VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ ZNALOSTÍ Z DAT POJIŠŤOVNY

KNOWLEDGE DISCOVERY FROM DATA OF AN INSURANCE COMPANY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ONDŘEJ KŘÍŽ**
AUTHOR

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

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

154630

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Kříž Ondřej**
Program:	Informační technologie
Název:	***Získávání znalostí z dat pojišťovny***
Kategorie:	Data mining
Akademický rok:	2023/24

Zadání:

1. Seznamte se s problematikou získávání znalostí z databází. Dále se seznamte s daty poskytnutými pojišťovnou.
2. Navrněte několik vhodných úloh získávání znalostí, které by bylo možné provést nad poskytnutými daty. Úlohy konzultujte s vedoucím.
3. Zvolte vhodné implementační prostředí a knihovny vhodné pro implementaci navržených úloh.
4. Implementujte navržené úlohy a provďte experimenty a vyhodnocení výsledků vyřešených úloh.
5. Zhodnoťte dosažené výsledky a další možnosti pokračování tohoto projektu.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1.

Při obhajobě semestrální části projektu je požadováno:
Semestrální projekt již byl obhájen.

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

Vedoucí práce:	**Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2023
Termín pro odevzdání:	9.5.2024
Datum schválení:	30.10.2023

Abstrakt

Tato bakalářská práce se zabývá problematikou získávání znalostí z databází. Jejím cílem je z provozních dat nejmenované pojišťovny sestavit algoritmicky zpracovatelné datasety, které budou následně analyzovány funkcemi knihovny scikit-learn jazyka Python za použitím různých algoritmů z oblasti klasifikace a algoritmu FP-growth v oblasti tvorby silných asociačních pravidel a následné vyhodnocení výsledků.

Abstract

This bachelor thesis deals with the issue of knowledge discovery from databases. Its aim is to compile algorithmically processable datasets from operational data of an unnamed insurance company, which will subsequently be analyzed by functions of the scikit-learn library in the Python language using various classification algorithms and the FP-growth algorithm in the area of creating strong association rules and subsequent evaluation of results.

Klíčová slova

získávání znalostí z databází, dolování z dat, scikit-learn, programovací jazyk Python, klasifikace, rozhodovací strom, náhodný les, vícevrstvé perceptrony, frequent-pattern strom, frekventované množiny, silná asociační pravidla, pojišťovna

Keywords

knowledge discovery from databases, data mining, scikit-learn, Python programming language, classification, decision tree, random forest, multilayer perceptrons, frequent-pattern tree, frequent itemsets, strong association rules, insurance company

Citace

KŘÍŽ, Ondřej. *Získávání znalostí z dat pojišťovny.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Získávání znalostí z dat pojišťovny

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, PhD. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ondřej Kříž
29. dubna 2024

Poděkování

Rád bych poděkoval Ing. Vladimíru Bartíkovi, PhD., za vedení mojí práce, nápady, pochopení a podporu v průběhu vypracování práce. Chtěl bych také poděkovat Ing. Jindřichu Zehnalovi za konzultaci výsledků z praktického hlediska a taktéž za podporu, a do třetice poděkovat svojí manželce za realistický pohled a pravidelné dávky motivace.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Každým rokem se celosvětově zvyšuje množství dat, které je lidmi generováno a konzumováno. Do roku 2003 bylo celkově vygenerováno 5 exabytů dat, v roce 2013 už bylo stejné množství dat vygenerováno za dva dny [7]. Data vznikají každou lidskou činností, ať už nakoupením v obchodě nebo provozem chytré domácnosti. Zpracování a vizualizace dat je nutností, ale o krok dál, za bezprostředním zpracováním, leží krajina dolování znalostí z databází, kde je možné ze zdánlivě využitých a nepotřebných dat získat informace, které skutečně prohloubí naše porozumění problematice v doméně dat.

Již roku 1763 publikoval Thomas Bayes svůj "An Essay towards solving a Problem in the Doctrine of Chances", ve kterém navrhl algoritmus, ze kterého později vznikl Bayesův teorém, jeden ze základních stavebních kamenů dolování znalostí z databází. Velký rozvoj v oblasti proběhl v druhé polovině 20. století, kdy byly představeny další významné algoritmy, jako například neuronové sítě, rozhodovací stromy nebo náhodný les, které budou představeny později v této práci. I dnes prochází toto odvětví informatiky dynamickým rozvojem, kdy jsou běžně k vidění nové algoritmy i efektivnější modifikace již existujících algoritmů.

Cílem této práce je přehledová studie významných dolovacích algoritmů a jejich aplikace na datech pojišťovací společnosti, která k tomu poskytla data, a následné vyhodnocení výsledků, které tyto algoritmy přinesou.

Práce je rozdělená do čtyř kapitol. V první kapitole bude čtenář uveden do problematiky získávání znalostí z databází, budou mu představeny různé zdroje dat, s kterými se může při dolování znalostí z dat pojišťovací společnosti setkat, poté budou představeny efektivní metodologie využívané při dolování a také přehled nejdůležitějších dolovacích algoritmů.

Ve druhé kapitole bude čtenář seznámen s praktickými základy dolování znalostí, tedy prostředím a knihovnami, které mohou být k dolování použity. Ve třetí kapitole je popsáno řešení, které bylo v rámci této práce vypracováno. V poslední kapitole jsou představeny a rozebrány experimenty, které byly provedeny nad poskytnutými daty, a rovněž jejich zhodnocení. Na závěr jsou diskutována možná rozšíření této práce.

Kapitola 2

Získávání znalostí z databází

Tato kapitola slouží jako obecný úvod do problematiky získávání znalostí z databází s ohledem na pozdější implementační část této práce. Nejdříve budou představeny zdroje dat, které jsou pro dolování používány, následovat bude představení metodologie CRISP-DM. Další část bude věnována přípravě dat pro dolování. Poté budou představeny různé druhy dolovacích úloh, nejdříve klasifikace a predikce, a algoritmy, které k těmto úlohám používáme, následně dolování asociačních pravidel a jejich problematika.

2.1 Úvod do získávání znalostí z databází

Získávání znalostí z databází (překlad z anglického „Knowledge Discovery in Databases“, zkráceně KDD nebo také „data mining“ - dolování dat) označuje proces extrakce netriviálních, skrytých, dříve neznámých a potenciálně užitečných informací z velkých objemů dat.

Netriviálnost a skrytost těchto informací spočívá v složitosti jejich získání, neměly by tedy být získatelné prostým příkazem v dotazovacím jazyce. Stejně tak by výsledkem dolování neměla být informace patrná z prostého prohlédnutí databáze. Požadavek na předchozí neznalost vydolované informace zdůrazňuje především explorativní povahu KDD. Pro potvrzení dříve známých hypotéz a vztahů mezi daty je možné opět použít příkazů dotazovacího jazyka, a pracně tyto vztahy dolovat je zbytečné. Znalosti vydolované z databáze by také měly být relevantní pro příjemce těchto znalostí. Tento aspekt KDD zajišťuje, aby dolování přinášelo výsledky použitelné pro zlepšení rozhodovacího procesu nebo lepší předvídání na základě dat [4].

2.2 Zdroje dat pro dolování

Data použitá pro KDD mohou pocházet z různých zdrojů a mít různé podoby. Podle zdroje a podoby dat je také vhodné vybírat obecné dolovací úlohy a konkrétní dolovací algoritmy. Obecně mohou jako zdroj dat pro dolování sloužit jakákoliv smysluplná data [4].

Zdroje dat mohou být nestrukturované, semi-strukturované a strukturované [7]. Mezi nestrukturované se řadí textové dokumenty, audiovizuální soubory nebo příspěvky na sociální síti. Za semi-strukturovaná data považujeme data s danou strukturou, jejíž obsah ale může být příklad od příkladu značně odlišný. Do této kategorie se řadí soubory formátu XML a JSON, webová data formátu HTML, logovací soubory a obsah některých

typů NoSQL databází, například MongoDB. Pro jednodušší zpracování bývají data z výše zmíněných zdrojů transformovány do strukturované podoby.

Tradičnější jsou poté data, jejichž struktura je oproti výše uvedeným značně jednodušší. Data strukturovaná pocházejí z různých typů databází, například relačních, transakčních, operačních nebo z datových skladů. Zdroje dat relevantní pro tuto práci jsou následující:

- Relační databáze. Data v relačních databázích jsou organizována v záznamech, které jsou uloženy v tabulkách. Každý záznam je n-tice, která představuje jednu instanci objektu, který je popsán n atributy této n-tice. Mezi atributy záznamů jsou primární a cizí klíče, jejichž pomocí je realizována relace mezi tabulkami. Cizí klíč záznamu je odkazem na primární klíč jiného záznamu do některé z tabulek databáze. Obecně jsou relační databáze nejběžnější a nejpraktičtější zdroj dat, proto bývají často používány jako zdroj dat pro dolovací úlohy.

- Transakční databáze. V transakčních databázích jsou data také organizovány v tabulkách, kde každý záznam tabulky zachycuje jednu transakci. Záznam se poté skládá z unikátního identifikátoru transakce a seznamu položek, které transakci tvoří. Součástí transakční databáze poté mohou být další tabulky, které obsahují informace o nabízeném zboží. Příklad tabulky transakční databáze, která je v ideálním tvaru pro analýzu nákupního košíku, vidíme na obrázku 2.1

- Datové sklady. Účel datového skladu (anglicky data warehouse, zkráceně DW nebo také DWH) je jednotným způsobem ukládat informace, které jsou sesbírány z mnoha zdrojů. Důležitým aspektem datového skladu je historická perspektiva. Data v datovém skladu jsou totiž ukládána kontinuálně za delší časové období. Ukládání dat do DWH probíhá procesem ETL (anglicky Extract, Transform, Load, tedy extrakce, transformace a nahrání do datového skladu), jehož pomocí je zajištěna konsolidace dat a jejich použitelnost pro pozdější analýzu.

Transaction_ID	**Purchased_Items_IDs**
100	12, 15, 25
200	15, 20, 35, 40
350	12, 20
...	...

Obrázek 2.1: Příklad tabulky transakční databáze [4]

Pro interakci s databázemi slouží strukturovaný dotazovací jazyk (SQL - Structured Query Language). SQL je deklarativní jazyk, který umožňuje dotazování nad daty, jejich aktualizaci a manipulaci, například agregaci dat. Jak bylo zmíněno výše, dolováním znalostí by měly být informace, které není možné získat dotazem jazyka SQL. Přesto hraje v kontextu dolování znalostí významnou roli, protože jeho pomocí jsou často data připravována a transformována pro dolovací úlohy.

2.3 CRISP-DM

Model CRISP-DM (CRoss Industry Standard Process for Data Mining - volně přeložitelné jako mezioborový standardní proces pro dolování dat) [8] je obecně uznávaná metodologie

pro řízení dolovacích projektů. Jedná se o strukturovaný a flexibilní rámec, který může být přizpůsoben konkrétnímu dolovacímu procesu. Dolovací proces je v modelu CRISP-DM rozdělen do šesti fází, jak můžeme vidět na obrázku 2.2.

Obrázek 2.2: Model CRISP-DM [8]

Pochopení kontextu, nebo také pochopení business zadání, je první a z určitého pohledu nejdůležitější fází jakéhokoliv dolovacího projektu. V rámci tohoto kroku jsou určeny business cíle, zhodnocena současná situace a možnost realizace dolovacího projektu. Na základě toho můžou být stanoveny cíle, kterých může být v rámci dolování dosaženo a vypracován plán projektu.

Druhým krokem je pochopení dat. V této fázi jsou data prozkoumána, sesbírána a podrobena verifikaci, čímž je zjištěna jejich kvalita. Tento krok je s prvním krokem provázaný, protože nároky zadavatele nemusí odpovídat kvalitě a možnostem dat. Stejně tak je potřeba zkoumat jen část dat, která bude později použita ve výsledném dolovacím procesu.

Po zvládnutí druhého kroku se může dolovací projekt přesunout do třetí fáze - přípravy dat. Data jsou vyčištěna od šumu, zintegrována (z více zdrojů do jedné datové sady), jsou vytvořeny užitečné nebo odebrány nadbytečné atributy. Datová sada je vytvářena s ohledem na dolovací úlohy, které nad ní mají být prováděny. Proto je tento krok opět provázaný s krokem následujícím.

Tím je krok modelování. V tomto kroku je vybrána konkrétní modelovací technika (dolovací úloha), která bude nad datovou sadou spouštěna. Model je v tomto kroku také sestaven a jeho parametry jsou laděny, aby dosahovaly požadovaných kritérií, ať už je to celková úspěšnost, nebo další metriky, které budou rozebrány později.

Následuje krok zhodnocení, při kterém se hodnotí úspěch dolovacího projektu z hlediska původního business zadání. Je zhodnocen také celý proces a jsou stanoveny následující kroky. Těmi může být buď vrácení na začátek procesu a nová dohoda s business zadavateli, nebo nasazení projektu.

Při kroku nasazení je samotný projekt zpřístupněn uživateli. Dále jsou dohodnuty podmínky údržby projektu a monitorování jeho stavu. Je vypracována dokumentace a finální report.

Metodologie CRISP-DM si od svého uvedení v roce 1996 vydobyla pozici standardního procesu pro jakékoliv dolovací projekty, přičemž pomáhá jak zkušeným datovým analytikům, tak začátečníkům. Toto můžeme metodologii připsat díky její praktické orientaci a skutečnosti, že není proprietární [8].

2.4 Předzpracování dat

Příprava dat je klíčovou a potenciálně velice časově náročnou součástí každého dolovacího projektu. Protože na kvalitě dat přímo závisí úspěšnost dolovacího projektu, a protože data v databázích jsou k šumu a obecně nízké kvalitě velmi náchylná, příprava dat může zabrat až 60 % celkového úsilí věnovaného dolovacímu projektu. Tato sekce vychází z [4] a [10].

Mezi běžné problémy způsobující nízkou kvalitu dat patří neúplnost dat, projevující se v chybějících hodnotách, zašumění dat, které se projevuje v nesmyslných hodnotách. Dalšími problémy můžou být nekonzistence dat, kdy si různé části databáze navzájem odporují, nebo špatný rozsah dat a to jak příliš nízký, tak příliš velký, jak v počtu atributů, tak v počtu záznamů.

2.4.1 Čištění chybějících dat

Každý z problémů nekvalitních dat má jiné řešení. Obecným řešením chybějících hodnot nebo šumu v datech může být ignorování špatných záznamů, pokud se tímto nedostane počet záznamů na příliš nízkou úroveň.

S chybějícími hodnotami je možné se vypořádat doplněním záznamů. Toto doplnění může být ruční, pokud chybějících záznamů není mnoho. Při větším počtu chybějících dat je ale tento přístup neproveditelný.

Doplnění může být provedeno automaticky, na výběr jsou různé přístupy:

- Doplnění globální konstantou značící chybějící údaj
- Doplnění hodnotou vypočítanou průměrem, mediánem nebo modem
- Doplnění hodnotou středu třídy
- Doplnění nejpravděpodobnější hodnotou vypočítanou například regresí

Při doplňování chybějících hodnot je důležité mít na paměti kontext dat, protože ne vždy musí být chybějící hodnota chybou. Obecně nejlepším přístupem při automatickém doplňování hodnot je volit poslední zmíněný přístup, protože je při něm do rozhodovacího procesu zapojeno největší množství informací. Je také dobré mít na paměti, že atribut, který obsahuje mnoho chybějících hodnot, nemusí být pro pozdější trénování dolovacího algoritmu užitečný.

2.4.2 Čištění zašuměných dat

Šum je náhodná chyba nebo odchylka v zkoumaném atributu. Řešení problému zašuměných dat se může lišit v závislosti na druhu atributu, tedy zda je kategorický nebo numerický.

