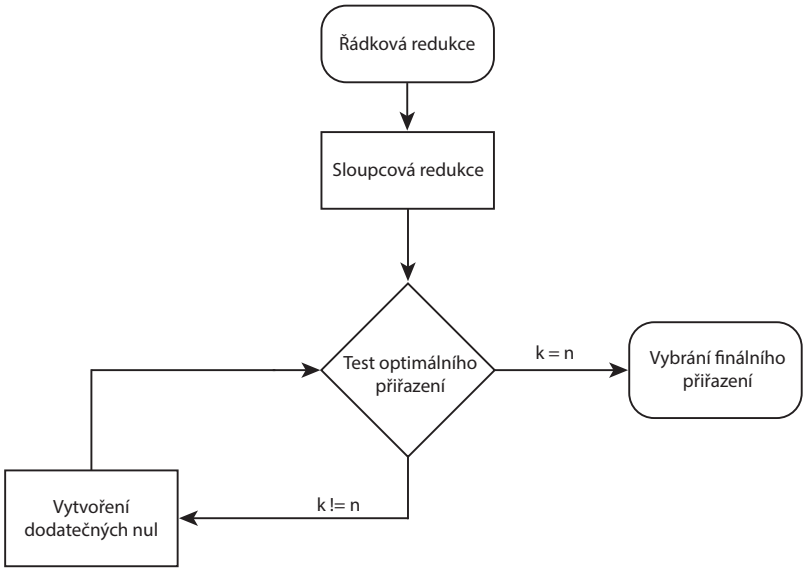
3. test optimálního přiřazení - zakryjeme všechny nuly minimálním počtem horizontálních a vertikálních čar (krycí čáry), pokud je počet těchto použitých čar roven n, tak pokračujeme krokem 5;

4. vytvoření dodatečných nul - nalezneme nejnižší hodnotu, která není zakrytá krycí čárou a odečteme ji od všech nezakrytých prvků, zároveň tuto hodnotu přičteme ke všech prvkům, které jsou zakryté horizontální i vertikální čárou zároveň, pokračujeme krokem 3;

5. vybrání finálního přiřazení - vybereme n nul, kde každý sloupec a řádek obsahuje právě jednu vybranou nulu, původní hodnoty na pozicích těchto vybraných nul reprezentují optimální přiřazení.

Výše popsané kroky jsou ještě názorně zobrazeny vývojovým diagramem na obrázku 1.5.

Níže je uveden praktický příklad použití Maďarské metody na základním problému přiřazení. Předpokládejme 4 zaměstnance, kterým mají být přiřazeny 4 pracovní úkoly. U každého zaměstnance známe výdaje (na cestu, mzdu…), které musíme vynaložit, aby danou práci vykonal. Tyto výdaje jsou uložené v matici 1.3, kde v řádku jsou jednotliví zaměstnanci a ve sloupci jsou jednotlivé úkoly. Naší snahou je minimalizovat celkové náklady na splnění všech čtyřech úkolů. K vyřešení problému použijeme Maďarskou metodu. Matice 1.4 zobrazuje matici po provedení řádkové redukce a matice 1.5 potom zobrazuje matici po provedení sloupcové redukce.



Obrázek 1.5: Vývojový diagram zachycující kroky Maďarského algoritmu

$$\begin{pmatrix} 5 & 15 & 10 & 20 \\ 15 & 5 & 20 & 15 \\ 20 & 5 & 20 & 15 \\ 20 & 5 & 25 & 15 \end{pmatrix} \quad (1.3)$$

$$\begin{pmatrix} 0 & 10 & 5 & 15 \\ 10 & 0 & 15 & 10 \\ 15 & 0 & 15 & 10 \\ 15 & 0 & 20 & 10 \end{pmatrix} \quad (1.4)$$

$$\begin{pmatrix} 0 & 10 & 0 & 5 \\ 10 & 0 & 10 & 0 \\ 15 & 0 & 10 & 0 \\ 15 & 0 & 15 & 0 \end{pmatrix} \quad (1.5)$$

Nyní provedeme test optimálního přiřazení. Položme k rovno minimálnímu počtu horizontálních či vertikálních krycích čar, kterými zakryjeme všechny nuly v matici. Tyto krycí čáry jsou zobrazeny v matici 1.6. V tomto případě je k rovno 3 a vzhledem k tomu, že $k \neq n$, tak musíme vytvořit dodatečné nuly. Nejnižší nezakrytá hodnota je 10. Tuto hodnotu tedy odečteme od všech nezakrytých hodnot a zároveň ji přičteme k prvkům, které jsou zakryty jak horizontální, tak i vertikální krycí čárou. Matice po provedení těchto úprav je zobrazena v matici 1.7. Opět provedeme test optimálního přiřazení, které je názorně zobrazeno v matici 1.8. Tentokrát dostaneme $k = n = 4$, čímž můžeme algoritmus zastavit a vyhodnotit optimální přiřazení.

$$\begin{pmatrix} 0 & 10 & 0 & 5 \\ 10 & 0 & 10 & 0 \\ 15 & 0 & 10 & 0 \\ 15 & 0 & 15 & 0 \end{pmatrix} \quad (1.6)$$

$$\begin{pmatrix} 0 & 20 & 0 & 15 \\ 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 \\ 5 & 0 & 5 & 0 \end{pmatrix} \quad (1.7)$$

$$\begin{pmatrix} 0 & 20 & 0 & 15 \\ 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 \\ 5 & 0 & 5 & 0 \end{pmatrix} \quad (1.8)$$

Cílem je vybrat n nul, jejíchž řádek či sloupec se nevyskytuje v již takto vybrané nule. Příkladem takto vybraných nul jsou zelenou barvou označené nuly v matici 1.9. Tyto vybrané pozice vyjadřují optimální hodnoty v původní matici 1.3. V případě, že můžeme zvolit více takovýchto nul, tak nezáleží na pořadí, jelikož každé takto zvolené řešení bude optimem.

$$\begin{pmatrix} 0 & 15 & 5 & 10 \\ 0 & 0 & 5 & 0 \\ 15 & 5 & 0 & 0 \\ 10 & 0 & 0 & 0 \end{pmatrix} \quad (1.9)$$

$$\begin{pmatrix} 5 & 15 & 10 & 20 \\ 15 & 5 & 20 & 15 \\ 20 & 5 & 20 & 15 \\ 20 & 5 & 25 & 15 \end{pmatrix} \quad (1.10)$$

Velkou nevýhodou maďarské metody je nutnost matici doplnit na čtvercovou. V mnoha případech je potom problém výpočetně velmi náročný a tato metoda se proto stává nepoužitelná. Časová složitost maďarské metody je $\mathcal{O}(n^3)$. [10]

1.3.3 Hladový aproximační algoritmus

Hladový algoritmus (angl. *Greedy Approximation Algorithm*) pro řešení lineárního problému přiřazení je optimalizační metoda, která se zakládá na rychlejším nalezení řešení problému přiřazení. Jedná se však pouze o aproximační metodu, což znamená, že nalezené řešení nemusí být optimálním přiřazením.

Tento hladový algoritmus je založený na nalezení minimálního nezakrytého prvku v cenové matici. Tento vybraný prvek si zapamatujeme a zakryjeme jeho řádek a sloupec. Tento postup opakujeme dokud nejsou zakryty všechny prvky cenové matice. Výsledným přiřazením hladového algoritmu je přiřazení právě těchto zvolených minimální prvků.