Pokud identifikujeme zašuměnou hodnotu v numerickém atributu, k jejímu vyčištění můžeme přistupovat podobně jako při doplňování chybějících hodnot. Záznamy s šumem je možné vynechat, nebo doplňovat pravděpodobnou hodnotu. K tomu můžeme využít regresi, nebo plnění do košů, kdy v seřazených datech doplňujeme například mediánem okolních hodnot.

U kategorických atributů k doplnění nemůžeme využít regresi. Spíše je možné zašuměná data nahradit pravděpodobnou hodnotou z oboru hodnot daného kategorického atributu, nebo zašuměné hodnoty úplně odstranit [4].

2.4.3 Řešení nevyváženosti dat

Pokud podkladní datová sada trpí malým počtem záznamů nějaké třídy, trénovaný algoritmus může vykazovat neoptimální výsledky. Při různých dolovacích úlohách (například klasifikaci, která je popsána později, hraje důležitou roli vyváženost tříd v datové sadě. V případě, kdy třídy nejsou rovnoměrně zastoupeny, může klasifikátor častěji přehlížet třídu, která je zastoupena minoritně.

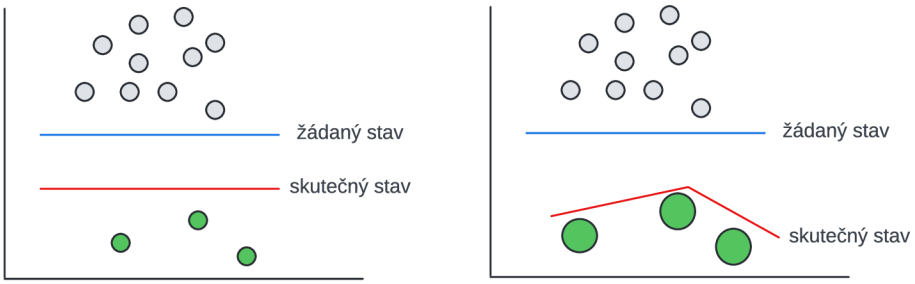
Tento problém je obzvláště nebezpečný pro svoji nenápadnost, jak bude ukázáno na příkladu.

Příklad 1 *Uvažujme lékařskou kliniku, na které jsou prováděny testy na blíže nespecifikovanou nemoc, kterou trpí přibližně 5 % populace. Kvůli ceně testu, která je vysoká, je vyvinut klasifikátor. K jeho vytrénování byla poskytnuta datová sada, v níž je 90 % záznamů z testů, které vyšly negativní, a zbývajících 10 % jsou záznamy pozitivní. Natrénovaný klasifikátor vykazuje přesnost 95 % při klasifikaci nemoci, což se na první pohled zdá být jako skvělý úspěch. Lepší náhled na skutečnou důvěryhodnost klasifikátoru ale získáme, pokud se podíváme na tzv. matici záměn.*

	Predikováno zdraví	*Predikována nemoc*
Zdravý člověk	*94 %*	*1 %*
Nemocný člověk	*4 %*	*1 %*

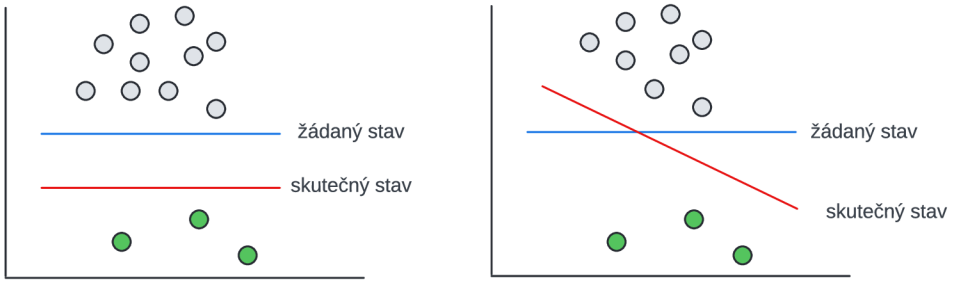
Protože klasifikátor nebyl natrénován na vyvážené datové sadě, nerozpozná méně zastoupenou třídu a v téměř každém případě predikuje zdraví pacienta.

Řešením problému nevyvážených dat může být nadvzorkování a podvzorkování [4]. Při nadvzorkování jsou do trénovací datové sady přidány duplicitně záznamy minoritně zastoupené třídy. Klasifikátor tedy dostane větší prostor pro naučení se vlastností minoritně zastoupené třídy. Častým problémem, který ovšem tento přístup přináší, je přeučení se. Klasifikátor se tak specializuje na několik konkrétních záznamů, ale nové záznamy poté může klasifikovat o to hůř (viz obrázek 2.3). Dalším problémem může být zvýšení výpočetní náročnosti při trénování na už tak obsáhlých datových sadách.



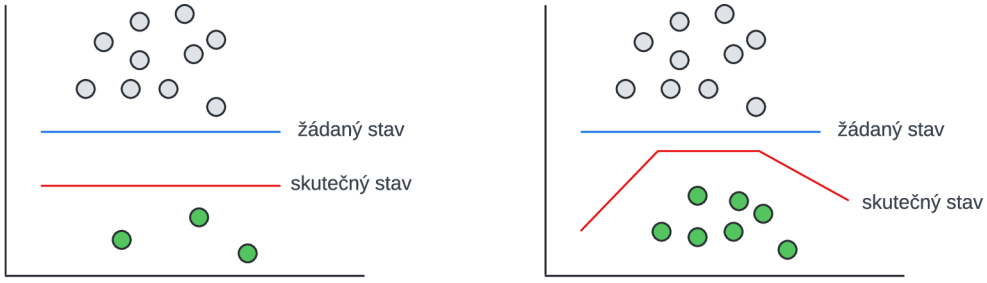
Obrázek 2.3: Příklad nadvzorkování - inspirováno [1]

Při podvzorkování jsou naopak některé záznamy majoritně zastoupené třídy vynechány. Může ale dojít ke ztrátě důležitých záznamů, které zachycují klíčové rozdíly mezi třídami, což také není žádoucí výsledek (viz obrázek 2.4).



Obrázek 2.4: Příklad podvzorkování - inspirováno [1]

Odpovědí na tyto nedostatky je technika SMOTE (z anglického Synthetic Minority Over-sampling Technique) [3]. Při ní je nevyváženost tříd řešena syntézou nových prvků. Ty jsou vytvořeny na základě existujících prvků minoritní třídy, doplněné o mírnou náhodnou odchylku (viz obrázek 2.5). Tím nedojde k žádnému z výše zmíněných problémů, i tak ale tato technika má svoje limitace. Pokud je minoritně zastoupená třída příliš komplexní, nebo se výrazně prolíná s jinou třídou, syntéza nových prvků nemusí problém nevyváženosti dat řešit. I tak se jedná o klasickou a efektivní techniku řešení problému vyváženosti tříd, která je pro svoji efektivitu a jednoduchost implementace stále využívána.



Obrázek 2.5: Příklad využití techniky SMOTE - inspirováno [1]

2.4.4 Transformace dat

Za účelem přípravy datové sady pro konkrétní dolovací algoritmus je v mnoha případech nutné data nějakým způsobem transformovat. Při transformaci dat je podstatné mít na paměti, pro jaký konkrétní algoritmus datovou sadu připravujeme, protože každý algoritmus má na data různé požadavky. Obecně řešíme následující strategie [4]:

- Vyhlazení dat. Souvisí s již dříve zmiňovanou problematikou čištění zašuměných dat, kdy pomocí regrese, rozdělování do košů nebo shlukování řešíme šum a nekonzistenci v datech.
- Konstrukce atributů. Z již stávájících atributů, které nejsou vhodné pro dolovací algoritmus, vytvoříme nový atribut, který bude v dolovacím procesu užitečnější.

- Agregace. Při přípravě dat pro datovou kostku nebo při seskupování více záznamů do jedné třídy je nutné vypočítat agregované nebo sumarizované hodnoty.
- Normalizace. Čísla z širokého nebo nestandardního rozsahu jsou normalizována na nižší rozsah, například na hodnoty od -1 do 1, nebo od 0.0 do 1.
- Diskretizace. Při přípravě datové sady pro algoritmus, který vyžaduje kategorické atributy, je nutné převést číselný atribut na několik intervalů nebo do několika košů.

2.4.5 Redukce dimenzionality

Často se vyskytujícím problémem u datové sady je příliš velký počet atributů. Tento problém se projevuje časovou náročností trénování dolovacích algoritmů. Z toho důvodu je potřeba redukovat dimenzionalitu datové sady.

Dimenzionalitu redukujeme konstrukcí atributů, jak bylo zmíněno v části transformace dat. Dalším způsobem redukce dimenzionality je extrakce atributů, v rámci níž jsou ve zdrojových datech hledány atributy, které nejlépe popisují zdrojová data s ohledem na cíle dolovacího projektu.

Korelační matice

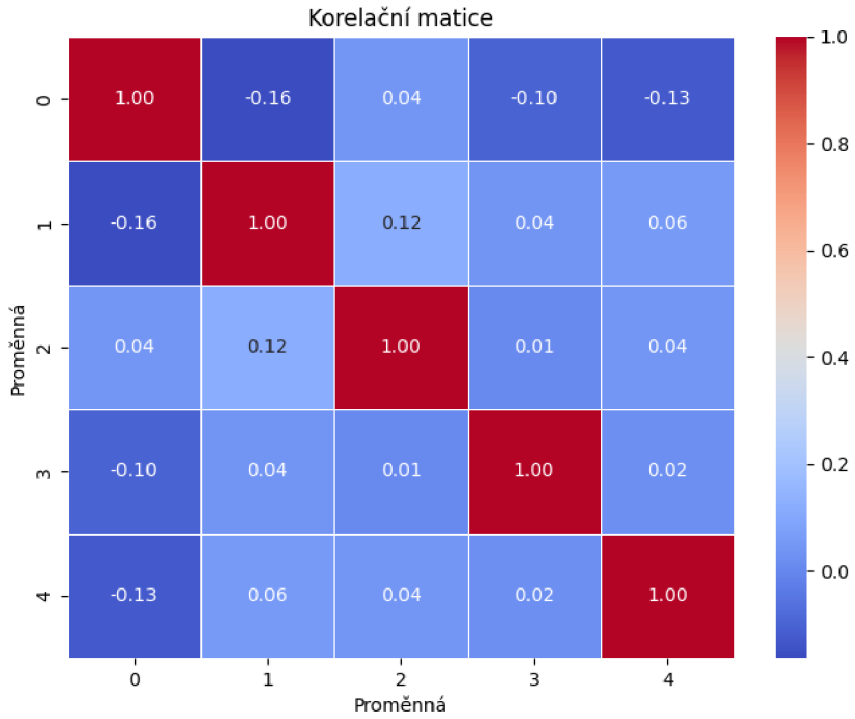
Pro výběr nadbytečných atributů lze použít silný nástroj ve formě korelační matice. Korelační matice je čtvercová matice, která zobrazuje korelační koeficienty každého páru atributů v datové sadě.

Definice 1 (Pearsonův korelační koeficient) *Pearsonův korelační koeficient r je spočítán vztahem:*

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

kde x_i a y_i jsou jednotlivé hodnoty dvou proměnných, $\bar{x}$ a $\bar{y}$ jsou průměrné hodnoty těchto dvou proměnných a n je počet záznamů.

Hodnoty korelačního koeficientu se nacházejí v rozmezí od -1 do $+1$ a vyjadřuje sílu linearity mezi dvěma proměnnými. Hodnota $+1$ značí silný kladný lineární vztah, hodnota -1 poté značí záporný lineární vztah. Korelační koeficient rovný 0 indikuje absenci lineárního vztahu mezi dvěma proměnnými. Příklad korelační matice můžeme vidět na obrázku 2.6.



Obrázek 2.6: Příklad jednoduché korelační matice

V některých případech je vhodné silně korelované atributy z datové sady odstranit, pokud je trénován algoritmus, který není vůči korelovaným hodnotám dostatečně robustní. Dalším důvodem pro odstranění korelovaných hodnot je poté redundance, kterou v sobě dva korelované atributy mohou nést, a tím zabránit přetrénování algoritmu. Ve výsledku ale závisí na kontextu dat a konkrétním dolovacím projektu.

2.5 Klasifikace a predikce

Klasifikace je druh datové analýzy, při níž jsou trénovány klasifikátory popisující důležité třídy dat. Tyto klasifikátory umějí řadit data na základě hodnot jejich atributů do tříd. Typickým případem je řazení zákazníků banky do skupin podle rizika. Predikce oproti tomu přiřazuje datům hodnoty, které můžou být spojité. Příkladem může být předvídání ceny nemovitosti na základě jejích atributů. Protože se model učí na trénovacích datech, jedná se z hlediska strojového učení o učení s učitelem. Tato část vychází z [4] a [10].

Průběh klasifikace začíná sbíráním a čištěním dat, následně jsou vybrány důležité atributy, jak bylo popsáno v oddíle 2.4. Následuje trénování modelu za použití historických dat. Model je následně vyhodnocen z různých hledisek, a poté jsou jeho pomocí predikována data.

Natrénovaný klasifikátor může být velice komplexní, a jak bylo ukázáno dříve (viz 2.4.3), jedno číslo nemusí zachytit komplexní realitu dolovacího projektu. Proto klasifikační metody hodnotíme z několika hledisek, například [4]:

- Přesnost. Jedná se o důležitou metriku zachycující, jaké procento záznamů umí klasifikátor správně zařadit.
- Rychlost a efektivita (zejména prostorová efekta, tedy nároky algoritmu na výpočetní paměť).
- Robustnost. Vyjadřuje míru odolnosti klasifikátoru vůči šumu a chybějícím hodnotám.
- Interpretovatelnost. Mnohdy je důležité, aby měl uživatel klasifikátoru důvěru v jeho rozhodnutí.

2.5.1 Rozhodovací stromy

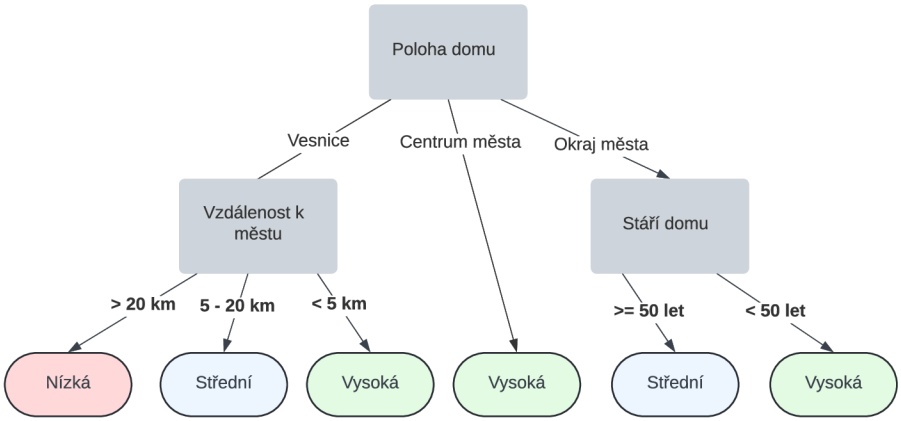
Rozhodovací strom je všestranný algoritmus strojového učení, který se učí predikovat třídu dat rekurzivním dělením vstupních hodnot na regiony. Rozhodovací strom má stromovou podobu, kde každý vnitřní uzel odpovídá jednomu rozhodovacímu kroku a každý listový uzel má hodnotu názvu třídy. Níže je popsán algoritmus rozhodovacího stromu.

Algorithm 1 Rozhodovací strom

Vstup : Datová sada D, sestávající ze záznamů a jejich označeních tříd; *seznamAtributu*, množinu kandidátních atributů na dělení; metodu VyberNejlepšíDělicíKritérium rozhodující o dělicím kritériu pro daný uzel.
Výstup : Rozhodovací strom.
Function `RozhodovacíStrom(`D, *seznamAtributu*`):`

```
Vytvořit uzel N;
if Všechny prvky v D patří stejné třídě C then
    return N jako listový uzel označený třídou C;
end
if Seznam atributů je prázdný then
    return N jako listový uzel označený majoritní třídou v D;
end
VyberNejlepšíDělicíKritérium(D, seznamAtributu) → kriterium;
Označit uzel N kriteriem dělení;
if Dělicí atribut má diskrétní hodnoty a je povoleno vícenásobné rozdělení then
    seznamAtributu ← seznamAtributu − deliciAtribut;
    for každý výsledek j dělicího kritéria do
        Nechť D_j je množina datových prvků v D splňující výsledek j;
        if D_j je prázdné then
            Připojit k uzlu N list označený majoritní třídou v D;
        end
        else
            Připojit uzel vrácený funkcí RozhodovacíStrom(D_j, seznamAtributu) k uzlu
             N;
        end
    end
end
return N;
```

Na obrázku 2.7 níže můžeme vidět příklad vygenerovaného rozhodovacího stromu, který predikuje hodnotu domu na základě atributů, které jsou o něm k dispozici.



Obrázek 2.7: Příklad jednoduchého rozhodovacího stromu

Výběr vhodného atributu pro dělení - metoda ID3

Proces výběru vhodného atributu (v algoritmu výše zaznačeného metodou VyberNejlepšíDělicíKritérium) je klíčový krok v procesu učení rozhodovacího stromu, protože nejvíce ovlivňuje celkovou efektivitu a vysvětlitelnost klasifikátoru. V metodě ID3 je pro zjištění nejlepšího atributu k dělení použit atribut, který přináší nejvyšší informační hodnotu.

Informace potřebná pro klasifikaci D, nebo také entropie D, je vypočten vztahem:

Definice 2 (Entropie)

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

kde p_i je nenulová pravděpodobnost že libovolná n-tice v D náleží do třídy C_i a je odhadnuta $\frac{|C_i,D|}{|D|}$. $Info(D)$ je průměrné množství informace potřebné k určení třídy n-tice v D. Logaritmus o základu dvou je použit proto, že informace je kódována v bitech.

Pokud bychom chtěli n-tice v D rozdělit podle atributu A který má v jedinečných diskrétních hodnot, označených jako $\{a_1, a_2, ..., a_v\}$. Při použití atributu A rozdělíme D na v podmnožin $\{D_1, D_2, ..., D_v\}$, kde D_j obsahuje n-tice D s výsledkem a_j z A. Každá podmnožina představuje jednu větev stromu z uzlu N. Ideálním výsledkem by byla podmnožina obsahující záznamy stejné třídy, běžnější je smíšení záznamů z více tříd v jedné podmnožině.

Informace, kterou potřebujeme po dělení pro dosažení přesného určení třídy, je

Definice 3 (Informace k atributu A)

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \cdot Info(D_j),$$

kde $\frac{|D_j|}{|D|}$ označuje část n-tic v datové sadě D, které náleží k dělení j.

Výsledný informační přínos je potom spočítán rozdílem entropie D a očekávanou informací nutnou ke klasifikaci n-tice z D při dělení podle A.

Definice 4 (Gain(A))

$$Gain_A = Info(D) - Info_A(D).$$

Význam $Gain(A)$ je tedy kolik informace získáme, pokud provedeme dělení podle atributu A. Pro dělení je následně zvolen takový atribut A, který má hodnotu $Gain(A)$ nejvyšší.

Split-point

V případě, že je atribut A spojitého charakteru, tedy v kroku přípravy dat neproběhla jeho diskretizace, musí být určen nejlepší dělicí hranice atributu A, tzv. Split-point.

Pro určení split-pointu je potřeba mít hodnoty v A seřazené. Pro každou dvojici a_i a a_{i+1} je spočítán split-point podle vztahu

Definice 5 (Split-point)

$$\textit{Split-point} = \frac{a_i + a_{i+1}}{2}$$

Následuje výpočet $Gain_A(D)$ pro každý split-point, vybrán je ten, který přináší největší množství informace. D je následně rozděleno do podmnožin D_1, v níž jsou n-tice z D splňující podmínku $A \leq split-point$, a D_2, jejíž n-tice splňují podmínku $A > split-point$.

Další vylepšení

Algoritmus rozhodovacího stromu má některá slabá místa, která jsou řešena metodami C4.5 a Gini index [4].

Přístup popsaný u metody ID3 zvýhodňuje atributy, které nabývají velkého množství hodnot. Metoda C4.5 pracuje s normalizovanými hodnotami $Gain_A(D)$, které normalizuje hodnotou $SplitInfo_A$.

Hodnota $SplitInfo_A$ zachycuje potenciální informace, které by přineslo dělení podle atributu A, s důrazem na celkovou magnitudu D. Výsledná rozhodovací hodnota se poté nazývá *GainRatio*.

Nevýhodou této metody zůstává výpočetně složitý krok v podobě logaritmu. Vylepšení v tomto ohledu přináší metoda Gini index, která nahrazuje $Info(A)$ hodnotou $Gini(A)$, která značí míru nejednoty celé datové sady D, pro jejíž výpočet není použit logaritmus.

Zhodnocení

Rozhodovací strom je základní a jednoduše pochopitelný algoritmus. Jeho velkými přednostmi jsou rychlost učení a snadná interpretovatelnost, protože jeho struktura je velmi transparentní a pro každý záznam je jasné, jak probíhal rozhodovací proces. Navíc je odolný vůči odlehlým hodnotám v datech.

Mezi jeho nevýhody patří náchylnost k přeučení a špatná škálovatelnost. Je totiž efektivní pouze tehdy, když se trénovací data vejdou do paměti.

2.5.2 Náhodné lesy

Jedná se o metodu rozšiřující algoritmus rozhodovacích stromů. Klíčová myšlenka zde spočívá ve vytvoření mnoha rozhodovacích stromů, které jsou každý natrénovány na jiném

vzorku dat. Při klasifikaci probíhá hlasování všech stromů, do které třídy bude vzorek zařazen [4].

Náhodný les byl v práci L. Breimana [2] zaveden takto:

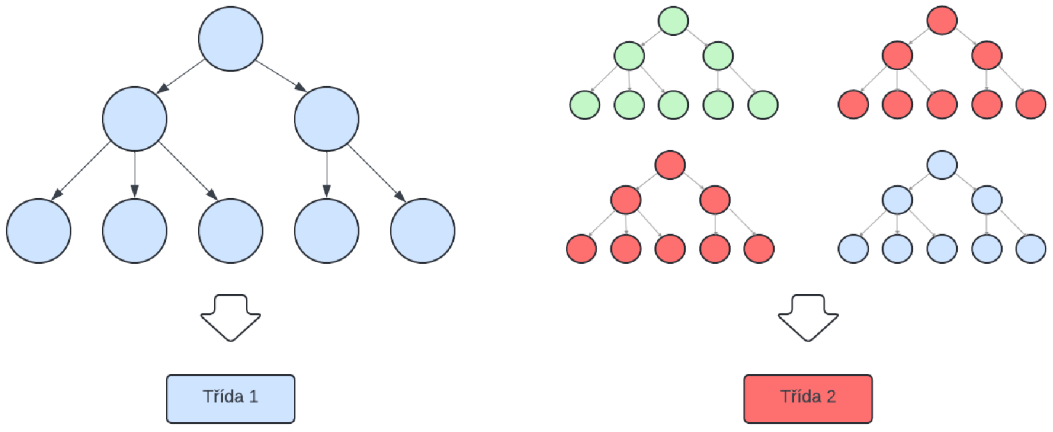
Definice 6 (Náhodný les) *Náhodný les je klasifikátor sestávající z kolekce klasifikátorů stromové struktury* $h(x, \Theta_k), k = 1, \ldots,$ *kde* Θ_k *jsou nezávislé vektory s identickým rozložením hodnot a každý strom vrhá jeden hlas pro nejvíce podporovanou třídu na vstupu* x.

Výběr trénovacího vzorku dat dílčího klasifikátoru probíhá pro každý klasifikátor náhodně z celé vstupní datové sady. Některé vzorky proto mohou být použity vícekrát. Jak je také uvedeno v tom samém článku, s přidáním více dílčích stromů do klasifikačního procesu nedochází k přetrénování, ale celková chyba limitně dosahuje generalizační chyby. Při opakovaném náhodném výběru totiž mohou být některé záznamy vynechány, s velkou pravděpodobností se ale jedná o malé množství případů, které v celkovém rozhodovacím procesu nemusí hrát žádnou roli [2].

Kromě náhodného výběru vzorků pro jednotlivé stromy dochází i k náhodnému výběru množiny uvažovaných atributů. Dalším používaným přístupem je poté les-RC, kdy je pro rozhodování vytvořen nový atribut skrze náhodnou lineární kombinaci existujících atributů.

Při klasifikaci pomocí náhodného lesa je nejdříve nutné určit počet dílčích rozhodovacích stromů, který bude vytvořen. Následně je vytvořeno stejné množství trénovacích vzorků dat, na každém je natrénován jeden dílčí strom. Klasifikace probíhá hlasováním, jak bylo zmíněno dříve.

Díky přístupu náhodného lesa ke klasifikaci jako k souboru několika podobných operací, je celá metoda značně škálovatelnější než rozhodovací strom, především díky možnosti paralelizace, ale také díky náhodnému výběru vzorků, který sám omezuje generování příliš velkého stromu. Další výhodou je poté zlepšení přesnosti oproti rozhodovacím stromům, kdy už nezáleží na přesnosti jednoho stromu, ale náhodného lesa jako celku. Nevýhodou náhodného lesa je poté nižší interpretovatelnost, obzvláště při srovnání s rozhodovacím stromem.



Obrázek 2.8: Princip fungování náhodného lesa v porovnání s rozhodovacím stromem

2.5.3 Naivní Bayesovská klasifikace

Bayesovská klasifikace je založena na statistice. Predikuje příslušenství k třídě na základě pravděpodobnosti, že daná n-tice by do třídy mohla patřit. Vychází z Bayesova vzorce:

Definice 7 (Bayesova klasifikace) *Vzorek je zařazen do třídy s maximální hodnotou* $P(C_i|X)$

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)},$$

je Bayesův vzorec, kde X *je vzorek dat a* C_i *je jedna ze tříd, do které může být vzorek zařazen.*

$$P(C_i) = \frac{|s_i|}{|S|},$$

kde s_i *je množina trénovacích vzorů ve třídě* C_i *a* S *je množina všech trénovacích vzorů.*

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i),$$

tedy apriori pravděpodobnost je spočítána vynásobením podmíněné pravděpodobnosti, že konkrétní vzorek x_k *náleží do třídy* C_i*.*

Pro diskrétní atribut:

$$P(x_k|C_i) = \frac{|s_{ik}|}{|s_i|},$$

kde s_{ik} *je množina trénovacích vzorů ze třídy* C_i *splňující podmínku, že hodnota jeho* k*-tého atributu je rovna* x_k*.*

Pro spojitý atribut:

$$P(X_k|C_i) = g(x_k, \mu_{Ci}, \sigma_{Ci}),$$

kde $g(x_k, \mu_{Ci}, \sigma_{Ci})$ *je Gaussova normální funkce.*

„Naivita“ přístupu spočívá v předpokladu, že proměnné v datové sadě jsou na sobě nezávislé, což se v praxi příliš často nestává. Je proto potřeba podle toho upravit datovou sadu a odstranit korelované atributy, nebo použít pokročilejší metody, které z naivní Bayesovské klasifikace vycházejí (např. Bayesovské sítě).

Další slabé místo naivního Bayesovského klasifikátoru je odhaleno, pokud má některá třída v trénovacích datech nulové zastoupení. V takovém případě je totiž hodnota $P(X|C_i)$ rovna nule, a tedy pravděpodobnost zařazení do třídy bude vždy nulová. Pro řešení tohoto problému se používá *Laplaceova korekce*, při níž je k magnitudě každé třídy přičtena malá nenulová konstantní hodnota, většinou 1. Tím dojde k zachování malé pravděpodobnosti zařazení klasifikovaného vzorku do třídy, která je jinak méně početná.

Naivní Bayesovská klasifikace má mnoho výhod, kvůli kterým je stále používána. Předně je implementace této metody snadná, a často dosahuje v porovnání podobné úspěšnosti jako jiné složitější klasifikátory, například rozhodovací strom [4] nebo logistická regrese [6], přičemž je výpočetně méně náročná. Další výhodou je snadná interpretovatelnost, protože rozhodnutí probíhá na základě pravděpodobností jednotlivých atributů. Také nevyžaduje velké množství trénovacích dat a funguje efektivně i na velkých datových sadách. Přes všechny tyto výhody ale většinou nedosahuje tak dobrých výsledků, jako komplexnější rozhodovací algoritmy.

2.5.4 Neuronové sítě

Algoritmus backpropagation je v dolovacích úlohách významným klasifikátorem z oblasti neuronových sítích. Vznikl v rámci mezioborových snah psychologů a neurobiologů. Skládá se z několika úrovní perceptronů, což je vstupně / výstupní mechanismus, který na základě vstupu, vektoru vah vstupů a vnitřního biasu vrací jedno výstupní číslo. Trénování takové neuronové sítě s sebou nese právě trénování vah vstupů všech perceptronů, čímž jsou trénovány prediktivní schopnosti neuronové sítě.

I když neuronové sítě potřebují značný čas pro natrénování a vykazují velice nízkou interpretovatelnost, způsobenou složitou vnitřní strukturou, jsou velice schopné ve zpracování zašuměných dat a rozpoznávání vzorců, na kterých nebyly natrénovány.

Jejich další výhodou jsou schopnost zpracovat spojitá data a možnost paralelního zpracování. Vynikají v případech, kdy je vztah mezi třídou a atributem nezřetelný a prokazují svoji účinnost v reálných dolovacích projektech.

Perceptron

Perceptron (nebo také neuron) je základní stavební prvek neuronové sítě, který je schopen binární klasifikace. Jeho vstupem je několik signálu, přičemž ke každému je přiřazena nějaká váha, a jeho výstupem je jedna binární hodnota. Perceptron počítá váženou sumu svých vstupních signálů, aplikuje na ni aktivační funkci a vrátí binární číslo, jehož hodnota závisí na tom, jestli výsledná hodnota překročila určitou hranici.

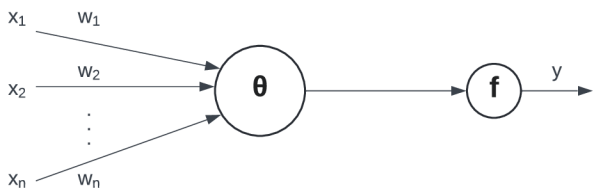
Matematicky je neuron definován následovně:

Definice 8 (Perceptron)

$$y = \begin{cases} 1, & if\ \sum_{i=1}^{n} w_i \cdot x_i + \theta \geq hranice \\ 0, & otherwise \end{cases}$$

kde w je vektor vah, x je vektor vstupních signálů, θ je bias neuronu, hodnota hranice je určena aktivační funkcí neuronu.

Fungování perceptronu je také zaznačeno na obrázku 2.9, kde f je aktivační funkce a ostatní značení je stejné.



Obrázek 2.9: Princip fungování perceptronu

Algoritmus, který popisuje trénování jednoho perceptronu, je následující:

1. Inicializace. Váhy w_i a bias θ jsou inicializovány náhodnými hodnotami.
2. Vstupy a aktivace. Pro každý vstup x_i perceptron spočítá váženou sumu ke které přičte bias θ. Tento součet je následně použit jako vstup pro aktivační funkci f,

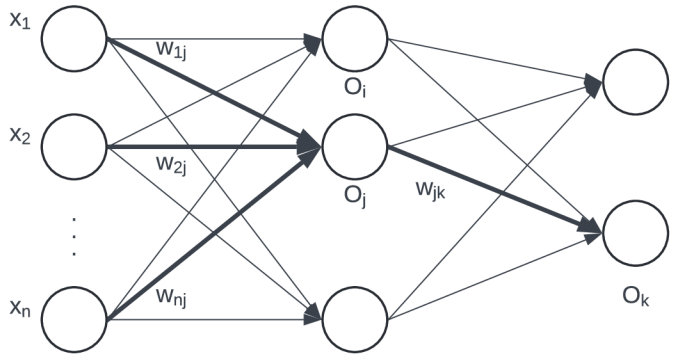
jejímž výstupem je číslo y. Tento výstup je porovnán se skutečnou třídou, kterou by měl perceptron určit.