Ukázka hladového algoritmu je zobrazena níže. Zelenou barvou jsou zvýrazněné vybrané minimální prvky cenové matice. V případě, že existuje více minimálních prvků, tak výsledkem hladové metody můžou být různá přiřazení. Cenová matice je v tomto příkladu stejná jako v případě maďarské metody v sekci 1.3.2. Výsledné přiřazení však není optimální.

$$\begin{pmatrix} 5 & 15 & 10 & 20 \\ 15 & 5 & 20 & 15 \\ 20 & 5 & 20 & 15 \\ 20 & 5 & 25 & 15 \end{pmatrix} \quad (1.11)$$

$$\begin{pmatrix} \mathbf{5} & 15 & 10 & 20 \\ 15 & 5 & 20 & 15 \\ 20 & 5 & 20 & 15 \\ 20 & 5 & 25 & 15 \end{pmatrix} \quad (1.12)$$

$$\begin{pmatrix} \mathbf{5} & 15 & 10 & 20 \\ 15 & \mathbf{5} & 20 & 15 \\ 20 & 5 & 20 & 15 \\ 20 & 5 & 25 & 15 \end{pmatrix} \quad (1.13)$$

$$\begin{pmatrix} \mathbf{5} & 15 & 10 & 20 \\ 15 & \mathbf{5} & 20 & 15 \\ 20 & 5 & 20 & \mathbf{15} \\ 20 & 5 & 25 & 15 \end{pmatrix} \quad (1.14)$$

$$\begin{pmatrix} \mathbf{5} & 15 & 10 & 20 \\ 15 & \mathbf{5} & 20 & 15 \\ 20 & 5 & 20 & \mathbf{15} \\ 20 & 5 & \mathbf{25} & 15 \end{pmatrix} \quad (1.15)$$

Časová složitost hladového algoritmu je $\mathcal{O}(n^2 \log n)$. Můžeme také odhadnout aproximační faktor α, který ukazuje, jak daleko je hladový algoritmus od nalezení optimálního přiřazení. Tento faktor můžeme odhadnout jako poměr minimální ceny přiřazení zkoumaného algoritmu ku optimální minimální ceně přiřazení. Přičemž platí, že α_g hladového algoritmu:

$$\alpha_g = \frac{H_n}{H_n^{(2)}}$$

kde H_n je optimální minimální cena přiřazení. Pro porovnání je aproximační faktor pro náhodné přiřazení roven: [15]

$$\alpha_r = \frac{n}{H_n^{(2)}}$$

Kapitola 2

Analytická část

V této kapitole je nejdříve podrobně definován praktický problém, kterým se teoretická část této bakalářské práce zabývá. Dále jsou zde popsány data, na kterých je problém přiřazení zaměstnanců call-centra ke klientům v práci řešen, spolu s výstupními požadavky společnosti, která call-centrum provozuje.

Tato kapitola se také zaměřuje na praktickou ukázku srovnání klasifikátorů jednotlivých klientů call-centra a optimalizační metody, která přiřadí zákazníka ke konkrétnímu operátorovi.

V sekci 2.3 této kapitoly je potom popsán postup předzpracování dat společnosti *XYZ* tak, aby byla tato data použitelná při trénování klasifikátoru. Sekce 2.4 a 2.5 se věnují volbě klasifikátoru, resp. optimalizační metody.

Vzhledem k citlivosti dat, jsou veškerá data anonymizována. Za tímto účelem bude v celé práci využita fiktivní firma *XYZ*.

2.1 Definice byznysového problému

Společnost *XYZ* je firma zabývající se srovnáním produktů v oblasti energetiky. V rámci této aktivity provozuje společnost několik call-center. Operátoři call-centra oslovují zákazníky, kteří vyplní na webových stránkách formulář s aktuálními daty týkajícími se jejich lokality, současného dodavatele a spotřeby plynu, případně elektrické energie. Klienti, kteří odešlou takto vyplněný formulář jsou následně oslovováni zaměstnanci call-centra. Podmínkou oslovení zákazníka je existující výhodnější nabídka, kterou může operátor klientovi nabídnout. Jakmile operátor osloví vybraného zákazníka, tak tento vybraný operátor je se zákazníkem spárován až do doby, než nedojde k ukončení spolupráce (zákazník se rozhodne pro změnu, nebo nikoliv). V případě, že se oslovený zákazník rozhodne pro změnu, která mu je ze strany operátora nabídnuta, tak z uskutečněného obchodu získá společnost *XYZ* určitý zisk.

Vzhledem k rostoucímu počtu klientů a omezeného množství aktivních operátorů je potřeba efektivně vybírat klienty, u kterých je vysoká pravdě-

podobnost zvolení si navržené nabídky. Někteří operátoři mají navíc větší přehled a zkušenosti s konkrétními produkty, případně mají lepší *know-how*, jak klienty oslovit, a proto mají vyšší pravděpodobnost úspěšného oslovení specifických klientů.

Cílem této práce je nalézt efektivní metodu splňující vyplývající požadavky výše uvedeného problému společnosti *XYZ*, a to konkrétně:

- klasifikovat zákazníky podle pravděpodobnosti, že se kladně vyjádří pro navrhovanou změnu;
- optimalizovat přiřazení zákazníka ke konkrétnímu operátorovi.

Pro řešení jsou k dispozici historická data z databáze společnosti *XYZ*, která obsahují informace z vyplněných webových formulářů přímo od zákazníků a také informace o operátorech. Tato data jsou podrobněji statisticky popsána v sekci 2.2.

Cílem této práce je demonstrovat použitelnost navrhovaných metod v reálném prostředí, nikoliv sestavení kompletní aplikace, která by zvolenou klasifikační a optimalizační metodu propojovala v jeden komplexní program nasazený přímo do call-centra společnosti.

2.2 Vstupní data

K řešení problému je k dispozici databáze společnosti *XYZ*, ve které jsou uchována veškerá historická data spojená s obchodními aktivitami od 1. 9. 2015 do 30. 9. 2017.

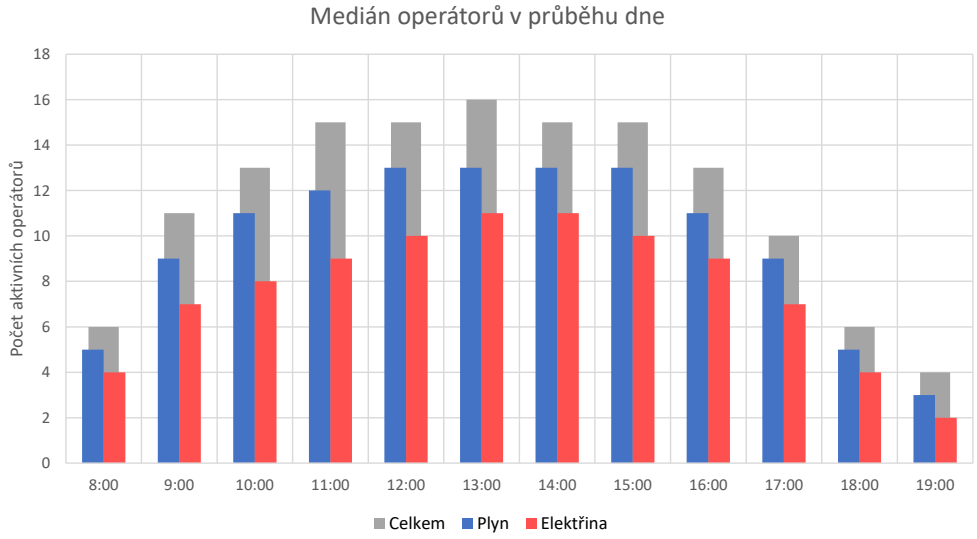
V databázi společnosti jsou evidovány základní informace o zákaznících, jejich historické aktivity se společností *XYZ*, vyplněné formuláře s aktuální spotřebou plynu, případně elektrické energie a jejich současné měsíční výdaje na tyto energie. Dále jsou v databázi uchovávány informace o operátorech společnosti, jejich uskutečněných telefonních hovorech s klienty a samozřejmě i stav jednotlivých objednávek. Objednávkou je ve smyslu této bakalářské práce myšlen jakýkoliv validně odeslaný formulář zákazníka přes webové rozhraní, obsahující jeho aktuální stav s energiemi a žádostí o kontaktování s výhodnější nabídkou ze strany společnosti *XYZ*.

V databázi je za tuto dobu celkem evidováno 235 380 vyplněných formulářů a 180 619 unikátních klientů. Z těchto formulářů bylo 31 440 objednávek evidováno jako úspěšné (tj. klient se rozhodl pro nabízenou změnu), což je 13,36 %.

Dále je evidováno 1 099 operátorů, z čehož je 285 vedeno jako aktivních. Za aktivního operátora se považuje operátor, který má se společností uzavřenou platnou smlouvu. Operátoři se celkem pokusili oslovit zákazníky 921 514×, z čehož 466 072 hovorů bylo uskutečněno (50,58 %).

Medián aktivních operátorů je graficky zobrazen v grafu 2.1. Vzhledem k tomu, že se operátoři nespecializují na jednotlivé komodity, ale zákazníky

oslovují jak s nabídkami na elektrickou energii, tak i na plyn zároveň, je v tomto grafu šedivou barvou vyznačený celkový průměrný počet operátorů v danou hodinu v call-centru. Červenou a modrou barvou jsou potom označeni operátoři, kteří provedli alespoň jeden telefonát týkající se dané kategorie. Medián nových klientů v průběhu pracovního dne je potom zobrazen v grafu 2.2. Novým zákazníkem je v tomto případě myšlena vytvořená objednávka přes webový formulář od takového klienta, který společnost *XYZ* v minulosti ještě neoslovil.