3. Spočítání chyby. Je spočítán rozdíl mezi predikovanou třídou y a skutečnou třídou, čímž je určena chyba.
4. Aktualizace vah. Váhy vstupů jsou aktualizovaný podle používaného učicího pravidla.
5. Iterace. Proces se opakuje od kroku 2, pokud nebyl dosažen maximální počet iterací, nebo pokud nebyly splněny požadované podmínky přesnosti. V každé iteraci jsou zpřesňovány váhy perceptronu.
6. Konvergence. Výsledkem jsou nastavené váhy perceptronu.

Vícevrstvé perceptrony

MLP (angicky Multi-Layer Perceptron) představují podstatný pokrok oproti jednoduchému perceptronu tím, že spojují neurony do vrstev, kterých může být v modelu několik, čímž umožňují učení se složitějších vzorů v datech. Skládají se z jedné vstupní a jedné výstupní vrstvy, mezi nimiž je jedna a více dalších vrstev, které se nazývají skryté vrstvy (viz obrázek 2.10. Tyto skryté vrstvy slouží jako mezistupně zpracování. Spojením neuronů do několika vrstev je značně navýšena výpočetní a klasifikační schopnost modelu oproti jednomu neuronu.

Vstupní vrstva dostává na vstup vektory čísel reprezentujících vstupní data. Tyto vstupy jsou následně propagovány skrze jednu nebo více skrytých vrstev, v nichž každý neuron spočítá vážený součet svých vstupů a výsledek pošle skrze aktivační funkci do další vrstvy, kde to samé číslo slouží jako vstup. Skrytá vrstva slouží jako exktraktor nelineárních rysů tím, že vstupní data transformuje na reprezentaci vyšší úrovně, čímž je dělá přístupnější pro následnou klasifikaci. Na závěr jsou výstupní vrstvou agregovány výsledky poslední skryté vrstvy a je vyprodukována predikce nebo rozhodnutí celé neuronové sítě.



Obrázek 2.10: Neuronová síť bez zpětné vazby s jednou skrytou vrstvou, O zde značí aktivační funkci, převzato z [4]

Algoritmus backproapgation

Algoritmus backpropagation (volně přeloženo jako zpětná propagace) se odlišuje od přístupů zmíněných výše tím, že do procesu trénování přidává část zpětného šíření chyby. Algoritmus je následující:

1. Inicializace. Všechny váhy w_{ij} a všechny biasy θ_j jsou inicializovány náhodnými malými hodnotami.

2. Šíření vstupu k výstupu. Pro každý trénovací vzor je pro všechny neurony ve všech skrytých a ve výstupní vrstvě spočítáno:

$$I_j = \sum_i w_{ij} \cdot O_i + \theta_j,$$

$$O_j = \frac{1}{1 + e^{-I_j}},$$

   kde I_j je vážený součet vstupů neuronu j, w_{ij} jsou váhy vstupů spojující neuron i a neuron j, O_i je výstup neuronu i, θ_j je bias neuronu j.

3. Zpětné šíření chyby. Pro každý neuron výstupní vrstvy je spočítána chyba:

$$Err_j = O_j(1 - O_j)(T_j - O_j),$$

   kde T_j je výstup, který měl vyjít.

   Pro každý neuron každé skryté vrstvy je spočítána chyba:

$$Err_j = O_j(1 - O_j) \sum_k Err_k \cdot w_{jk},$$

   podle toho jsou modifikovány všechny váhy w_{ij} a biasy θ_j následovně:

$$\begin{aligned} \Delta w_{ij} &= l \cdot Err_j \cdot O_i, \\ \Delta \theta_j &= l \cdot Err_j, \end{aligned}$$

$$\begin{aligned} w_{ij} &= w_{ij} + \Delta w_{ij}, \\ \theta_j &= \theta_j + \Delta \theta_j \end{aligned}$$

   kde $l \in \langle 0; 1 \rangle$ je koeficient učení. Koeficient by měl začínat na maximu a v průběhu trénování klasifikátoru se postupně snižovat, například $l = 1/t$, kde t je číslo aktuální iterace.

Neuronová síť se přestává učit, když dosáhne předem určeného počtu iterací, přesáhne definovanou hranici správně určených vzorků, a nebo jsou změny ve váhách příliš malé.

Oproti teorii se v praxi osvědčily v některých případech odlišné přístupy. To se týká například frekvence aktualizování vah a biasů, které by se měly počítat po celé iteraci, v praxi se mění po každém vzorku, což vede k rychlejší konvergenci celé sítě. Jiným příkladem je předpoklad, že čím více vrstev neuronová síť má, tím lepší budou její výsledky. To se ale u algoritmu backpropagation ukazuje jako mylná představa, protože zpětné šíření chyby se směrem od výstupní vrstvy ke vstupní vrstvě oslabuje, takže na vrstvy nejblíže vstupní vrstvy nemá téměř žádný účinek. Řešením tohoto problému je předtrénování neuronové sítě, aby nebyla inicializována náhodnými čísly.

Zhodnocení

Neuronová síť typu MLP použitá pro klasifikaci má spoustu výhod, a to jejich všestrannost a přirozená paralelita. Do neuronové sítě můžou vstupovat spojité i nespojité hodnoty, bez jakéhokoliv zvýšení výpočetní náročnosti. Dalšími výhodami z hlediska přípravy dat jsou tolerance zašuměných dat a fakt, že u podkladní datové sady není vyžadována analýza vztahů mezi atributy.

Hlavními nevýhodami MLP jsou dlouhé trénovací časy a nízká interpretovatelnost dat, protože analýza významu vah jednotlivých neuronů může být velmi náročný proces. Další nevýhodou je poté nutnost určovat parametry a topologii neuronové sítě [10].

2.6 Dolování asociačních pravidel

Dolování asociačních pravidel je silný a užitečný nástroj při odkrývání skrytých vzorů a vztahů ve velkých datových sadách. Spočívá v analýze frekventovaných množin, tedy množin prvků, které se spolu často vyskytují v jednom záznamu. Jako příklad bývá často uváděn nákupní košík, ve kterém můžeme často najít například chléb a šunku, méně často potom brokolici a chlazený hotový pokrm. Tato část vychází především z [4].

Frekventovaná množina je podmnožina položek nebo atributů datové sady, které se dostatečně často vyskytují společně v záznamu, čímž přesahují předem definované minimum hodnoty podpory. Jsou tedy považovány za smysluplné nebo významné oproti náhodným vzorům v datech.

Asociační pravidlo je logický výraz který popisuje vztah mezi dvěma množinami prvků v datové sadě. Může být interpretováno jako pravidlo tvaru „pokud, tak“, nebo „if-then“, kde jedna strana pravidla (antecedent) implikuje druhou stranu pravidla (konsekvenci). Asociační pravidla jsou vydolována z frekventovaných množin, přičemž každé pravidlo indikuje potenciální korelaci nebo asociaci mezi specifickými kombinacemi prvků.

2.6.1 Podpora a spolehlivost

Spolehlivost (anglicky confidence) a podpora (anglicky support) jsou dvě klíčové hodnoty, kterými je určována důvěryhodnost nalezených pravidel. Podpora kvantifikuje četnost, s kterou se kombinace položek objevuje v datové sadě. Je spočítána jako poměr počtu transakcí obsahujících danou položku a počtu transakcí celkově.

Definice 9 (Podpora a spolehlivost) *Pravidlo* $X \implies Y$ *má podporu* s *v množině transakcí* D*, jestliže* $s\%$ *transakcí v* D *obsahuje množinu položek* $X \cup Y$*. Pravidlo* $X \implies Y$ *má ve množině transakcí* D *spolehlivost* c*, jestliže* $c\%$ *transakcí, které obsahují* X*, obsahují také* Y*, tedy:*

$$s(A \implies B) = P(A \cup B),$$
$$c(A \implies B) = P(A|B).$$

2.6.2 Asociační pravidla

Hledání asociačních pravidel probíhá právě na základě spolehlivosti a podpory, a to ve dvou krocích:

1. Nalezení množin položek, které splňují podmínku minimální podpory, tedy frekventovaných množin.

2. Generování pravidel z frekventovaných množin, které splňují podmínku minimální podpory i spolehlivosti, tedy silných asociačních pravidel.

Asociační pravidla se dělí podle několika kritérií na:

- Booleovské vs. kvantitativní. Booleovské asociační pravidla jsou založeny na binárních atributech, vychází z přítomnosti nebo nepřítomnosti položky v nákupním koši. Oproti tomu kvantitativní asociační pravidla zahrnují numerické atributy. Místo přítomnosti a nepřítomnosti se zaměřují na numerické hodnoty asociované s položkami. Příkladem může být booleovské asociační pravidlo

$$(\text{koupí myš, koupí klávesnici}) \implies (\text{koupí počítač}),$$

  oproti kvantitativnímu

$$(\text{věk} > 30, \text{příjem} > 50\ 000) \implies (\text{koupí počítač}).$$

- Jednodimenzionální vs. vícedimenzionální. Jednodimenzionální pravidla zahrnují jediný atribut, vícedimenzionální se oproti tomu týkají více atributů a nezaměřují se pouze na položky přítomné v nákupním koši, ale i na demografické údaje nakupujícího apod.

- Jednoúrovňová vs. víceúrovňová. Jednoúrovňová asociační pravidla neuvažují hierarchii položek, zatímco víceúrovňová ano. U víceúrovňových asociačních pravidel proto získáváme možnost analyzovat konkrétní poddruhy položek, které zákazník nakupuje. Příkladem můžou být mobilní telefony, které jsou chytré střední třídy, chytré vyšší třídy, ale i tlačítkové. Tyto druhy ale při analýze jednoúrovňových asociačních pravidel zanedbáváme.

2.6.3 Algoritmus Apriori

Základním algoritmem kterým mohou být dolovány frekventované množiny, z nichž jsou následně generovány asociační pravidla, je algoritmus Apriori. Vychází z „Apriori“ vlastnosti, která říká, že pokud má být množina frekventovaná, musí být frekventované i všechny její podmnožiny.

Algoritmus Apriori je složen ze dvou základních kroků:

1. Slučovací krok. V tomto kroku jsou algoritmem generovány kandidátní k-množiny prvků spojováním frekventovaných $k-1$ množin se sebou samými. Tito kandidáti jsou tvořeni kombinací dvou $k-1$ množin, pokud jsou jejich první $k-2$ prvky identické. Výsledná k-množina je vytvořena připojením poslední položky jedné $k-1$ množiny položek k druhé, čímž je zajištěno, že nedojde k duplicitám. Tento proces vytvoří sadu kandidátních množin C_k.

2. Vylučovací krok. Jakmile jsou vygenerovány kandidátní množiny C_k, je cílem tohoto kroku odstranění těch, které pravděpodobně nebudou frekventované. Protože C_k obsahuje frekventované i nefrekventované množiny, určování počtu každého kandidáta skrz

procházení celé databáze může být výpočetně velmi náročné. Aby bylo předejito procházení celé databáze, je využita Apriori vlastnost: pokud jakákoliv $k-1$-podmnožina kandidátní k množiny není frekventovaná, potom tento kandidát nemůže být frekventovaný. Proto každý kandidát, jehož $k-1$ podmnožiny nejsou frekventované, může být z C_k odstraněn. Tento proces může být prováděn efektivně díky uchovávání hašovacího stromu všech frekventovaných množin.

Obrovské množství čtení z transakční databáze a obrovské množství kandidátů na frekventované množiny, u kterých je nutné poměrně složitě počítat jejich podporu, vedla k identifikaci a vylepšování algoritmu v několika oblastech [4]:

- Redukce počtu průchodů transakční databází,
- Snížení počtu kandidátů,
- Usnadnění výpočtů podpory kandidátů.

2.6.4 Efektivní dolování frekventovaných vzorů: FP-growth

Přes různá vylepšení má algoritmus Apriori vážné nedostatky. Stále je možné, že v rámci něho bude vygenerováno velké množství kandidátních množin. Pokud by bylo nalezeno 10^4 kandidátních 1-množin, bude z nich vygenerováno více než 10^7 kandidátních 2-množin. Dalším faktem je, že může opakovaně procházet celou databázi a kontrolovat velkou množinu kandidátů neustálým porovnáváním. Počítat takto minimální podporu je výpočetně velmi náročné.

Zajímavou metodou, která se vydává přístupem „rozděl a panuj“, je FP-growth (z anglického Frequent Pattern - frekventované vzory) navržený Hanem J. a Peiem J. [5], který nejdříve komprimuje informace z databáze vytvořením FP-stromu kandidátních množin, ve kterém jsou zachovány všechny informace o asociacích množin. Zkomprimovanou databázi následně rozdělí do podmíněných databází, které jsou propojeny s frekventovaným prvkem. Pro každý prvek jsou proto nutné zkoumat jen záznamy, které jsou s ním propojené.

Algorithm 2 FP-growth [4]

Vstup : Transakční databáze D, minimální podpora min_sup.
Výstup : Kompletní množina frekventovaných vzorů

FP-strom je vytvořen v následných krocích:

- Projdi jednou transakční databázi D. Sesbírej množinu frekventovaných prvků F a hodnoty jejich podpory. Seřaď F podle podpory sestupně do seznamu frekventovaných prvků L.
- Vytvoř kořen FP_stromu T a označ ho jako *null*. Pro každou transakci t v D proveď následující:
  - Seřaď frekventované množiny v t podle pořadí L. Seřazený frekventovaný seznam prvků v t nechť je $[p|P]$, kde p je první prvek a P je zbytek seznamu.
  - Pokud má strom T potomka N takového, že $N.item\_name = p.item\_name$, inkrementuj čítač N o 1, jinak vytvoř nový uzel N s čítačem rovným 1, jeho rodičovská větev ať je napojená na T a propoj ho s ostatními uzly, které obsahují stejné $N.item\_name$, tím je vytvářena struktura node-links.
  - Opakuj, dokud je P neprázdné.
- Zavolej metodu `FP_growth` (implementovaná níže)

Function `FP_growth`($Tree$, α)**:**
  if *Tree obsahuje jedinou cestu* P **then**
    foreach *kombinaci* β *uzlů v cestě* P **do**
      generuj vzor $\beta \cup \alpha$ s hodnotou podpory = minimální podpora uzlu v β
    end
  else
    foreach a_i *v záhlaví* $Tree$ **do**
      generuj vzor $\beta = a_i \cup \alpha$ s podporou = a_i.podpora
      vytvoř základ podmíněného vzoru β
      vytvoř podmíněný FP_strom β, označený jako $Tree_\beta$
      if $Tree_\beta \neq \emptyset$ **then**
        zavolej `FP_Growth`($Tree_\beta$, β)
      end
    end
  end

Příklad

Pro ilustraci uvažujme následující příklad. Mějme transakční databázi D 2.1. Při jejím prvním průchodu jsou nalezeny 1-množiny, tedy množiny s kardinalitou 1. Ty jsou v tabulce 2.1 ve sloupci frekventované položky. Tyto frekventované položky jsou v dalším kroku seřazeny podle počtu výskytů sestupně jak v tabulce počtu výskytů (obrázek 2.11, levá část), tak v sloupci frekventovaných položek.

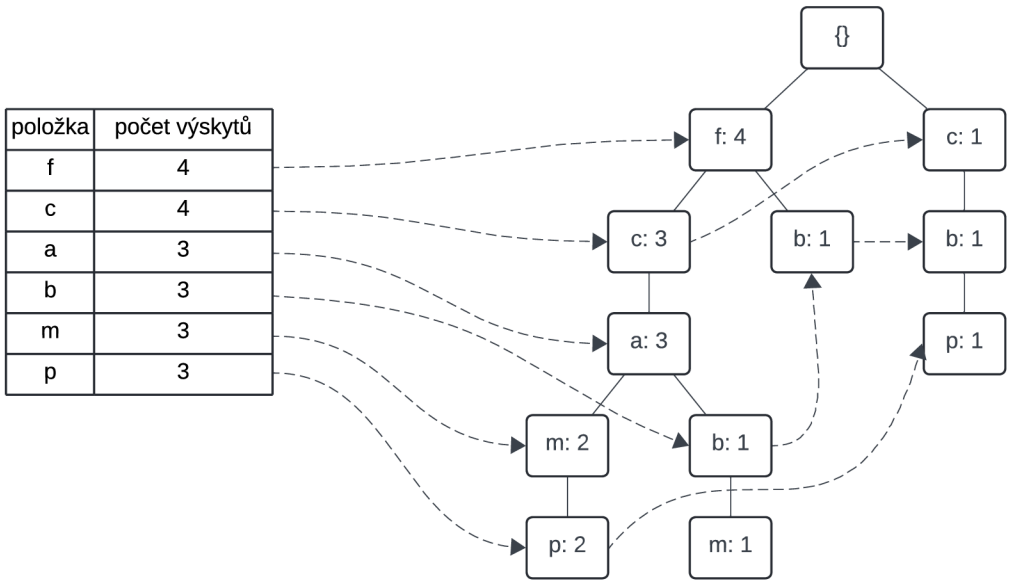
Tabulka 2.1: Transakční databáze D

TID	Položky	Frekventované položky
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

Po seřazení položek začne samotná konstrukce FP-stromu. První transakce obsahuje (seřazeno sestupně podle celkového počtu výskytů) hodnoty f, c, a, m, p. Za kořen *null* je napojeno f, za něj c, atd. Získáme tedy strom, v němž každý uzel má jednoho potomka a každý uzel má v čítači výskytů hodnotu 1. Následuje druhý transakce f, c, a, b, m. Položky f, c, a už v FP-stromu máme, tedy jim pouze zvýšíme čítač o 1. Položky b, m ovšem v FP-stromu zatím nejsou, proto je uzel b vytvořen a napojen na a, poté následuje uzel m, oba mají hodnotu 1.

V další transakci f, b je nejdříve výskyt f připočten do uzlu f (nyní dosahuje hodnoty 3), uzel b ovšem zatím není bezprostředním následníkem uzlu f, proto je vytvořen nový uzel b, který je za uzel f napojen. Následně je strukturou node-link propojen s uzlem b, který vznikl při zpracování předchozí transakce. U transakce c, b, p je vytvořena nová větev s uzly c, b a p a tyto uzly jsou opět propojeny strukturou node-link s dříve vytvořenými uzly stejného jména. Poslední transakce f, c, a, m, p už pouze zvýší čítače u větve vytvořené první transakcí a topologii stromu nijak nemění.

Výsledný FP-strom vidíme na obrázku 2.11.



Obrázek 2.11: Konstrukce FP stromu. Přerušovaná spojnice zobrazuje tzv. node-links.

Zhodnocení

FP-strom má dvě velké výhody, a to že je kompletní a kompaktní. Kompletnost se projevuje tím, že v struktuře FP-stromu je udržována kompletní informace pro pozdější získávání frekventovaných množin. Kompaktnost je poté zapříčiněna absolutní redukcí irelevantních informací, protože všechny nefrekventované položky jsou smazány. Tyto položky jsou dále seřazené podle frekvence výskytu, tedy tím spíše budou blízko kořene stromu a sdílené mezi transakcemi. FP-strom také nikdy nebude větší než původní databáze. V nejhorším případě by měl zcela odlišné prvky, v tom případě by ale také neexistovaly žádné frekventované množiny.

Nevýhodami FP-stromu je náročnost jeho budování a možná velikost. Košatý FP-strom se nemusí vejít do paměti. Ve výsledku je ale metoda mnohem rychlejší než Apriori algoritmus a další metody [5].

2.6.5 Hodnocení asociačních pravidel

Asociační pravidla jsou generována vždy na základě splnění minimální hodnoty nějaké metriky. Již dříve byly zmíněny podpora a spolehlivost. Tyto metriky jsou objektivní a lehce použitelné i vysvětlitelné. Nejsou to ale jediné metriky, které můžeme u asociačních pravidel posuzovat. Silberschatz a Tuzhilin [9] představili subjektivní metriky zajímavosti asociačního pravidla. Pravidlo je podle nich zajímavé, pokud je neočekávané, překvapivé, přináší novou informaci, nebo akční, tedy že na základě jeho odhalení lze vykonat akci, a věří, že tyto dvě subjektivní metriky spolu souvisejí.

Posouzení těchto subjektivních metrik je na konečném zákazníkovi, pro kterého jsou dolovací úlohy vypracovány, pro datového analytika se ale nabízí ještě několik jiných metrik, které hodnotí asociační pravidla z více úhlů a každá svým kouskem přispívá ke zprávě o celkové kvalitě a použitelnosti vydolovaného asociačního pravidla. Kromě podpory a spolehlivosti jimi jsou:

- Lift (zdvih). Je spočítán následovně:

$$Lift(A \to B) = \frac{s(A \cup B)}{s(A) \cdot s(B)}.$$

  Tedy porovnává pravděpodobnost, že se dva prvky objeví společně ku pravděpodobnosti, že se objeví kterýkoliv z nich. Hodnota větší než 1 značí pozitivní korelaci prvků, tedy že se spolu objevují častěji, než bychom očekávali při zcela náhodném výskytu. Hodnota menší než 1 ukazuje negativní korelaci, tedy že jeden prvek snižuje pravděpodobnost výskytu druhého. Hodnoty kolem 1 značí vzájemnou nezávislost těchto dvou prvků. Lift je symetrický a nebere v potaz směr implikace asociačního pravidla.

- Conviction (přesvědčivost). Je spočítána jako

$$Conviction(A \to B) = \frac{s(A) \cdot s(\neg B)}{s(A \cup \neg B)},$$

  tedy kvantifikuje, do jaké míry výskyt první položky ovlivňuje nepřítomnost následné položky. Významem se podobá spolehlivosti a stejně jako u liftu výsledek vyšší než 1 značí pozitivní závislost. Nevýhodou je, že obor hodnot přesvědčivosti je neohraničený.

- Leverage (vliv). Je spočítán vztahem

$$Leverage(A \rightarrow B) = s(A \cup B) - s(A) \cdot s(B).$$

  Měří rozdíl mezi frekvencí společného výskytu dvou položek a frekvencí, která by se očekávala, kdyby byly tyto dvě položky nezávislé. Hodnoty leverage se pohybují v rozmezí od -1 do 1, kdy 0 značí nezávislost, záporné hodnoty značí negativní korelaci a kladné hodnoty značí pozitivní korelaci. Oproti liftu, kterému je leverage podobný, bere více v potaz frekvenci současného výskytu. Lift oproti tomu může být vysoký i pro položky s poměrně nízkým počtem výskytů.

2.7 Shrnutí

V této kapitole jsme se věnovali teoretickému základu dolování znalostí z databází s ohledem na implementaci řešení představeného později v této práci. Nejdříve byly shrnuty zdroje dat vhodné pro dolování, jejich podoby, specifika a dolovací úlohy, pro které jsou vhodné. Následně byl představen model CRISP-DM, standardní metodologie, využívaná při dolovacích projektech. Další část kapitoly byla věnována předzpracování dat, kdy byly zevrubně představeny nedostatky, které můžou reálná data obsahovat, a způsoby, jak tyto nedostatky řešit. Následující podkapitola se zaměřila na problematiku klasifikace a predikce. Byly představeny 4 významné algoritmy, způsob, jakým jsou nejčastěji implementovány, jejich limitace a varianty. Na závěr jsme se věnovali dolování asociačních pravidel, byly představeny teoreticky, následně byl rozebrán algoritmus Apriori, jeho limitace a vylepšení, a algoritmus FP-growth. Na závěr byly rozebrány různé metriky hodnocení asociačních pravidel.

Kapitola 3

Dolování znalostí v jazyce Python

V této kapitole je prostor věnován popisu programovacího jazyka Python, následně je popsána webová aplikace Jupyter Notebook a dále jsou popsány knihovny použitelné k implementaci dolovacího projektu v prostředí jazyka Python. Tento jazyk byl pro implementaci vybrán pro svůj bohatý systém knihoven udržovaný rozsáhlou komunitou, v nichž jsou dolovací algoritmy již implementované. Různé knihovny, které budou představeny později v této kapitole se navzájem doplňují a s výsledky dolovacích úloh je možné následně pracovat v kterémkoliv prostředí dle volby datového analytika. Další výhodou pro tento projekt byla škálovatelnost a možnost použití knihoven usnadňující zpracování velkých objemů dat. Poslední výhodou je poté svobodné prostředí jazyka Python, tedy nezávislost na jakýchkoliv proprietárních produktech.

3.1 Popis jazyka

Jazyk Python[1] je programovací jezyk, který byl navržen roku 1991 Guidem van Rossumem. Podle indexu TIOBE[2] se Python dlouhodobě řadí mezi nejoblíbenější programovací jazyky, v době vzniku této práce je vůbec nejoblíbenějším jazykem.

Python je vysokoúrovňový, interpretovaný, objektově orientovaný, dynamicky typovaný jazyk. Velký důraz je u něj kladen na čitelnost a jednoduchost použití. Liší se syntaxí oproti ostatním jazykům, které používají pro zanoření řidících sekvencí závorky tím, že využívá různé úrovně odsazení. Jednoduchost použití je podpořena automatickou správou paměti, o kterou se stará *garbage collector*. Jakákoliv paměť nutná pro běh programu je automaticky přidělena a následně automaticky dealokována.

Od svého vzniku měl jazyk 3 velké verze. V současnosti je mimo verzi Python 3, která byla vydána v roce 2008, ještě používána verze jazyka Python 2 uvedená v roce 2000. Tyto verze spolu nejsou kompatibilní a poslední verze Pythonu 2 vyšla v roce 2020.

3.2 Webová aplikace Jupyter Notebook

Webová aplikace Jupyter Notebook je zaštiťována projektem Jupyter[3] a jedná se o aplikaci pro tvorbu a sdílení výpočetních dokumentů s podporou různých programovacích jazyků. Dokument spustitelný v prostředí Jupyter Notebook má příponu `.notebook`, je kódovaný

[1] https://www.pythong.org
[2] https://www.tiobe.com
[3] https://www.jupyter.org

ve formátu JSON a skládá se z metadat, výpočetních buněk a popisných buněk, které používají formát Markdown. V praxi je tak velmi snadné komentovat a vysvětlovat jednotlivé výpočtení kroky, které můžou být prováděny například v jazyce Python.

3.3 Použité knihovny

Jak bylo zmíněno dříve, Python se vyznačuje vysokým počtem různých knihoven, které mohou být použity. Protože každá knihovna se zaměřuje na určitý problém, který je při implementaci dolovacího projektu řešit, je zde uvedeno relativně velké množství balíčků. Některé z nich usnadňují práci s daty (Pandas, Dask), některé umožňují vizualizaci dat a tvorbu grafů (Matplotlib, Seaborn) a jiné mají implementované algoritmy popsané v minulé kapitole (scikit-learn, mlxtend).

3.3.1 Knihovna Pandas

Knihovna Pandas[4] se zaměřuje na manipulaci a analýzu dat. Jejími základními datovými typy jsou `DataFrame`, dvoudimenzionální datová struktura do které může být uložena tabulka, a `Series`, jednorozměrná struktura reprezentující sloupec `DataFrame`.

Jedna z klíčových vlastností Pandas je způsob, jakým se dokáže vypořádat s chybějícími daty. Přináší metody pro detekování chybějících dat, jejich odstranění nebo doplnění, čímž značně zjednodušuje proces čištění dat.

Celkově je možné pomocí metod knihovny Pandas načíst data, analyzovat, řadit, transformovat a čistit. Díky provázanosti s ostatními balíky používanými pro vizualizaci dat je Pandas nepostradatelným nástrojem při analýze dat.

3.3.2 Knihovna Numpy

Numpy[5] je další ze základních knihoven Pythonu. Poskytuje efektivní nástroje pro numerické výpočty nad velkými (potenciálně vícedimenzionálními) poli nebo maticemi, jmenovitě například datový typ `numpy.ndarray`. Zaměřuje se na vysoký výkon výpočtů, který je zajištěn díky metodám implementovaných v jazycích C a Fortran. Sám slouží také jako základ pro ostatní knihovny, které jeho metody často využívají, a doplňuje se s ostatními knihovnami.

3.3.3 Knihovna Dask

Dask[6] je flexibilní knihovna pro paralelní výpočty, která se snaží vyřešit problémy komplexních výpočtů nad velkými datasety, které by jinak nároky přesáhly paměť počítače. Rozšiřuje tím funkcionalitu knihoven jako Pandas nebo Numpy.

Dask poskytuje vysokoúrovňové rozhraní, které napodobuje rozhraní výše zmíněných knihoven, čímž značně usnadňuje proces učení a používání novým uživatelům. Samotné výpočty poté mohou probíhat nejen ve více vláknech, ale také napříč více stroji, přičemž Dask zajišťuje rozvržení výpočetních úkonů, dělení dat, ošetření chyb i rovnoměrné rozložení výpočtů mezi výpočetními uzly.

[4] https://pandas.pydata.org
[5] https://numpy.org
[6] https://www.dask.org

3.3.4 Knihovna Matplotlib

Knihovna Matplotlib[7] s sebou přináší metody pro tvorbu statických, ale také interaktivních a animovaných vizualizací dat. Podporuje velké množství grafů (čárové, bodové, sloupcové, koláčové grafy, histogramy a mnoho dalších) a poskytuje možnost kompletního přizpůsobení grafu potřebám uživatele. Knihovna je také kompatibilní s výše zmíněnými objekty (`numpy.ndarray`, `pandas.dataFrame`) a je dobře kombinovatelná s knihovnou Seaborn. Grafy vygenerované touto knihovnou jsou ve vysoké kvalitě a tím pádem i použitelné ve vědeckých publikacích, ale také prezentacích nebo reportech v soukromé sféře.

3.3.5 Knihovna Seaborn

Seaborn[8] je knihovna Pythonu pro vizualizaci dat, založená na výše zmíněné knihovně Matplotlib. Stejně jako knihovna Matplotlib je propojená s knihovnami Pandas a Numpy, ale nabízí oproti ní i širší seznam grafů (jedná se o statistické grafy zahrnující v sobě více informace, například houslový graf, boxplot, nebo lineární regresi), jejichž statistické hodnoty jsou počítány automaticky. Umožňuje tak jednoduché vypracování složitějších analytických úkolů.

3.3.6 Knihovna scikit-learn

Knihovna scikit-learn[9] poskytuje nástroje pro mnoho složitějších analytických úkolů, včetně strojového učení nebo dolování dat. Velká výhoda knihovny je jednoduché a konzistentní aplikační rozhraní, které usnadňuje používání a zjednodušuje analýzu pomocí různých nástrojů.

Nabízí množství algoritmů učení s učitelem i bez učitele, včetně klasifikačních, regresních nebo shlukovacích algoritmů, jmenovitě algoritmy lineární regrese, SVM, náhodné lesy, k-means shlukování a mnoho dalších. Spolu s těmito modely přináší také metriky měření jejich úspěšnosti a výkonu, což je nezbytná podmínka pro validaci výsledků modelu.

Další důležitou součástí jsou metody pro předzpracování dat, které tato knihovna také nabízí, včetně normalizace atributu, ošetření chybějících hodnot nebo kódování kategorických atributů. Knihovna je taky dobře integrovaná s dříve zmíněnými knihovnami včetně Numpy, Pandas nebo Matplotlib, což jen dále usnadňuje získání a prezentaci výsledků.