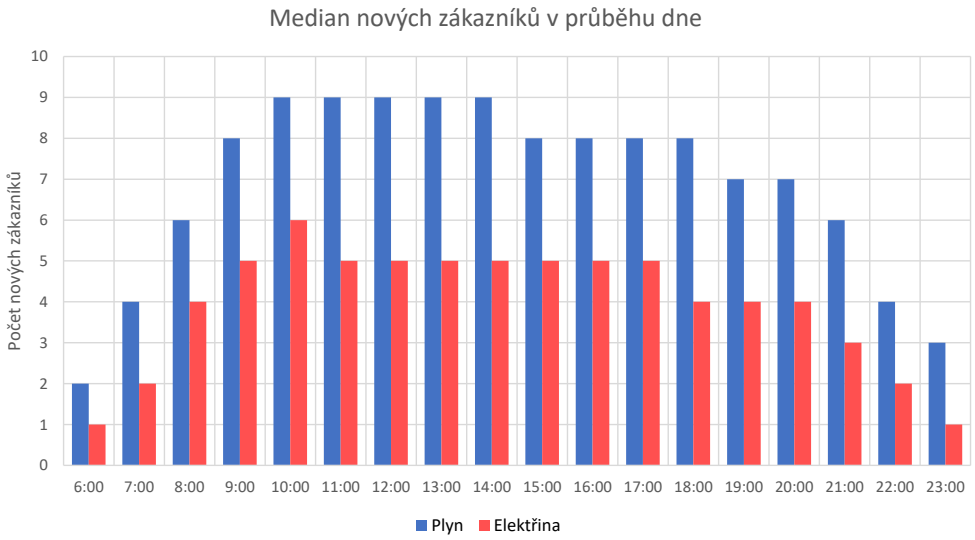
Obrázek 2.1: Medián aktivních operátorů v průběhu dne

Průměrná provize za nového klienta je vyšší v případě plynu. Graficky je zobrazena v grafu 2.3, průměrná provize za nového klienta v případě elektrické energie potom na grafu 2.4. Provize, kterou společnost *XYZ* za uzavření smlouvy získá, je závislá na aktuálním stavu spotřeby elektrické energie, případně plynu klienta a navrhované nabídce ze strany společnosti.

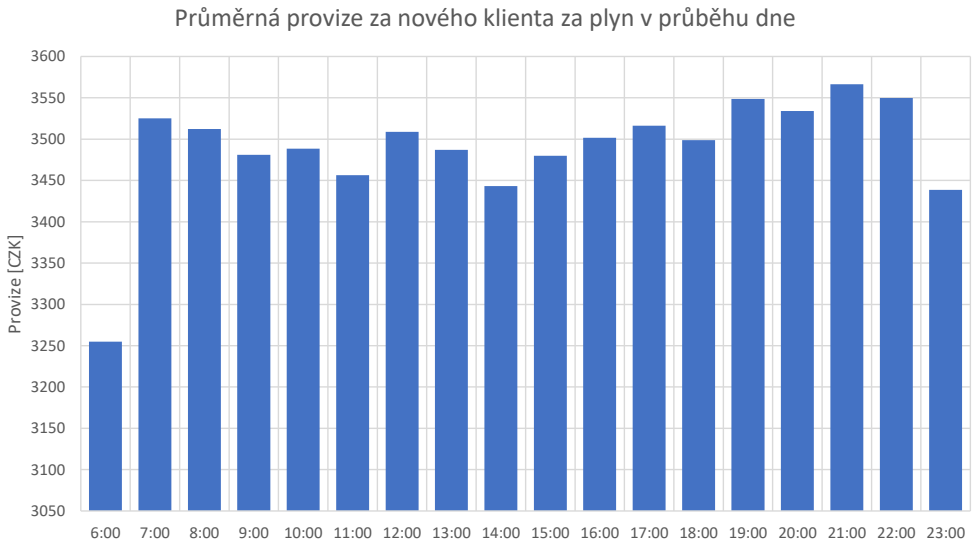
Z této analýzy vyplývá, že dat pro vypracování požadovaného řešení je dostatek. Vzhledem k tomu, že jsou data získávána přímo od uživatelů přes webové rozhraní, tak jsou některá data poškozená či nepřesná. Proto je nutné mimo sestavení trénovacího datasetu provést i validaci dat. Validaci dat a podrobnému popisu trénovacího datasetu, podle kterého jsou zákazníci klasifikováni, se věnuje sekce .

2.3 Předzpracování dat

Nejdůležitější data, která použijeme pro klasifikaci klientů, společnosti *XYZ* jsou získávána přímo od zákazníků společnosti. Tito zákazníci odešlou vypl-

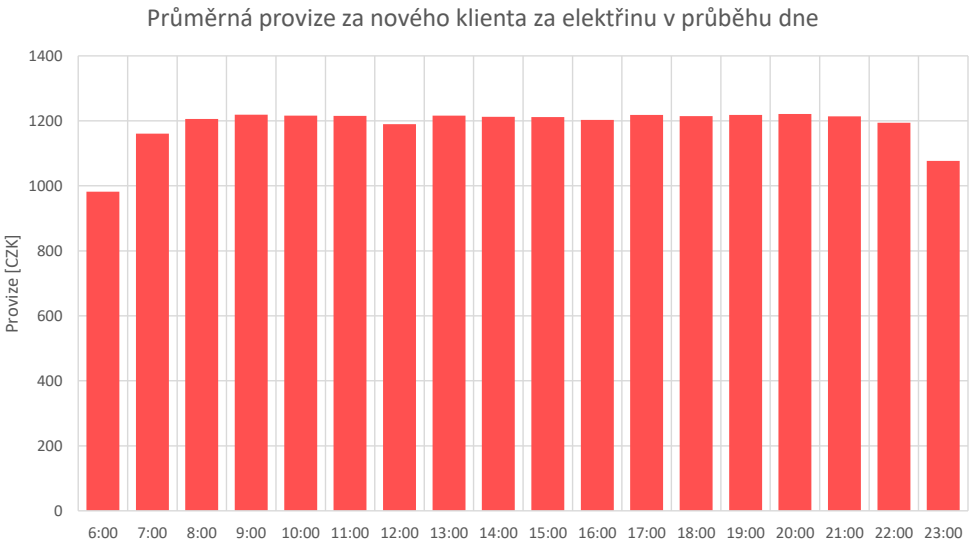


Obrázek 2.2: Medián nových zákazníků v průběhu dne



Obrázek 2.3: Průměrná provize za nového klienta v průběhu dne - plyn

něný formulář s informacemi ohledně jejich lokality, způsobu použití vybrané komodity, současné spotřebě plynu, případně elektrické energie a dalších požadovaných údajů přes webové rozhraní. Tato data jsou následně uložena do relační *MySQL* databáze společnosti. V této databázi jsou současně uloženy informace ohledně jednotlivých objednávkách a operátorech společnosti.



Obrázek 2.4: Průměrná provize za nového klienta v průběhu dne - elektřina

2.3.1 Čištění dat

Validaci dat získaných přímo od uživatele zajišťuje webové rozhraní, které kontroluje uživatelem vložená data. Pokud jsou data nevalidní, nebo neúplná, tak systém klientovi neumožní odeslat formulář až do doby, než jsou všechny vyžadované údaje vyplněné, nebo dokud uživatel neopraví nevalidní údaj (např. záporná spotřeba elektrické energie). Tento proces velmi ulehčuje fázi čištění dat pro správnou klasifikaci zákazníků.

V databázi společnosti je zavedena historizace jednotlivých objednávek. Pro správnou klasifikaci musíme zajistit, že uvedená data týkající se objednávek budou nejaktuálnější pro každého zákazníka. V databázi tedy musíme ignorovat historické verze objednávek a případně některé důležité údaje agregovat podle zákazníků (viz kalkulované atributy v sekci 2.3.3). K eliminaci historických objednávek nám poslouží základní *SQL* příkazy.

Pro klasifikaci je také nutné nastavit tzv. *label*, podle kterého budeme klasifikovat. V případě problému oslovení nás zajímá, zda zákazník smlouvu nakonec se společností *XYZ* uzavřel, či nikoliv. K zjištění této skutečnosti nám pomůže příznak *status_id*, který obsahuje informace o stavu objednávky. V případě, že objednávka byla ve stavu, kdy $status\_id = 35$, tak klient se společností smlouvu uzavřel a *label* tedy nastavíme na 1. Ve všech ostatních případech nastavíme *label* na 0.

2.3.2 Datová integrace

Veškerá data je potřeba agregovat do jednoho datasetu, nad kterým budeme klasifikaci provádět. V případě společnosti *XYZ* jsou všechna data uložena v jedné relační databázi. Data tak nemusíme propojovat z různých databází, datových kostek, nebo například textových souborů a vystačíme si tedy s jednoduchými *SQL* příkazy.

Na základě klíčů spojíme jednotlivé relační tabulky, které obsahují požadované příznaky a z vybraných atributů vytvoříme trénovací dataset. Zvolené příznaky byly vybrány po konzultaci s vedoucím práce. Tyto vybrané atributy, tvořící finální dataset jsou popsány v příloze v tabulce B.1.

2.3.3 Datová transformace

V případě, že chceme do klasifikace zahrnout časový trend – chování zákazníka mohlo být jiné v roce 2015 než v roce 2017, můžeme rozdělit jednotlivé objednávky do přibližně rovnoměrných skupin podle data vytvoření objednávky. K tomuto účelu vytvoříme nový atribut *time_created_fold*, který rozděluje objednávky dle data vytvoření do dvanácti skupin. Ve validaci potom můžeme tento atribut nastavit jako tzv. *batch*, což nám zajistí, že jednotlivé objednávky budeme klasifikovat v omezeném časovém období.

V této práci nás u každého zákazníka zajímají pouze nejaktuálnější objednávky, přesto můžeme využít některá historická data vybraného zákazníka. Konkrétně nás může zajímat, zda vybraný zákazník byl již někdy v minulosti společností *XYZ* osloven. V případě že ano, zajímá nás, jak jeho objednávka dopadla. Za tímto účelem vytvoříme jak pro plyn, tak i pro elektrickou energii nové atributy:

- storno_count – počet dříve stornovaných objednávek;
- successful_count – počet dříve úspěšně dokončených objednávek (zákazník se společností uzavřel smlouvu);
- last_order – stav poslední zákazníkovi objednávky.

Tyto vytvořené (kalkulované) atributy jsou již zahrnuty v tabulce B.1 a mají u názvu atributu označeny dvěma hvězdičkami.

2.3.4 Datová redukce

Datová redukce v případě našeho problému není vzhledem k velikosti trénovacího datasetu potřebná.

2.4 Analýza klasifikačních metod

Ke klasifikaci zákazníků porovnáme tři klasifikátory – rozhodovací stromy, logistickou regresi a umělou neuronovou síť. Všechny vybrané klasifikátory jsou již implementovány v mnoha *frameworcích*, které velmi zjednodušují jejich použití. Pro tuto klasifikaci použijeme RapidMiner Studio[3] (verze 8.1) a konkrétně operátory *Deep Learning*, *Gradient Boosted Trees* a *Generalized Linear Model* (vše od *H2O* ve verzi 3.8.2.6).

Pro ohodnocení jednotlivých klasifikátorů použijeme křížovou validaci. Může se stát, že se zákazník choval jinak v roce 2015 než v roce 2017. Proto v křížové validaci nastavíme *batch* parametr (připravený atribut *time_created_fold*), který bude zjednodušeně simulovat chování zákazníků v čase – nestane se nám tak, že budeme porovnávat zákazníky z roku 2015 se zákazníky z roku 2017.

Při klasifikaci budeme zkoumat následující parametry:

- *accuracy*,
- plocha pod *ROC* křivkou,
- *precision*,
- *recall*,
- plocha pod *PR* křivkou,
- *F-Measure*,
- trénovací čas,
- spotřeba RAM,
- standardní odchylka.

Trénovacím časem se rozumí čas, který je potřebný k naučení klasifikátoru. Spotřeba RAM je také měřena na trénování klasifikátoru.

Vzhledem k tomu, že třídy, do kterých zákazníky klasifikujeme, jsou poměrně nevyvážené (viz. sekce 2.2), má větší přínos *F-Measure* a plocha pod *PR* křivkou než *accuracy* či plocha pod *ROC* křivkou. V případě, že by náš naučený model klasifikoval všechny zákazníky jako zákazníky, kteří smlouvu neuzavřou, tak by totiž dosahoval *accuracy* kolem 90 %. Takový model by nám ale mnoho informací nepřinesl.

[3] `https://rapidminer.com/products/studio/`

2.4.1 Nastavení parametrů klasifikátorů

Tato sekce popisuje nastavení parametrů pro klasifikátory v *RapidMineru*.

Celkové nastavení parametrů je k dispozici v exportu jednotlivých klasifikátoru v přiloženém CD. Optimalizace těchto vybraných parametrů byla provedena evolučním algoritmem, v *RapidMineru* operátorem *Optimize Parameters (Evolutionary)*, konkrétně turnajovou selekcí.

Gradient Boosted Trees:

- počet rozhodovacích stromů - 19
- maximální hloubka stromu - 5
- počet košů histogramu - 29

Generalized Linear Model:

- alpha - 0,265
- lambda - 2,558
- počet lambd - 8
- maximální počet iterací - 28

2.4.2 Porovnání klasifikačních metod

Pro porovnání klasifikátorů nás může zajímat, zda je výhodnější trénovat klasifikátor na zákaznících jako celku, nebo zda je lepší zákazníky rozdělit podle typu objednávky na zákazníky, kteří se zajímají o plyn a zákazníky, kteří se zajímají o elektrickou energii.

Klasifikátory, které budeme trénovat na všech zákaznících, naučíme na trénovacím datasetu, který je detailněji popsán v tabulce B.1. V případě, že zákazníky rozdělíme, použijeme trénovací dataset B.2 pro plyn, resp. dataset B.3 pro elektrickou energii.

Klasifikátory naučené na trénovacím datasetu objednávek plynu a elektrické energie jsou porovnány v tabulce 2.1, resp. v tabulce 2.2.

Srovnání modelů, které klasifikují zákazníky dohromady je zobrazené v tabulce 2.3.

Průměr klasifikátorů naučených jednotlivě podle typu objednávky potom zachycuje tabulka 2.4. Vzhledem k tomu, že klasifikátory byly naučené na rozdílných datasetech a tedy i na jiných počtech instancích, je tento průměr vážený právě podle počtu instancí, které trénovací dataset obsahoval. V případě umělé neuronové sítě bylo v porovnání s ostatními klasifikátory použito méně instancí z důvodu neúměrné časové náročnosti. Počty instancí a matice záměn všech modelů jsou k dispozici v příloze v sekci C.

	GBDT	GLM	ANN
accuracy	83,75 %	88,53 %	83,68 %
AUC	0,719	0,711	0.733
precision	30,63 %	63,35 %	31,78 %
recall	29,60 %	4,89 %	31,22 %
AUPRC	0,196	0,199	0,205
F-Measure	29,40 %	9,05 %	31,20 %
trénovací čas	47 s	325 s	12 980 s
spotřeba RAM	1300 Mb	1400 Mb	1100 Mb
standardní odchylka	0,024	0,014	0,028

Tabulka 2.1: Porovnání modelů klasifikujících zákazníky zajímající se o plyn

	GBDT	GLM	ANN
accuracy	89,01 %	88,57 %	85,67 %
AUC	0,840	0,712	0,804
precision	59,75 %	60,38 %	41,28 %
recall	39,68 %	4,81 %	39,24 %
AUPRC	0,255	0,197	0,232
F-Measure	47,41 %	8,86 %	39,33 %
trénovací čas	47 s	325 s	12 980 s
spotřeba RAM	1300 Mb	1400 Mb	1100 Mb
standardní odchylka	0,021	0,013	0,022

Tabulka 2.2: Porovnání modelů klasifikujících zákazníky zajímající se o elektrickou energii

	GBDT	GLM	ANN
accuracy	77,81 %	88,75 %	81,62 %
AUC	0,724	0,675	0,679
precision	25,06 %	55,72 %	24,59 %
recall	41,52 %	00,97 %	28,75 %
AUPRC	0,186	0,177	0,169
F-Measure	29,78 %	01,99 %	25,09 %
trénovací čas	47 s	325 s	12 980 s
spotřeba RAM	1300 Mb	1400 Mb	1100 Mb
standardní odchylka	0,088	0,018	0,053

Tabulka 2.3: Porovnání modelů klasifikujících zákazníky dohromady

Z tabulek je patrné, že je výhodnější zákazníky rozdělit podle druhu zvolené komodity. Z hlediska výsledků vychází nejlépe rozhodovací stromy, které umělou neuronovou síť předčily, jak v úspěšnosti, tak i v časové náročnosti,