3.3.7 Doplňkové knihovny imbalanced-learn a mlxtend

Knihovny imbalanced-learn[10] a mlxtend[11] jsou doplňkovými knihovnami navazující na Numpy, Pandas a Matplotlib, především ale rozšiřují funkcionalitu knihovny scikit-learn. Obě nabízí jednoduché použití v již existujících projektech, kde jsou využity všechny výše zmíněné knihovny.

Knihovna imbalanced-learn se zaměřuje specificky na problematiku nevyváženosti dat a implementuje metody pro nadvzorkování a podvzorkování dat. Komunikuje jak s knihovnou Pandas, tak i její nástavbou Dask, což výrazně usnadňuje její použití.

[7] https://matplotlib.org
[8] https://seaborn.pydata.org
[9] https://scikit-learn.org
[10] https://imbalanced-learn.org/stable/
[11] https://rasbt.github.io/mlxtend/

Knihovna mlxtend rozšiřuje funkcionalitu knihovny scikit-learn o implementaci dalších funkcionalit a algoritmů, které v této knihovně nejsou zahrnuty. Pro tuto práci je důležitá implementace algoritmu FP-strom.

3.4 Shrnutí

V této kapitole byl představen Python jako všestranný a široce oblíbený programovací jazyk v prostředí datové analýzy. Popularita Pythonu v této doméně je dána jeho jednoduchostí, čitelností a širokou nabídkou knihoven vytvořených na míru konkrétním úkolům datové analýzy.

Následně byla prozkoumána aplikace Jupyter Notebook, interaktivní prostředí umožňující přehledné a interaktivní tvoření programů s možností vizualizace a vkládání vysvětlujícího textu.

V poslední části kapitoly byly diskutovány důležité knihovny Pythonu z hlediska datové analýzy a strojového učení. Knihovny NumPy a Pandas pro základní a pokročilou manipulaci s daty, knihovny Matplotlib a Seaborn pro jednodušší a rozšířenou tvorbu grafů a vizualizací dat, a nakonec scikit-learn, knihovna představující základní stavební kámen pro strojové učení v Pythonu.

Dohromady tyto aplikace a knihovny spolu s Pythonem samotným tvoří silné prostředí, ve kterém může být implementace dolovacího projektu řešena od explorativní analýzy až po trénování a nasazení hotových modelů.

Kapitola 4

Implementace řešení

V této kapitole bude popsána implementace řešení. V rámci této práce byly vypracovány dvě dolovací úlohy, a to predikce hodnot atributu a dolování asociačních pravidel. Budou popsány dostupné tabulky, z kterých byly dolovány znalosti, následně bude popsán proces, kterým byla data připravena a vyčištěna.

Vypracování této práce bylo možné díky datům, které byly autorovi práce poskytnuty pojišťovnou, která si nepřála být v rámci zachování obchodního tajemství jmenována. Oba dolovací úkoly mají tedy návaznost na pojišťovací prostředí, jehož kontext bude v rámci této kapitoly představen a vysvětlen spolu s kroky, které byly provedeny.

4.1 Kontext dat

Dolovací projekt podle modelu CRISP-DM začíná pochopením kontextu z hlediska businessu. I v této práci byl první krok pochopení business problému, konkrétně dvou problematik, které jsou v pojišťovacím prostředí přítomny.

První problematikou jsou tzv. likvidace. Dojde-li k pojistné události, pojistitel (v tomto případě pojišťovna) je povinnen podle zákona č.89/2012 sb.[1] poskytnout jednorázové či opakované pojistné plnění v ujednaném rozsahu. Nejdříve ale musí pojistnou událost posoudit likvidátor, který prověří nárok na pojistné plnění a vyčíslí škodu. Na výkonu konkrétního likvidátora tedy do určité míry závisí, kolik peněz musí pojišťovna v rámci pojistného plnění vyplácet. Z pohledu pojišťovny je proto důležité svěřovat likvidaci pojistných událostí zkušeným likvidátorům a sledovat atributy plnění pojistných událostí, například celková délka procesu likvidace, vyplacené pojistné plnění a jiné.

Druhou problematikou je tzv. propojištěnost. Jedná se o zcela business problém, kdy je hlavním cílem navýšení prodejů pojistných produktů. Toho lze dosáhnout tím, že zákazníci pojištění v jedné oblasti pojištění jsou kontaktování, aby uzavřeli pojištění v dalších oblastech. K tomu jsou motivováni například slevou za propojištěnost. Z pohledu pojišťovny je tedy opět důležité mít znalost, jaká je pravděpodobnost, že zákazník uzavře pojištění z konkrétní oblasti, pokud byl již dříve pojištěný v oblasti jiné.

4.2 Pochopení dat

Pro tvorbu obou datových sad byla použita produkční databáze pojišťovny. Jedná se o klasickou relační databázi, kde jsou navzájem tabulky odkazovány cizími klíči. Seznam

[1] Zde konkrétně paragraf §2821

atributů v tabulkách není kompletní, neboť jsou všechny velmi rozsáhlé. Klíčové atributy jsou podrobně vysvětleny v následující kapitole.

První datová sada popisuje pojistné události. Byla použita k zodpovězení bussiness otázek:

- Dokážeme v momentě, kdy došlo k pojistné události, predikovat přibližnou výši pojistného plnění?
- Dokážeme predikovat délku trvání vyřízení likvidace?

Datová sada byla složena z následujících tabulek[2]:

- `hlaseni_pojistnych_udalosti`. Záznam v této tabulce vzniká bezprostředně po vzniku pojistné události. Obsahuje informace známé v době nahlášení pojistné události, ale také obsahuje atributy, které jsou k záznamu později doplněny, jako například datum likvidace. Protože je tabulka poměrně rozsáhlá, jsou zde uvedeny ilustračně některé atributy. Kromě očekávatelných atributů popisujících pojistnou událost (datum pojistné události, místo pojistné události, příčina, cizí klíče do jiných tabulek, apod.) je zde také uveden likvidátor, jeho organizační jednotka, informace o pojištění (zda může vzniknout nárok na rentu, zda může proběhnout automatická likvidace).
- `druh_pojistneho_plneni`. Jedná se o tabulku ve které jsou záznamy o druzích konkrétního pojistného plnění. Jsou zde atributy popisující, zda je pojistné plnění daněno, jakým způsobem má být zúčtováno, zda je nutná kontrola lékaře nebo existence diagnózy. Obsahuje i další informace odbornějšího rázu, například příznak svodní události.
- `ciselnik_pricin`. Tato číselníková tabulka obsahuje informace k jednotlivým příčinám, jmenovitě kód příčiny, název příčiny, název příčiny pro tisk apod.
- `likvidace`. V této tabulce lze nalézt záznamy o probíhající i dokončené likvidaci. Protože je tato tabulka součástí provozní databáze, je její obsah průběžně aktualizován, proto obsahuje kromě předem známých informací (hodnota vozidla, náklady na opravu, informaci o viníkovi nehody, zda je škoda totální, částka pojištění) také informace neznámé ve chvíli hlášení pojistné události. Takové informace byly v rámci tvorby datové sady odstraněny, aby výsledky nebyly zkresleny.

Další dvě datové sady byly vytvořeny pro problematiku propojištěnosti. Klíčová otázka, na kterou je s pomocí této datové sady hledána odpověď, zní: „Pokud víme, že zákazník má zájem o produkt A, jaký produkt B mu máme doporučit, aby byla co nejvyšší pravděpodobnost, že si zákazník tento produkt také koupí? A o který produkt nejspíše nebude mít zájem?“.

Datová sada vznikla složením následujících tabulek:

- `partneri`. Tato tabulka v sobě zahrnuje informace o všech klientech pojišťovny, včetně jejich osobních údajů. Z této tabulky byl pro analýzu použit pouze její primární klíč, s jehož pomocí byly později hledány smlouvy, které klient uzavřel.

[2] Jména tabulek jsou pro účely práce a zachování anonymity změněna. Z tabulek byly odstraněny všechny osobní údaje.

- `smlouvy`. V této tabulce jsou uvedeny všechny aktivní i již neaktivní smlouvy, které kdy byly s pojišťovnou uzavřeny. Kromě prostého faktu, že smlouva existuje, a jejího cizího klíče do číselníkové tabulky typů pojistných produktů nebyla z této tabulky využita žádná informace.
- `ciselnik_pojistnych_produktu`. V této tabulce nalezneme obecné informace o konkrétních pojistných produktech, tedy typech pojištění. Z této tabulky byly použity informace o obecném druhu pojištění a konkrétním pojistném produktu, na jejichž základě byly následně vytvořeny dvě datové sady.

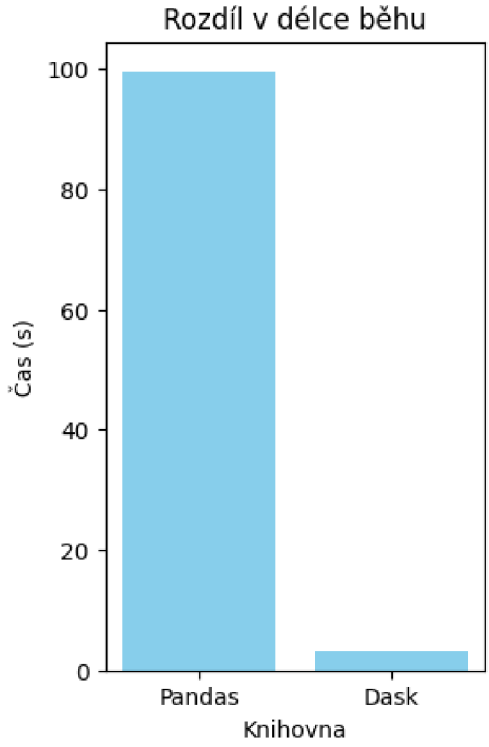
Jedna datová sada se zaměřuje pouze na obecné typy produktů (například cestovní pojištění, podnikatelské pojištění, apod.), následně je tato datová sada zmiňována jako datová sada s nižší granularitou. Druhá datová sada s vyšší granularitou se zaměřuje na konkrétní pojistné produkty, které byly zákazníky využívány. Jedná se tedy o konkrétní cestovní pojištění nebo konkrétní povinné ručení, přičemž u každé z těchto i jiných skupin má pojišťovna širokou nabídku produktů. Tato datová sada je proto oproti té s nižší granularitou řidší.

4.3 Příprava dat

Dalším krokem je podle CRISP-DM příprava dat, tedy krok, při kterém jsou data z původního formátu, obecně pro dolování jen stěží použitelného, přetransformována a připravena pro konkrétní dolovací úlohy, které jsou předem známé. Obě datové sady byly proto z podkladních dat vytvořeny na míru pro klasifikaci a dolování asociačních pravidel.

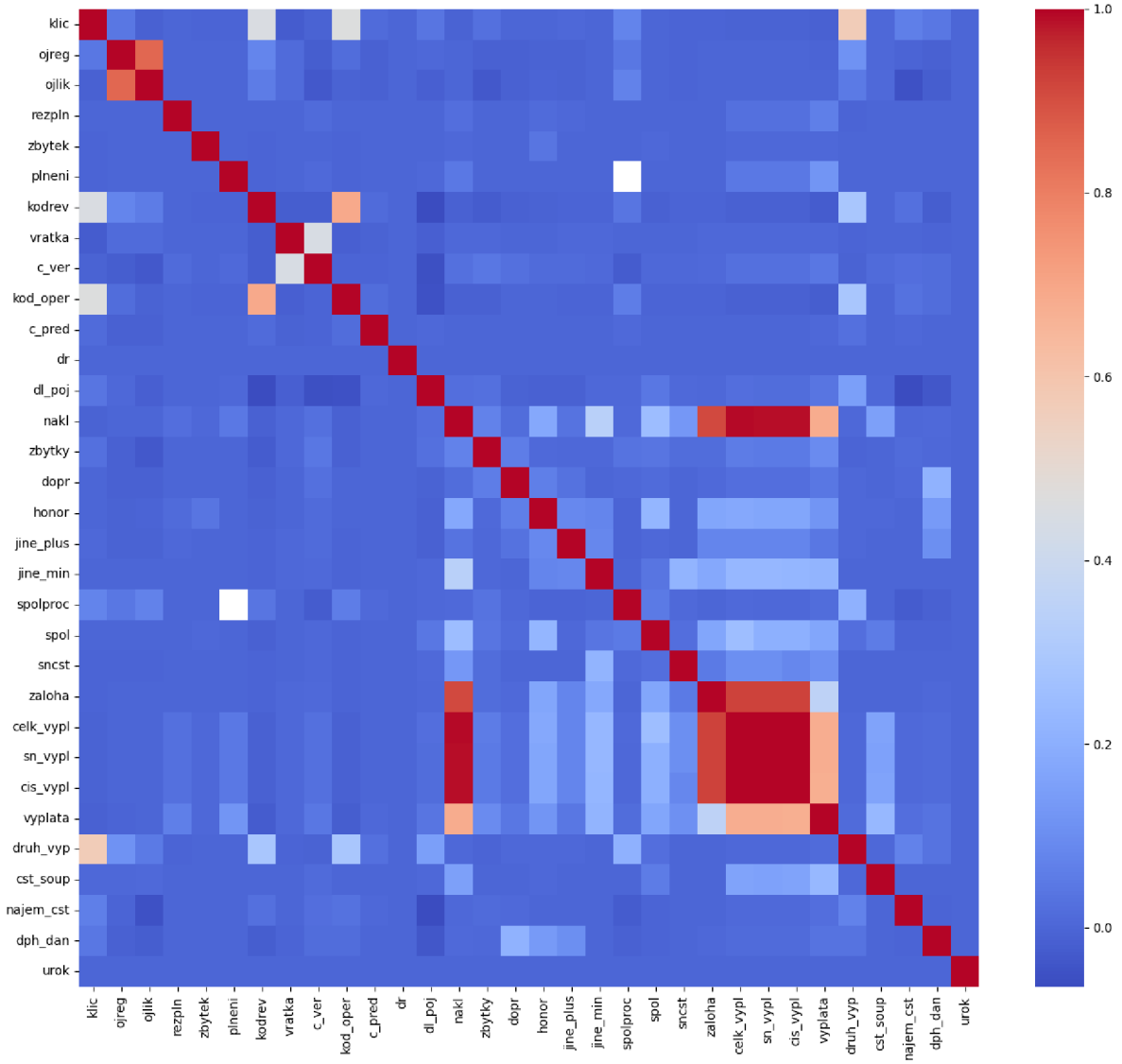
4.3.1 Datová sada likvidací

První nečekanou překážkou při vypracování tohoto podúkolu byla již samotná velikost datové sady popisující pojistné události. Protože je tato datová sada rozsáhlá co do počtu sloupců i záznamů, ukázalo se, že pro její zpracování nestačí použití knihovny `Pandas`, jako je běžné u datové analýzy v Pythonu. Místo této knihovny byla proto použita knihovna `Dask`, která umožňuje paralelizaci a tím i značné zrychlení výpočtu. Rychlost načtení datové sady příkazem pro čtení souboru formátu `csv` (`pandas.read_csv()`, respektive `dask.read_csv()`), který má 6,5 milionu záznamů, je vidět na obrázku 4.1. Zatímco Dask zvládl soubor načíst průměrně za 3 vteřiny, Pandas ten samý soubor načítal 1 minutu a 39 vteřin.



Obrázek 4.1: Načtení analyzovaného souboru knihovnami Dask a Pandas. Číslo bylo spočítáno z průměrné délky trvání při 10 spuštěních.