	GBDT	GLM	ANN
accuracy	82,59 %	88,66 %	83,65 %
AUC	0,774	0,694	0,742
precision	39,87 %	58,16 %	32,94 %
recall	40,73 %	2,98 %	34,00 %
AUPRC	0,215	0,187	0,201
F-Measure	37,30 %	5,58 %	32,21 %
trénovací čas	47 s	325 s	12 980 s
spotřeba RAM	1300 Mb	1400 Mb	1100 Mb
standardní odchylka	0,059	0,015	0,038

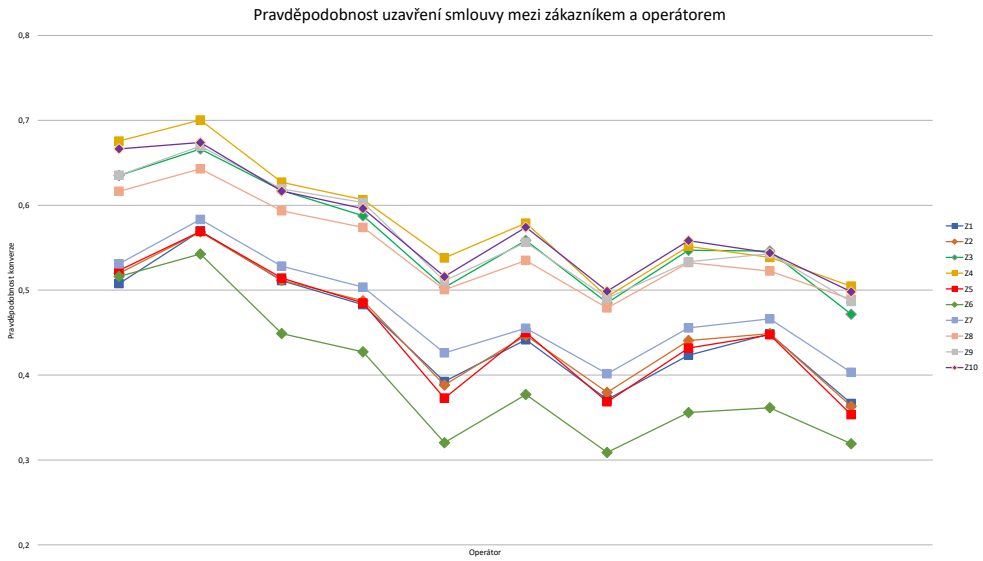
Tabulka 2.4: Porovnání modelů – vážený průměr modelu 2.1 a 2.2 podle počtu instancí

která je v porovnání s neuronovou sítí v případě rozhodovacích stromů o poznání nižší. Naopak u logistické regrese je vidět, že má klasifikátor poměrně vysokou přesnost, ale velmi nízkou *F-Measure*. To je způsobené především tím, že tento operátor ohodnotil velmi velkou část zákazníků negativně (viz příloha C).

2.5 Analýza optimalizačních metod

Z hlediska optimalizace nás především zajímá, jak moc nám tato optimalizace pomůže k zvětšení zisku společnosti. Graf 2.5 zobrazuje pravděpodobnost uzavření smlouvy mezi náhodným výběrem deseti operátorů a deseti zákazníků. Tato pravděpodobnost je napočítána klasifikátorem rozhodovacích stromů (viz sekce 2.4). Z grafu je patrné, že optimalizace pomocí celočíselného lineárního programování v tomto případně v porovnání s hladovou metodou mnoho nepřinese, jelikož většina „dobrých“ zákazníků má vysokou pravděpodobnost uzavření smlouvy s každým operátorem. Naopak „horší“ zákazníci mají nižší pravděpodobnost uzavření smlouvy u všech operátorů.

Vzhledem ke skutečnosti, že můžeme proces optimalizace přiřazení napočítat dopředu v době, kdy systém není zatížený a zároveň budeme optimalizovat přiřazení operátorů a zákazníků na jeden den, případně nějakou krátkou dobu, nemusíme se příliš starat o paměťovou a časovou náročnost. Vhodným kandidátem je tedy celočíselné lineární programování, které nám zaručuje nalezení optimálního přiřazení a zároveň oproti použití maďarské metody ušetříme na paměťové složitosti, protože si budeme moci odpustit použití tzv. *dummy* operátorů. Pro implementaci celočíselného lineárního programování zároveň existuje celá řada *frameworků*, implementační složitost je tedy nízká.



Obrázek 2.5: Graf zobrazující pravděpodobnosti uzavření smlouvy mezi náhodným výběrem deseti operátorů a deseti zákazníků (*graf je spojitý pro zvýšení přehlednosti*

Kapitola 3

Implementační část

Tato kapitola popisuje demonstraci použití rozhodovacích stromů jako klasifikátoru a celočíselného lineárního programování na příkladu jednoduché simulace pracovního dne call-centra.

V sekci 3.1 je nejdříve popsáno schéma zapojení klasifikátoru rozhodovacích stromů v programu *RapidMiner Studio*. Sekce 3.2 se potom věnuje implementaci celočíselného lineárního programování v programovacím jazyce *Java*. Závěr této kapitoly tvoří analýza výsledků provedené simulace.

Vzhledem ke skutečnosti, že v sekci 2.4.2 vyšlo lépe zákazníky rozdělit, je simulace v této bakalářské práci demonstrována pouze na zákaznících, kteří mají zájem o plyn. U zákazníků zajímajících se o elektrickou energii by byl postup identický, akorát s použitím jiného datasetu.

3.1 Implementace klasifikační metody

Pro implementaci klasifikátoru použijeme RapidMiner Studio[4] ve verzi 8.1.000.

V *RapidMineru* se připojíme k databázi, ve které máme připravený trénovací dataset (viz příloha B.2), pomocí operátoru *Read Database*. V tomto operátoru definujeme *SQL* dotaz, kterým získáme data z tohoto datasetu.

Získána data musíme rozdělit na trénovací část a část, kterou použijeme pro simulaci pracovního dne. K tomu můžeme použít operátor *Split Data*, který nám umožňuje načtená data disjunktně rozdělit. Druhým způsobem je použít dvakrát operátor *Read Database*. Jednou v definovaném dotazu vybereme instance, které mají datum vytvoření před zvoleným datem simulace (tato data budou sloužit jako historická data k trénování klasifikátoru) a ve druhém operátoru *Read Database* definujeme dotaz, který nám vybere instance vytvořené ve zvolený pracovní den (instance sloužící k simulaci).

Dalším krokem je předzpracování dat a s tím související nastavení *labelu*. Tento *label* nám rozdělí zákazníky na dvě třídy (jedná se o binární klasi-

[4] `https://rapidminer.com/products/studio/`

fikaci) – zákazníky, kteří se společností smlouvu uzavřeli (1) a naopak na zákazníky, kteří se společností smlouvu neuzavřeli (0). K nastavení *labelu* použijeme operátor *Set Role*, ve kterém vybereme atribut „*label*“ z datasetu. Zároveň potřebujeme vytvořit pro simulaci kartézský součin operátorů se zákazníky, které chceme ohodnotit. Získáme tak ohodnocení pro každou dvojici operátor-zákazník aktivní ve zvolený den. K tomu nám stačí v operátoru *Read Database*, určenému pro získání dat určených k simulaci, upravit SQL dotaz s využitím kartézského součinu.

Další operátor, který použijeme, je samotný klasifikátor – z analýzy v kapitole 2.4.2 nejlepších výsledků dosáhl klasifikátor rozhodovacích stromů. Zvolíme tedy klasifikátor *Gradient Boosted Trees* z *frameworku H2O* (použitá verze 3.8.2.6), kterému nastavíme jednotlivé parametry. Do tohoto klasifikátoru přivedeme data určená pro trénování z kroku výše.

Na nový operátor *Apply Model* právě tento naučený model aplikujeme a jako vstupní data tomuto operátoru přivedeme data, která jsou určená pro simulaci.

Výsledek klasifikace uložíme do nové relační tabulky v databázi pomocí operátoru *Write Database*. Nad touto tabulkou se bude provádět následná optimalizace přiřazení.

Celé výše popsané schéma zapojení klasifikátoru, které bylo exportováno z *RapidMineru*, je k dispozici na CD, které je součástí této práce.

3.2 Implementace optimalizační metody

Simulaci pracovní dne call-centra je demonstrována na jednoduché aplikaci, která je napsána v programovacím jazyce Java 8[5]. V aplikaci je využito několik frameworků, které jsou uvedeny spolu s jejich stručným využitím níže:

- Maven[6] (verze 3.0.5) – zjednodušení sestavení aplikace,
- Gurobi Optimizer[7] (verze 7.5.2) – implementace celočíselného lineárního programování,
- Hibernate ORM[8] (verze 5.0.2) – objektově-relační mapování,
- JUnit[9] (verze 4.12) – testování,
- JFreeChart[10] (verze 1.0.13) – grafické znázornění výsledků.

[5] `https://java.com/en/download/`
[6] `https://maven.apache.org/`
[7] `http://www.gurobi.com/`
[8] `http://hibernate.org/orm/`
[9] `https://junit.org/junit4/`
[10] `http://www.jfree.org/jfreechart/`

Aplikace se pomocí *Hibernate ORM* a *PostgreSQL JDBC Driveru* (verze 42.1.4) připojí do relační databáze. Z tabulky, kterou jsme získali v sekci 3.1 (ve které jsou již klasifikované dvojice – operátor-zákazník), vytvoříme tři relační tabulky – *model_operator*, *model_lead* a *model_lead_operator_pair*. Tyto tabulky reprezentují zvolené operátory, zákazníky, resp. dvojice, které budeme optimalizovat.

Z databáze se při simulaci náhodně vybere definovaný počet operátorů, kteří reprezentují operátory aktivní v simulovaný den. K těmto operátorům se následně optimalizuje přiřazení zákazníků, které se operátoři mají pokusit oslovit.

Optimalizace probíhá takovým způsobem, aby se dosáhlo co nejvyššího zisku – tj. *provize za zákazníka * jeho pravděpodobnost uzavření smlouvy*. Implementovány jsou následující metody optimalizace přiřazení zákazníka k operátorovi:

1. ILP metoda – optimalizace za použití celočíselného lineárního programování;
2. greedy metoda – zvolenému operátorovi se přiřadí zákazník, který má nejvyšší koeficient *provize za zákazníka * jeho pravděpodobnost uzavření smlouvy se společností*;
3. náhodné oslovení – zvolenému operátorovi se přiřadí zákazník náhodně.

Aplikace obsahuje třídy pro testování optimalizace hladového přiřazení i celočíselného lineárního programování na příkladu, který je díky své velikosti možné ověřit manuálně.

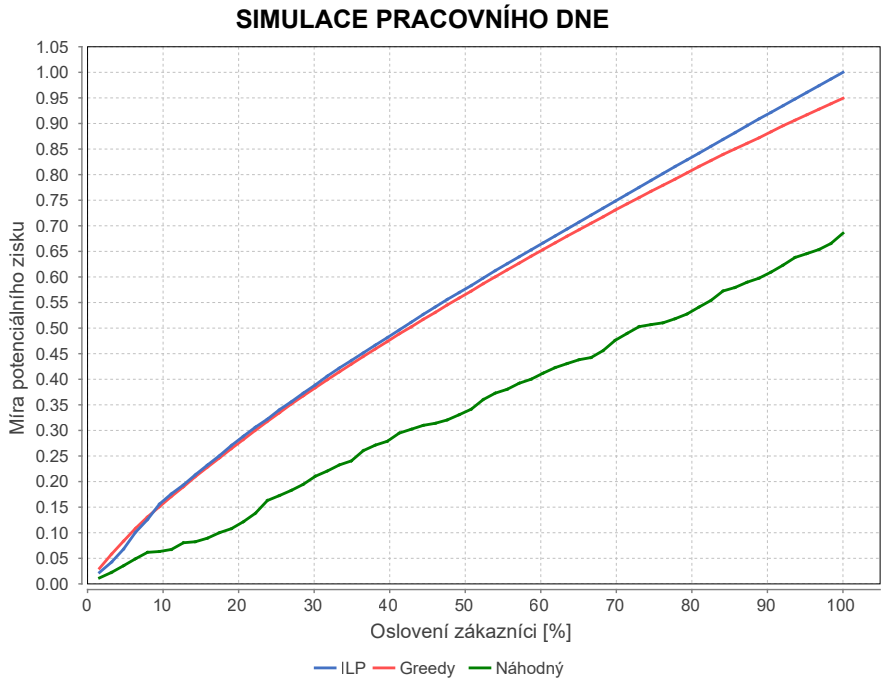
Grafické znázornění výsledků jednotlivých metod a jejich analýza je detailněji popsána v následující sekci 3.3.

3.3 Analýza výsledků

Graf 3.1 zobrazuje výsledky optimalizačních metod z předešlé simulace. Na ose x je vyjádřeno procentuální obvolání zákazníků v průběhu dne, zatímco osa y vyjadřuje míru potenciálního zisku – *provize za osloveného klienta * jeho pravděpodobnost uzavření smlouvy se společností*. Tato míra je normalizována podle metody, která dosáhla nejlepších výsledků (optimalizace za použití celočíselného lineárního programování).

Z grafu se potvrdila hypotéza, že optimalizace pomocí celočíselného lineárního programování nepřinese v porovnání s hladovým algoritmem velké zlepšení. Interakce mezi různými operátory a zvoleným zákazníkem zobrazené v grafu 2.5 se při optimalizaci začínají projevovat až na konci, kdy jsou „nejlepší“ zákazníci již osloveni.

Na druhou stranu je z grafu patrné, že v porovnání s náhodným oslovením zákazníků, které je využívané ve společnosti aktuálně, dosahují obě metody



Obrázek 3.1: Graf zobrazující míru potenciálního zisku jednotlivých optimalizačních metod

potenciálně až o třetinu větší zisk. Z tohoto důvodu je vhodné optimalizaci do společnosti implementovat. Vzhledem ke skutečnosti, že přiřazení operátora ke klientovi budeme optimalizovat na poměrně krátkou dobu (maximálně několik dní dopředu), je možné do společnosti implementovat celočíselné lineární programování jak z hlediska časové, tak i paměťové náročnosti. To nám zajistí ve srovnání s hladovým algoritmem alespoň částečné navýšení zisku společnosti.

Závěr

Tato bakalářská práce se zabývala teoretickou rešerší, analýzou a následnou simulací problému oslovení zákazníků společnosti, která provozuje call-centrum. Odpovídá na dvě klíčové otázky, které byly definované v úvodu této práce, a to konkrétně:

1. Kteří zákazníci call-centra mají být osloveni?
2. Který operátor má oslovit vybraného zákazníka?

V analytické části byly porovnány tři klasifikační metody – *Gradient Boosted Decision Trees*, *umělá neuronová síť*, *logistická regrese*, a tři optimalizační metody – *Maďarský algoritmus*, *celočíselné lineární programování* a *hladový algoritmus*.

Nejlepších výsledků v klasifikačních metodách dosáhly rozhodovací stromy, podle kterých dokážeme s určitou pravděpodobností určit, jestli vybraný zákazník se společností smlouvu uzavře, či nikoliv.

V optimalizaci přiřazení operátora ke konkrétnímu zákazníkovi dosáhlo s ohledem na zisk společnosti nejlepších výsledků celočíselné lineární programování. Bohužel se nepotvrdila původní hypotéza, že vybraný operátor dokáže oslovit zákazníka lépe než jiný. I přesto, v porovnání se současným řešením společnosti, které je založené na náhodném přiřazení operátora k zákazníkovi, vychází navýšení potenciálního zisku společnosti až o třetinu. Nevýhodou celočíselného lineárního programování je jeho časová a paměťová náročnost. Vzhledem k tomu, že celá optimalizace přiřazení probíha s časovým předstihem, je možné tuto nevýhodu eliminovat.

Použitelnost vybraných metod je demonstrována na jednoduché simulaci pracovního dne call-centra. V této aplikaci jsou graficky porovnány jednotlivé optimalizační metody přiřazení. Tyto optimalizační metody využívají rozhodovacích stromů, podle kterých jsme ohodnotily zákazníky podle pravděpodobnosti uzavření smlouvy se společností.

Vytvořená studie nezahrnuje implementaci kompletní aplikace, která by byla implementována v call-centru a automaticky by přiřazovala operátorovi vybrané zákazníky. Proto by bylo možné práci rozšířit o implementaci aplikace, která by toto umožňovala.