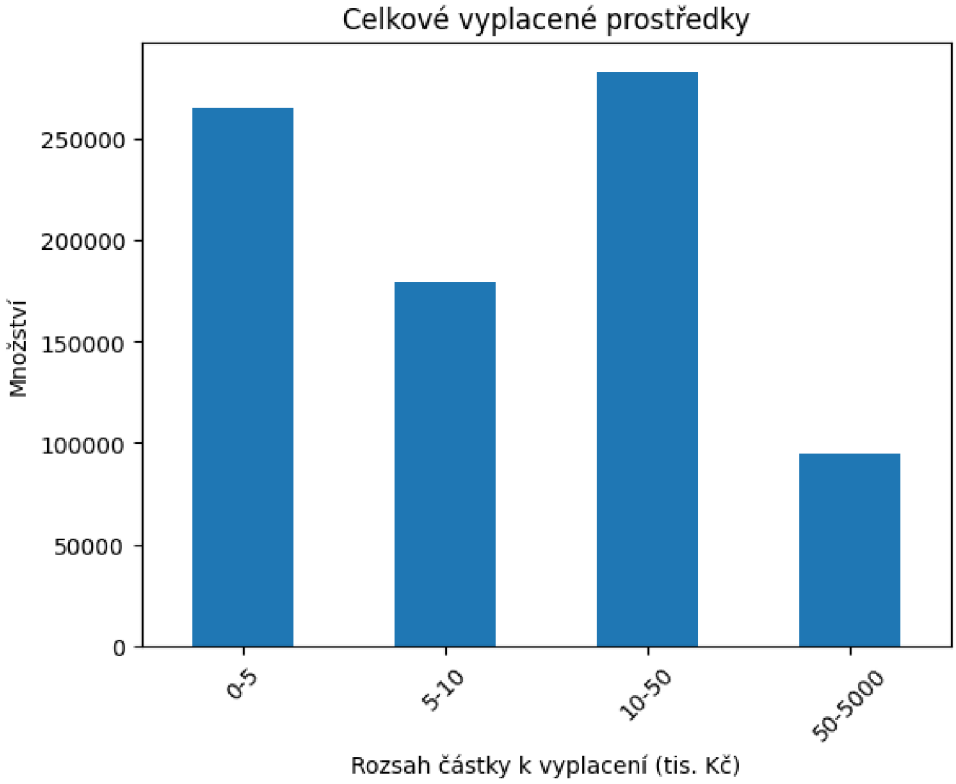
Po načtení souboru byla provedena korelační analýza numerických atributů a její vizualizace pomocí metod `dataframe.corr()` `seaborn.heatmap()` (viz obrázek 4.2). Na grafu lze vidět některé silně korelované atributy, které byly proto odstraněny. Zároveň s korelovanými atributy byly odstraněny rovněž atributy bez informačního přínosu. Mezi těmito atributy byly například datumy, nadbytečné informace popisující organizační strukturu zaměstnanců pojišťovny a jiné atributy, jejichž význam je z důvodů nedostatečné dokumentace zdrojových tabulek neznámý.



Obrázek 4.2: Korelační matice numerických atributů datové sady pojistných událostí.

Následné pokusy o zpracování ukázaly, že v záznamech datové sady je poměrně rozsáhlý šum a mnoho záznamů nemá známé hodnoty podstatné části atributů. Kvůli velkému množství podkladních dat se jako nejlepší způsob řešení tohoto problému jevilo smazání všech takových záznamů. Tento krok výrazně přispěl v rychlosti zpracování datové sady, protože její velikost se tímto snížila z původních 8,5 GB čistých dat na 3,3 GB. Počet záznamů po tomto kroku stále dosahoval 6,5 milionu záznamů výrazně vyšší kvality než původně, což bylo vyhodnoceno jako dostatečný počet pro trénování klasifikačních modelů.

Pro přípravu atributů ke klasifikaci bylo ještě nutné převedení kategorických atributů na numerické, protože knihovní metody klasifikátorů z knihovny `scikit-learn` takové atributy vyžadují. Pro tento převod byla využita metoda z knihovny `scikit-learn.preprocessing` jménem `LabelEncoder()`. Tato metoda nahradí všechny hodnoty kategorického atributu kladnou nenulovou celočíselnou hodnotou. Speciální hodnotou byly nahrazeny také jednotky chybějících hodnot, které v datové sadě zůstaly i přes předchozí kroky.

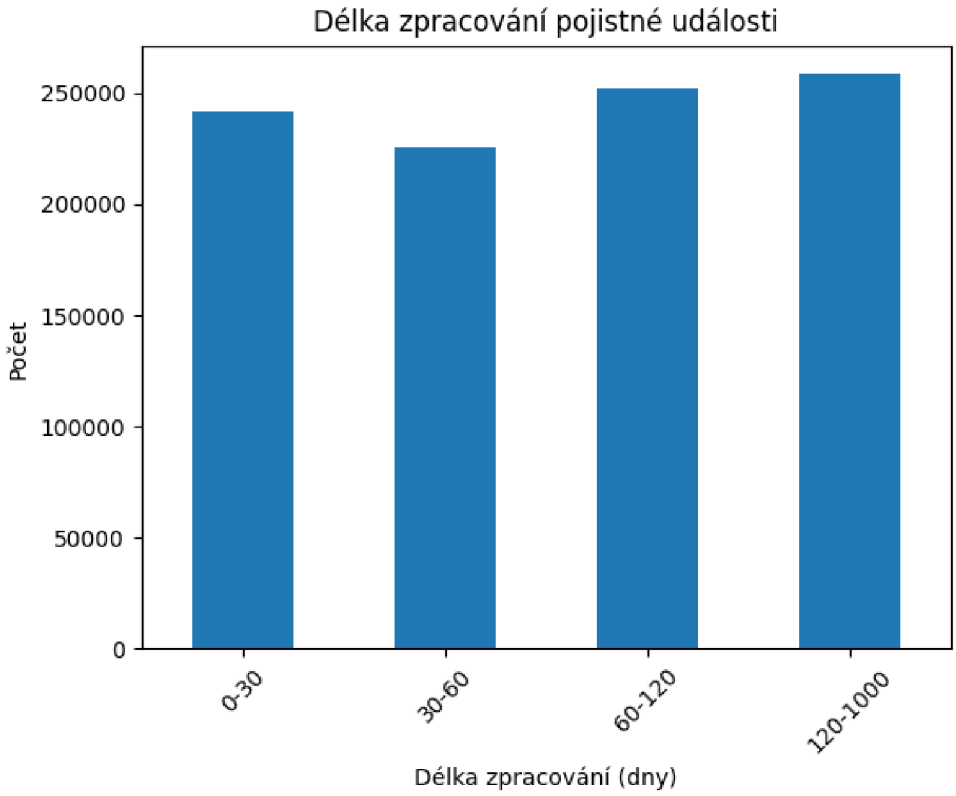


Obrázek 4.3: Počet vyplacených prostředků v tis. Kč.

Následujícím krokem bylo rozdělení záznamů do tříd, protože v podkladních tabulkách datové sady se nenachází žádný atribut, který by tuto roli plnil. Rozsahy tříd byly voleny experimentálně s přihlédnutím k významům jednotlivých hranic a podobným velikostem tříd.

U atributu popisující celkové vyplacené prostředky (viz obrázek 4.3) byly záznamy rozděleny do čtyř tříd. Hranice byly stanoveny v rozmezí do pěti tisíc korun, do deseti tisíc korun, do 50-ti tisíc korun a nad padesát tisíc korun.

Při rozdělování záznamů do tříd podle délky zpracování pojistné události bylo nutné brát v úvahu hranici 30-ti dnů, do které má pojišťovna povinnost vyplatit pojistné plnění. Při pohledu na graf 4.4 si lze povšimnout, že do této lhůty je zlikvidována jen část pojistných událostí. Další hranice tříd byly stanoveny do dvou standardních měsíců, poté do čtyř měsíců, a nad čtyři měsíce.



Obrázek 4.4: Délka trvání vyřízení pojistné události ve dnech.

Posledním krokem před trénováním klasifikátorů bylo nadvzorkování a podvzorkování záznamů. To bylo provedeno pomocí knihovny `imbalanced-learn` a jejích metod, konkrétně `RandomUndersampler` a `RandomOversampler`.

4.3.2 Datová sada druhů pojištění

U datových sad druhů pojištění nebylo nutné řešit problémy týkající se kvality dat. Práce nutná k přípravě této datové sady spočívala kompletně na jejich transformaci. Bylo totiž nutné z relačních tabulek, které uchovávají informace o nákupu pojištění (tedy transakci), vytvořit tabulku transakční, tedy data úplně přetransformovat.

K této transformaci byly využity prostředky jazyka SQL. Výsledná podoba obou tabulek napodobuje transakční databázi. Tabulka nižší granularity se skládá z klíče a n booleovských hodnot pro každý druh pojistného produktu, tabulka vyšší granularity obsahuje také klíč a rovněž booleovské hodnoty, tentokráte ovšem pro každý konkrétní pojistný produkt.

Z hlediska dalších druhů úpravy dat nebylo nutné použít žádné čisticí techniky, protože datové sady neobsahují žádnou redundanci, pouze seznam všech relevantních zákazníků. Ten není sám o sobě zašuměný ani neobsahuje nekvalitní informace.

4.4 Shrnutí

V této kapitole byly shrnuty kroky nutné k přípravě dolovacího projektu, z hlediska CRISP-DM se jedná o kroky pochopení kontextu, pochopení dat a přípravy dat. Byl vysvětlen

kontext obou dolovacích úloh a důvod, proč by v praxi byly úlohy prováděny. Byly také představeny klíčové otázky, na které má dolovací projekt hledat odpovědi. Byly popsány zdrojové tabulky, z kterých vznikly obě podkladní datové sady. Na závěr byl popsán proces přípravy dat, byly uvedeny konkrétní využité knihovny a klíčové metody, díky kterým byla příprava dat provedena. Byly popsány konkrétní překážky, které při řešení dolovacího projektu nastaly, a také způsob, jakým byly tyto překážky vyřešeny.

Kapitola 5

Experimenty

V této kapitole budou provedeny experimenty nad datovou sadou likvidací, natrénované klasifikátory a jejich výsledky. Následně budou představena a diskutována vydolovaná asociační pravidla.

5.1 Predikce výše pojistného plnění

Trénování probíhalo nad vzorkem jednoho milionu záznamů, z něhož sto tisíc bylo použito jako testovací podmnožina.

Na datové sadě likvidací bylo pro predikci vyplaceného pojistného plnění natrénováno celkem 12 klasifikátorů. Jako základní klasifikátor pro porovnání s ostatními byl vybrán algoritmus rozhodovacího stromu. Dalšími vyzkoušenými algoritmy byly náhodný les, MLP a naivní Bayesovská klasifikace. Každý druh klasifikátoru byl trénován nad základní datovou sadou a následně nad podvzorkovanou a nadvzorkovanou datovou sadou. Přesnosti klasifikátorů jsou zaneseny v tabulce 5.1.

Model	Základní	Podvzorkovaný	Nadvzorkovaný
Rozhodovací strom	0.618	0.562	0.579
Náhodný les	0.634	0.566	0.582
MLP	0.616	0.555	0.555
Naive Bayes	0.392	0.403	0.392

Tabulka 5.1: Přesnost klasifikátorů při predikci vyplaceného pojistného plnění.

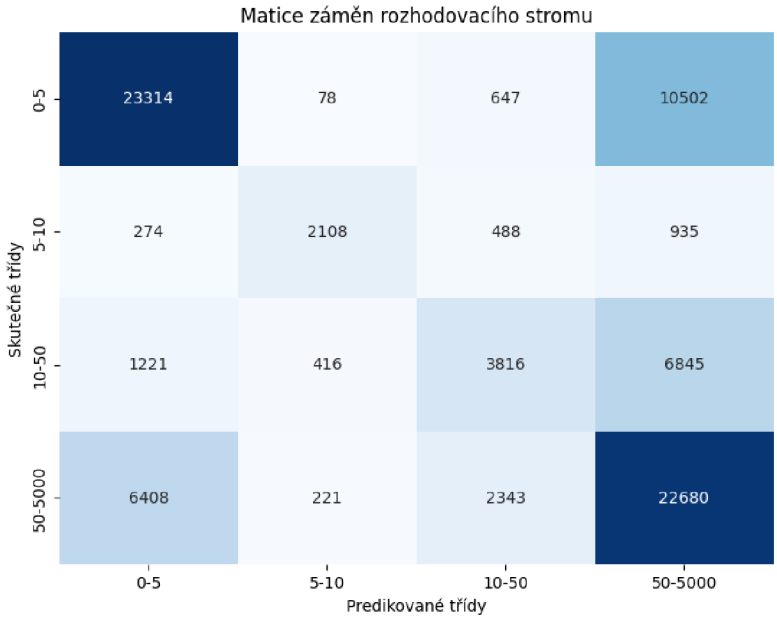
Na hodnotě přesnosti si můžeme povšimnout několika důležitých skutečností:

- Překvapivě nízké úspěšnosti některých klasifikátorů. V porovnání s rozhodovacím stromem přesností predikce překvapivě podléhají jak klasifikátor MLP, tak naivní Bayesovská klasifikace.
- Podvzorkování ani nadvzorkování nezvyšuje úspěšnost klasifikátorů. Jedinou výjimkou je naivní Bayesovská klasifikace, která, pokud je natrénována na podvzorkované datové sadě, má vyšší úspěšnost než pokud je natrénována na základní datové sadě.

Vzhledem ke komplikovanosti dat je úspěšnost 63,4 % výsledek, který má smysl dále analyzovat. Pro lepší pochopení výsledků klasifikace se nyní podívejme na matici záměn pro

dva nejúspěšnější klasifikátory - rozhodovací strom a náhodný les, natrénované na základní datové sadě (viz obrázek 5.1).

Rozhodovací strom nejlépe rozpozná třídu záznamů s nejnižší a nejvyšší vyplacenou částkou. U těchto tříd dosahuje 67,5 %, respektive 71,7 % přesnosti. Naopak třídu s druhou nejvyšší vyplacenou částkou klasifikuje poměrně špatně. Správně u této třídy zařadí rozhodovací strom pouze 31,0 % záznamů.



Obrázek 5.1: Matice záměn rozhodovacího stromu natrénovaného na základní datové sadě.

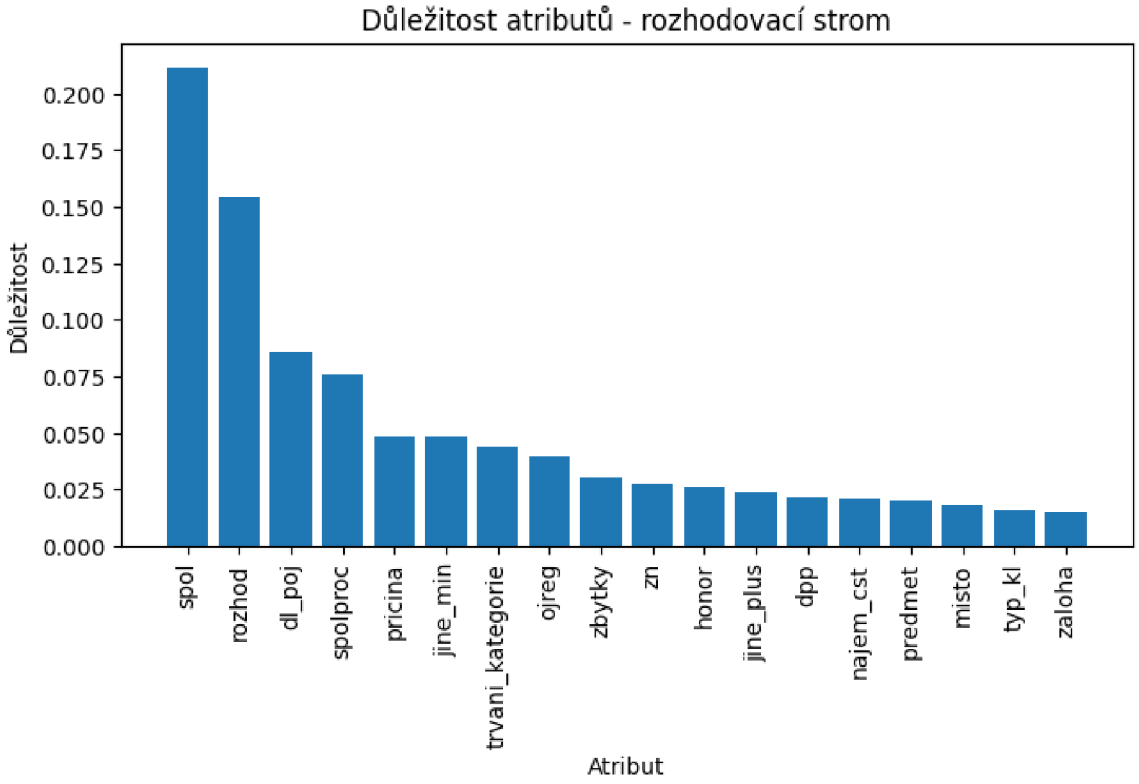
Na obrázku 5.2 můžeme vidět, jaká váha byla přiřazena jednotlivým atributům při rozhodování. Nejdůležitějším atributem byl pro rozhodovací strom částka spoluúčasti (`spol`), na kterou je zákazník pojištěný (s tím souvisí i čtvrtý nejdůležitější atribut `spolproc`, který vyjadřuje tu samou skutečnost, ale v procentech). To není překvapivá informace, protože odvozovat vyplacenou částku od částky, na kterou je zákazník pojištěný, je standardní postup. Dalším důležitým atributem je kategorický atribut určující, zda byla škoda zaviněná, nezaviněná, nebo zaviněná s alkoholem (`rozhod`).