Bibliografie

1. CHANDOLA, VARUN; BANERJEE, ARINDAM; KUMAR, VIPIN. Outlier Detection: A Survey. Dostupné také z: https://web.cs.hacettepe.edu.tr/~aykut/classes/spring2013/bil682/supplemental/Outlier_Detection_A_Survey.pdf.
2. GRZYMALA-BUSSE, Jerzy W; HU, Ming. A comparison of several approaches to missing attribute values in data mining. In: *International Conference on Rough Sets and Current Trends in Computing*. 2000, s. 378–385. Dostupné také z: https://doi.org/10.1007/3-540-45554-X_46.
3. KANTARDZIC, Mehmed. *Data Reduction*. Wiley Online Library, 2003.
4. WOLPERT, David H; MACREADY, William G. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*. 1997, roč. 1, č. 1, s. 67–82. Dostupné také z: 10.1109/4235.585893.
5. HAN, Jiawei; PEI, Jian; KAMBER, Micheline. *Data mining: concepts and techniques*. Elsevier, 2011.
6. MCCULLOCH, Warren S; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*. 1943, roč. 5, č. 4, s. 115–133. Dostupné také z: https://doi.org/10.1007/BF02478259.
7. MAAS, Andrew L; HANNUN, Awni Y; NG, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In: *Proc. icml*. 2013, sv. 30, s. 3. Č. 1.
8. MUNKRES, James. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*. 1957, roč. 5, č. 1, s. 32–38. Dostupné také z: https://doi.org/10.1137/0105003.

9. YEDIDSION, Liron; SHABTAY, Dvir; KASPI, Moshe. Complexity analysis of an assignment problem with controllable assignment costs and its applications in scheduling. *Discrete Applied Mathematics*. 2011, roč. 159, č. 12, s. 1264–1278. Dostupné také z: https://doi.org/10.1016/j.dam.2011.04.001.

10. LAWLER, Eugene L. *Combinatorial optimization: networks and matroids*. Courier Corporation, 1976.

11. KARP, Richard M. Reducibility among combinatorial problems. In: *Complexity of computer computations*. Springer, 1972, s. 85–103. Dostupné také z: https://doi.org/10.1007/978-1-4684-2001-2_9.

12. MARCHAND, Hugues; MARTIN, Alexander; WEISMANTEL, Robert; WOLSEY, Laurence. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*. 2002, roč. 123, č. 1-3, s. 397–446. Dostupné také z: 10.1016/S0166-218X(01)00348-1.

13. LAND, Ailsa H; DOIG, Alison G. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*. 1960, s. 497–520. Dostupné také z: https://doi.org/10.1007/978-3-540-68279-0_5.

14. KUHN, Harold W. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*. 1955, roč. 2, č. 1-2, s. 83–97. Dostupné také z: https://doi.org/10.1002/nav.3800020109.

15. LEWELLEN. *A Greedy Approximation Algorithm for the Linear Assignment Problem*. Dostupné také z: https://antimatroid.wordpress.com/2017/03/21/a-greedy-approximation-algorithm-for-the-linear-assignment-problem/ 2017-03-24.

Příloha A

Seznam použitých zkratek

ANN Artificial Neural Network – umělá neuronová síť

GBDT Gradient Boosted Decision Trees – klasifikační model rozhodovacích stromů tvoří pomocí metody *boosting*

GLM Generalized Linear Model – obecný lineární model

ILP Integer Linear Programming – celočíselné lineární programování

Příloha B

Popis trénovacího datasetu

Tabulka B.1: Popis trénovacího datasetu – *zákaznící jsou klasifikování dohromady jak na plyn, tak i na elektrickou energii*

Název příznaku	**Popis příznaku**
time_created	Datum vytvoření objednávky
category_id	Informace zda se jedná o objednávku na plyn nebo elektřinu
client_id	Identifikační číslo zákazníka
country_region_id	Identifikační číslo kraje, ke kterému se váže objednávka
house_type_id	Identifikační číslo typu domu (rodinný dům / byt)
previously_unanswered_count	Počet dříve nepřijatých hovorů zákazníkovi ze strany call-centra
user_agent	Informace o internetovém prohlížeči zákazníka, ze kterého byla objednávka vytvořena
device	Informace o zařízení (stolní PC / mobil / tablet) zákazníka, ze kterého byla objednávka vytvořena
origin_calculator_name	Identifikační klíč webového rozhraní, ze kterého zákazník objednávku vyplnil
	Pokračování na další straně

Tabulka B.1 – pokračování z předešlé strany

Název příznaku	**Popis příznaku**
utm_source	Název zdroje, ze kterého zákazník přišel vyplnit objednávku (UTM parametr)
utm_campaign	Název konkrétní kampaně, která zákazníka přivedla k vyplnění objednávky (UTM parametr)
utm_medium	Informace, zda zákazník přišel vyplnit objednávku přes reklamu nebo email (UTM parametr)
gas_accurate *	Informace zda zákazník zná svou přesnou spotřebu plynu
gas_consumption *	Zákazníkova roční spotřeba plynu v MWh
gas_usage_purpose_1 *	Informace zda zákazník využívá plyn k ohřevu vody
gas_usage_purpose_2 *	Informace zda zákazník využívá plyn k vaření
gas_usage_purpose_3 *	Informace zda zákazník využívá plyn k vytápění
gas_actual_product_id *	Identifikační klíč současného zákazníkova dodavatele plynu
gas_saving	Odhadovaná částka, kterou zákazník může za plyn ušetřit
electricity_accurate *	Informace zda zákazník zná svou přesnou spotřebu elektrické energie
electricity_consumer_type *	Informace zda se jedná o objednávku pro domácnost / podnikatele
electricity_consumption *	Zákazníkova současná roční spotřeba elektrické energie v kWh
electricity_usage_purpose *	Hlavní využití elektrické energie zákazníka (elektrický bojler / akumulační vytápění / přímotopy / tepelné čerpadlo)
electricity_rate_id *	Identifikační klíč současné sazby zákazníka za elektrickou energii

Pokračování na další straně

Tabulka B.1 – pokračování z předešlé strany

Název příznaku	Popis příznaku
electricity_circuit_id *	Identifikační klíč jističe zákazníka
electricity_contract_period *	Informace zda má současně zákazník uzavřenou smlouvu s dodavatelem elektrické energie na dobu určitou
electricity_actual_product *	Identifikační klíč současného zákazníkova dodavatele elektrické energie
electricity_saving	Odhadovaná částka, kterou zákazník může za elektrickou energii ušetřit
operator_id	Identifikační číslo operátora call-centra
operator_department_id	Identifikační číslo oddělení operátora call-centra
operator_experience_seconds	Celková doba v call-centru operátora převedená na sekundy (kalkulovaný příznak)
time_created_fold **	Rozdělení objednávek dle data vytvoření do dvanácti skupin (kalkulovaný příznak)
commission **	Provize, kterou společnost získá za uzavření smlouvy (kalkulovaný příznak)
gas_storno_count **	Počet předešlých zákazníkových objednávek na plyn, které byly stornovány (kalkulovaný příznak)
gas_successful_count **	Počet předešlých zákazníkových objednávek na plyn, které skončily uzavřením smlouvy (kalkulovaný příznak)
gas_last_order **	Informace zda poslední zákazníkova objednávka plynu skončila uzavřením smlouvy (kalkulovaný příznak)
electricity_storno_count **	Počet předešlých zákazníkových objednávek na elektrickou energii, které byly stornovány (kalkulovaný příznak)
Pokračování na další straně	

Tabulka B.1 – pokračování z předešlé strany

Název příznaku	**Popis příznaku**
electricity_successful_count **	Počet předešlých zákazníkových objednávek na elektrickou energii, které skončily uzavřením smlouvy (kalkulovaný příznak)
electricity_last_order **	Informace zda poslední zákazníkova objednávka elektrické energie skončila uzavřením smlouvy (kalkulovaný příznak)

Tabulka B.1:
* *data získána přímo od uživatelů přes webové rozhraní*
** *kalkulované atributy*

Tabulka B.2: Popis trénovacího datasetu – *klasifikováni jsou pouze zákaznící zajímající se o plyn*

Název příznaku	**Popis příznaku**
time_created	Datum vytvoření objednávky
category_id	Informace zda se jedná o objednávku na plyn nebo elektřinu
client_id	Identifikační číslo zákazníka
country_region_id	Identifikační číslo kraje, ke kterému se váže objednávka
house_type_id	Identifikační číslo typu domu (rodinný dům / byt)
previously_unanswered_count	Počet dříve nepřijatých hovorů zákazníkovi ze strany call-centra
user_agent	Informace o internetovém prohlížeči zákazníka, ze kterého byla objednávka vytvořena
device	Informace o zařízení (stolní PC / mobil / tablet) zákazníka, ze kterého byla objednávka vytvořena
Pokračování na další straně	

Tabulka B.2 – pokračování z předešlé strany

Název příznaku	Popis příznaku
origin_calculator_name	Identifikační klíč webového rozhraní, ze kterého zákazník objednávku vyplnil
utm_source	Název zdroje, ze kterého zákazník přišel vyplnit objednávku (UTM parametr)
utm_campaign	Název konkrétní kampaně, která zákazníka přivedla k vyplnění objednávky (UTM parametr)
utm_medium	Informace, zda zákazník přišel vyplnit objednávku přes reklamu nebo email (UTM parametr)
gas_accurate *	Informace zda zákazník zná svou přesnou spotřebu plynu
gas_consumption *	Zákazníkova roční spotřeba plynu v MWh
gas_usage_purpose_1 *	Informace zda zákazník využívá plyn k ohřevu vody
gas_usage_purpose_2 *	Informace zda zákazník využívá plyn k vaření
gas_usage_purpose_3 *	Informace zda zákazník využívá plyn k vytápění
gas_actual_product_id *	Identifikační klíč současného zákazníkova dodavatele plynu
gas_saving	Odhadovaná částka, kterou zákazník může za plyn ušetřit
operator_id	Identifikační číslo operátora call-centra
operator_department_id	Identifikační číslo oddělení operátora call-centra
operator_experience_seconds	Celková doba v call-centru operátora převedená na sekundy (kalkulovaný příznak)
time_created_fold **	Rozdělení objednávek dle data vytvoření do dvanácti skupin (kalkulovaný příznak)

Pokračování na další straně

Tabulka B.2 – pokračování z předešlé strany

Název příznaku	**Popis příznaku**
commission **	Provize, kterou společnost získá za uzavření smlouvy (kalkulovaný příznak)
gas_storno_count **	Počet předešlých zákazníkových objednávek na plyn, které byly stornovány (kalkulovaný příznak)
gas_successful_count **	Počet předešlých zákazníkových objednávek na plyn, které skončily uzavřením smlouvy (kalkulovaný příznak)
gas_last_order **	Informace zda poslední zákazníkova objednávka plynu skončila uzavřením smlouvy (kalkulovaný příznak)

Tabulka B.2:
* *data získána přímo od uživatelů přes webové rozhraní*
** *kalkulované atributy*

Tabulka B.3: Popis trénovacího datasetu – *klasifikováni jsou pouze zákaznící zajímající se o elektrickou energii*

Název příznaku	**Popis příznaku**
time_created	Datum vytvoření objednávky
category_id	Informace zda se jedná o objednávku na plyn nebo elektřinu
client_id	Identifikační číslo zákazníka
country_region_id	Identifikační číslo kraje, ke kterému se váže objednávka
house_type_id	Identifikační číslo typu domu (rodinný dům / byt)
previously_unanswered_count	Počet dříve nepřijatých hovorů zákazníkovi ze strany call-centra
Pokračování na další straně	

Tabulka B.3 – pokračování z předešlé strany

Název příznaku	Popis příznaku
user_agent	Informace o internetovém prohlížeči zákazníka, ze kterého byla objednávka vytvořena
device	Informace o zařízení (stolní PC / mobil / tablet) zákazníka, ze kterého byla objednávka vytvořena
origin_calculator_name	Identifikační klíč webového rozhraní, ze kterého zákazník objednávku vyplnil
utm_source	Název zdroje, ze kterého zákazník přišel vyplnit objednávku (UTM parametr)
utm_campaign	Název konkrétní kampaně, která zákazníka přivedla k vyplnění objednávky (UTM parametr)
utm_medium	Informace, zda zákazník přišel vyplnit objednávku přes reklamu nebo email (UTM parametr)
electricity_accurate *	Informace zda zákazník zná svou přesnou spotřebu elektrické energie
electricity_consumer_type *	Informace zda se jedná o objednávku pro domácnost / podnikatele
electricity_consumption *	Zákazníkova současná roční spotřeba elektrické energie v kWh
electricity_usage_purpose *	Hlavní využití elektrické energie zákazníka (elektrický bojler / akumulační vytápění / přímotopy / tepelné čerpadlo)
electricity_rate_id *	Identifikační klíč současné sazby zákazníka za elektrickou energii
electricity_circuit_id *	Identifikační klíč jističe zákazníka
electricity_contract_period *	Informace zda má současně zákazník uzavřenou smlouvu s dodavatelem elektrické energie na dobu určitou

Pokračování na další straně

Tabulka B.3 – pokračování z předešlé strany

Název příznaku	**Popis příznaku**
electricity_actual_product *	Identifikační klíč současného zákazníkova dodavatele elektrické energie
electricity_saving	Odhadovaná částka, kterou zákazník může za elektrickou energii ušetřit
operator_id	Identifikační číslo operátora call-centra
operator_department_id	Identifikační číslo oddělení operátora call-centra
operator_experience_seconds	Celková doba v call-centru operátora převedená na sekundy (kalkulovaný příznak)
time_created_fold **	Rozdělení objednávek dle data vytvoření do dvanácti skupin (kalkulovaný příznak)
commission **	Provize, kterou společnost získá za uzavření smlouvy (kalkulovaný příznak)
electricity_storno_count **	Počet předešlých zákazníkových objednávek na elektrickou energii, které byly stornovány (kalkulovaný příznak)
electricity_successful_count **	Počet předešlých zákazníkových objednávek na elektrickou energii, které skončily uzavřením smlouvy (kalkulovaný příznak)
electricity_last_order **	Informace zda poslední zákazníkova objednávka elektrické energie skončila uzavřením smlouvy (kalkulovaný příznak)

Tabulka B.3:
* *data získána přímo od uživatelů přes webové rozhraní*
** *kalkulované atributy*

Příloha C

Matice záměn klasifikátorů

		skutečnost	
		ANO	NE
klasifikace	ANO	4 114	13 642
	NE	5 588	62 699

Tabulka C.1: Matice záměn rozhodovacích stromů v případě modelu naučeného na zákaznících zajímajících se jak o plyn, tak i elektrickou energii

		skutečnost	
		ANO	NE
klasifikace	ANO	99	57
	NE	9 697	76 190

Tabulka C.2: Matice záměn logistické regrese v případě modelu naučeného na zákaznících zajímajících se jak o plyn, tak i elektrickou energii

		skutečnost	
		ANO	NE
klasifikace	ANO	385	1 266
	NE	940	9 409

Tabulka C.3: Matice záměn umělé neuronové sítě v případě modelu naučeného na zákaznících zajímajících se jak o plyn, tak i elektrickou energii

		skutečnost	
		ANO	NE
klasifikace	ANO	3 208	7 590
	NE	7 688	75 697

Tabulka C.4: Matice záměn rozhodovacích stromů v případě modelu naučeného na zákaznících zajímajících se o plyn

		skutečnost	
		ANO	NE
klasifikace	ANO	542	318
	NE	10 488	82 835

Tabulka C.5: Matice záměn logistické regrese v případě modelu naučeného na zákaznících zajímajících se o plyn

		skutečnost	
		ANO	NE
klasifikace	ANO	457	986
	NE	972	9 585

Tabulka C.6: Matice záměn umělé neuronové sítě v případě modelu naučeného na zákaznících zajímajících se o plyn

		skutečnost	
		ANO	NE
klasifikace	ANO	3 137	2 016
	NE	4 685	54 232

Tabulka C.7: Matice záměn rozhodovacích stromů v případě modelu naučeného na zákaznících zajímajících se o elektrickou energii

		skutečnost	
		ANO	NE
klasifikace	ANO	527	347
	NE	10 415	82 894

Tabulka C.8: Matice záměn logistické regrese v případě modelu naučeného na zákaznících zajímajících se o elektrickou energii

		skutečnost	
		ANO	NE
klasifikace	ANO	573	825
	NE	894	9 708

Tabulka C.9: Matice záměn umělé neuronové sítě v případě modelu naučeného na zákaznících zajímajících se o elektrickou energii

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `thesis.pdf` ..... text bakalářské práce ve formátu PDF
- `classification` ..... adresář s vygenerovanými procesy
  - `model` ..... adresář s procesem pro klasifikaci
  - `validation` ..... adresář s procesy pro validaci
- `optimization` ..... adresář s implementací optimalizační metody
  - `logs` ..... adresář s grafem zobrazujícím výsledky
  - `pom.xml` ..... XML soubor obsahující nastavení projektu
  - `src` ..... zdrojové kódy implementace simulace
  - `sql` ..... create a insert skripty k optimalizaci