Třetím nejdůležitějším atributem je příznak kontroly dlužného pojistného(`dl_poj`). Tato informace nám říká, že u pojištění může docházet k zvyšování pojistného, pokud není placeno včas. Jedná se tedy o informaci o druhu pojištění obecně, ne o konkrétním pojištění, na jehož základě vzniká nárok na pojistné plnění. Fakt, že tato informace může hrát roli při rozhodování o výši vyplaceného pojistného plnění, lze vysvětlit souvislostí mezi určitým druhem pojištění a obvyklou částkou vyplaceného pojistného plnění. Například u havarijního pojištění je tato informace kontrolována, kdežto u pojištění odpovědnosti není. Rozdíl pojistného plnění mezi těmito dvěma druhy pojištění může být ovšem značný.

Dalšími důležitými atributy jsou:

- příčina pojistné události (`pricina`),
- jiné výdaje pojišťovny (`jine_min`),
- délka zpracování pojistné události (`trvani_kategorie`),

- organizační jednotka, tedy oblast, ve které se pojistná událost stala (`ojreg`),
- hodnota zbytků vozidla (`zbytky`) a
- název pojistného produktu (`zn`).

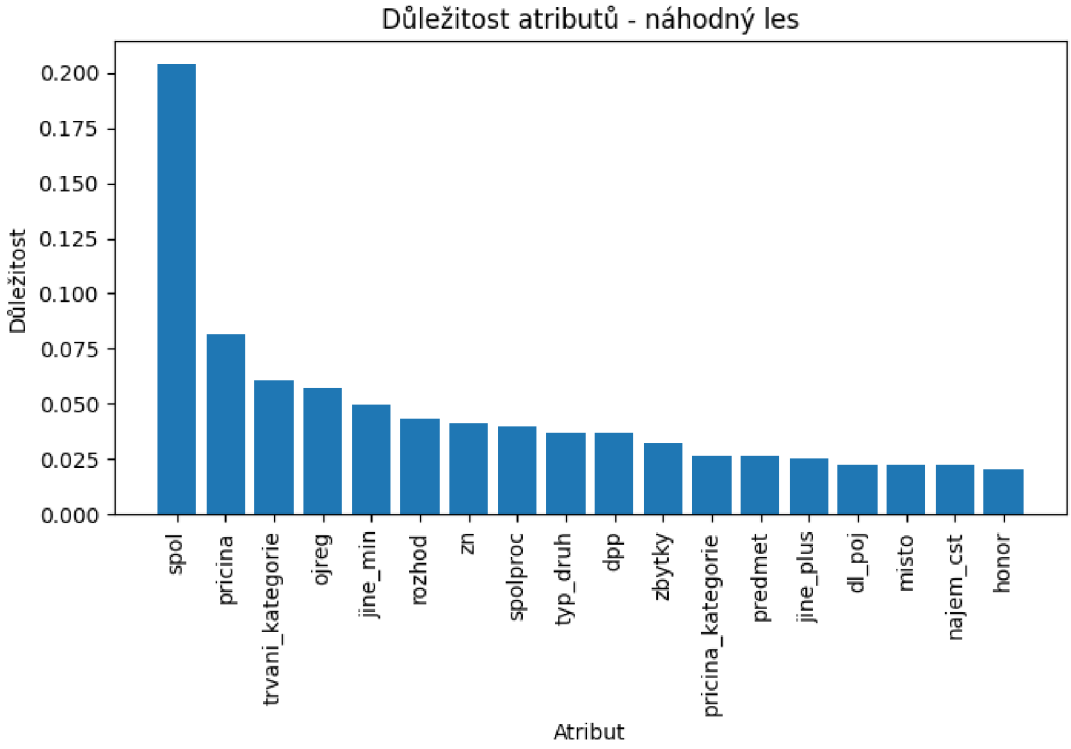


Obrázek 5.2: Důležitost atributů rozhodovacího stromu.

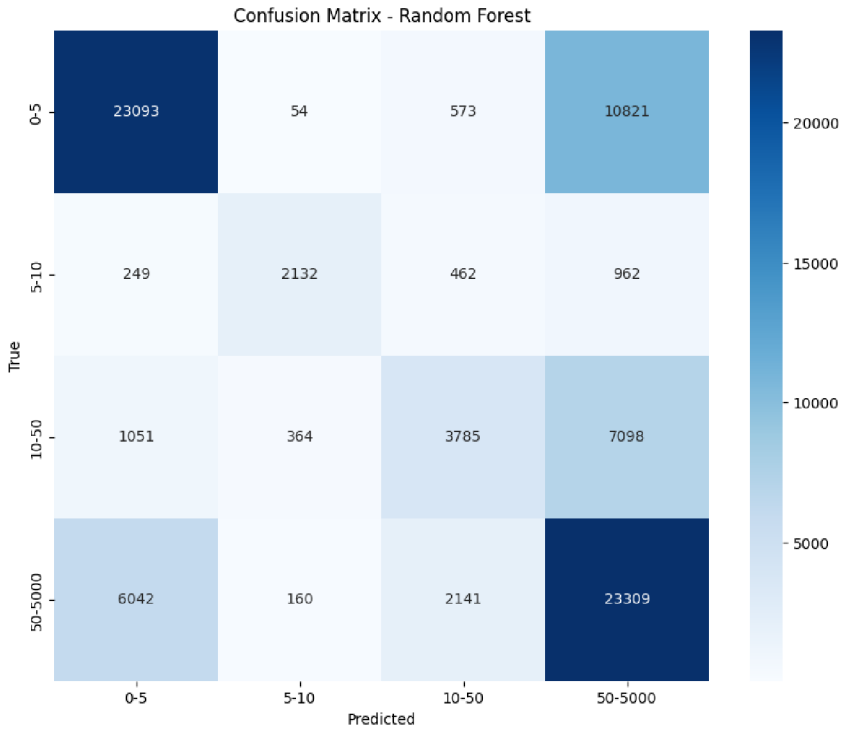
Velmi podobné výsledky vidíme i při analýze natrénovaného náhodného lesa (viz obrázky 5.3 a 5.4). Jako významné byly vyhodnoceny stejné atributy, pouze v jiném pořadí. Náhodný les se tedy při rozhodování shodně opíral o částku pojištěné spoluúčasti (`spol`), následně o příčinu (která byla pro rozhodovací strom až na pátém místě, `pricina`), poté o délku trvání (ta byla u rozhodovacího stromu na sedmém místě, `trvani_kategorie`).

Větší váhu přikládá náhodný les organizační jednotce (`ojreg`). Vzhledem k nerovnoměrnému rozložení bohatství napříč Českem [13] je očekávatelné, že v různých částech země se bude průměrná výše vyplaceného pojistného plnění lišit.

Pozoruhodný je rozdíl významnosti atributu kontroly dlužného pojistného (`dl_poj`), který je pro rozhodovací strom na třetím místě, zatímco náhodný les mu tak velký význam nepřikládá. Povšimněme si také, že kromě názvu pojistného plnění (`zn`) je výše než `dl_poj` také atribut druhu pojistného plnění (`typ_druh`). Očividně tak došlo k situaci, kdy byl rozhodovacím stromem vybrán atribut s nejvyšším informačním přínosem, který plnil stejnou roli, jako v případě náhodného lesa plnily čitelnější atributy názvu a druhu pojistného plnění.



Obrázek 5.3: Důležitost atributů náhodného lesa.



Obrázek 5.4: Matice záměn náhodného lesa natrénovaného na základní datové sadě.

5.2 Predikce délky trvání likvidace

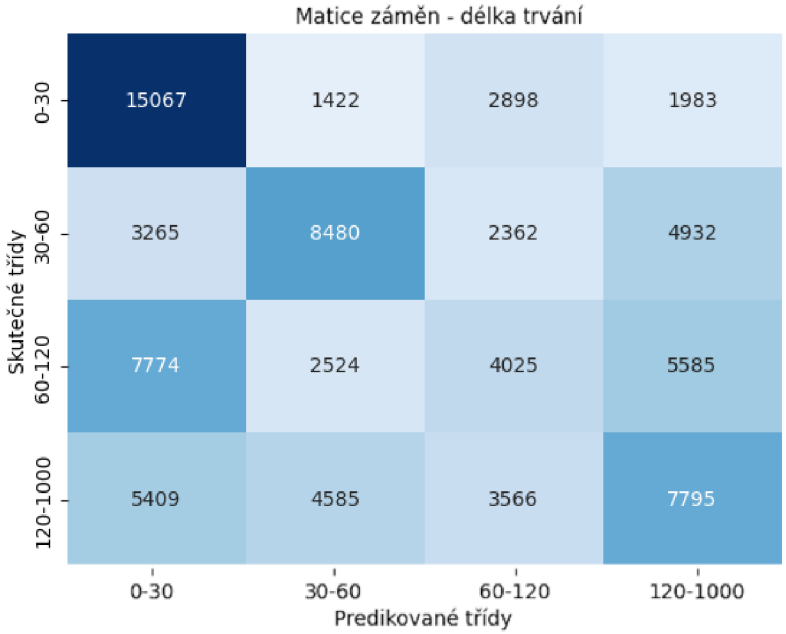
Pro predikci délky trvání likvidace bylo natrénováno 9 klasifikátorů. Nadvzorkování ani podvzorkování se opět neukázaly jako účinné, nejúspěšnějším klasifikátorem z hlediska přesnosti predikce byl náhodný les s přesností 43 %, následován rozhodovacím stromem s přesností 42,7 % (viz tabulka 5.2).

Model	Základní	Podvzorkovaný	Nadvzorkovaný
Rozhodovací strom	0.427	0.286	0.386
Náhodný les	0.430	0.286	0.389
MLP	0.388	0.294	0.337

Tabulka 5.2: Přesnost predikce délky trvání likvidace

Na matici záměn (obrázek 5.5) vidíme, že se klasifikátor nenaučil většinu tříd klasifikovat s vysokou přesností. Výjimkou jsou v tomto ohledu záznamy zlikvidované do jednoho měsíce. Ty rozpoznal klasifikátor v 70,5 % případů. Velmi špatně poté klasifikátor predikoval třídu záznamů zlikvidované v rozmezí 60-ti až 120-ti dnů. U nich klasifikátor určil pouze 20,2 % správně, tedy o necelých 5 % méně, než kdybychom nechali délku trvání předvídat čtyřstěnnou kostku.

Celkově je klasifikátor úspěšnější než náhodný odhad, proto má smysl analyzovat, jaká pravidla a vzory se naučil.



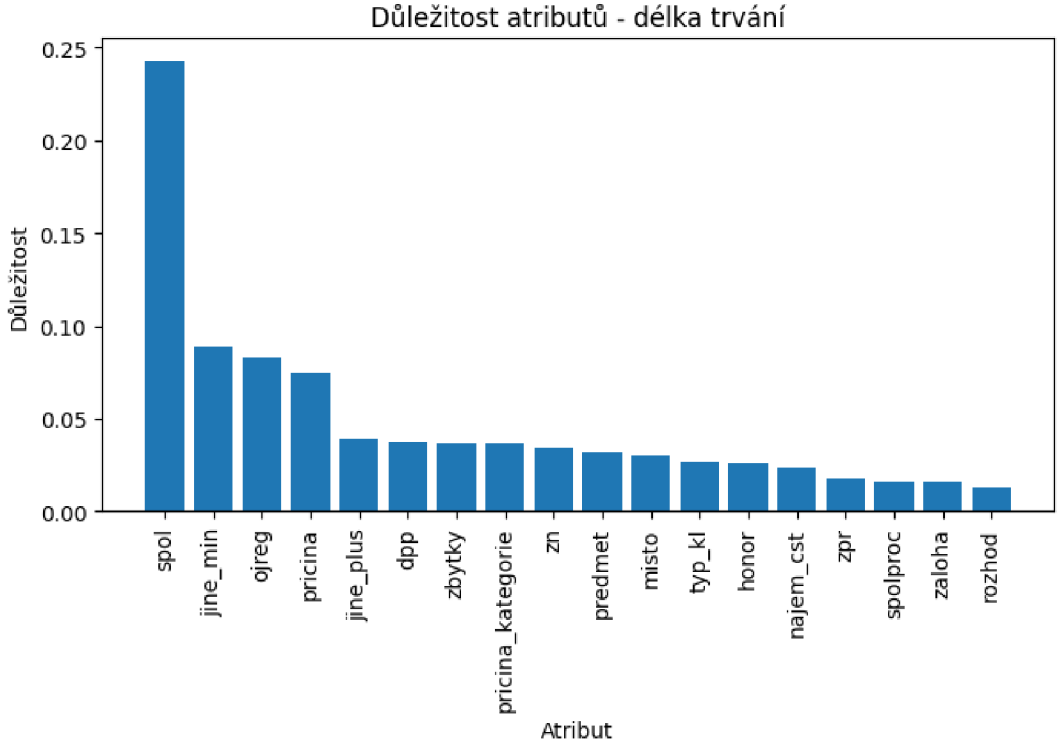
Obrázek 5.5: Matice záměn náhodného lesa při predikci trvání likvidace.

Pro interpretaci náhodného lesa opět použijeme graf důležitosti atributů (viz obrázek 5.6). Při pohledu na graf si můžeme povšimnout, že důležité atributy jsou relativně podobné s důležitými atributy pro predikci výše pojistného plnění. Rozdíl je atribut určující zavinění pojistné události (`rozhod`), který pro predikci trvání likvidace není téměř vůbec důležitý. Důležitým je opět atribut pojištění spoluúčasti (`spol`), což není překvapivá informace. Při

likvidacích velkých částek je z hlediska pojišťovny nepochybně nutné postupovat důsledně, zatímco u menších pojistných událostí není výjimkou tzv. automatická likvidace, která je likvidátorem pouze zkontrolována a potvrzena.

Dalšími důležitými atributy jsou atributy jiných výdajů a příjmů pojišťovny (`jine_min`, `jine_plus`). Tato informace znovu potvrzuje, že délku likvidace doopravdy nejvíce ovlivňuje výsledná částka, která může být v rámci pojistného plnění vyplacena.

Poslední z pětice nejdůležitějších atributů jsou atributy organizační jednotky (`ojreg`) a příčiny pojistné události (`pricina`). Toto nám přináší informaci o vztahu náročnosti likvidace a příčiny, nebo také části Česka, kde došlo k pojistné události. To může být z hlediska pojišťovny zajímavá informace, protože s její pomocí může určit druhy příčin, které mohou být složitější k likvidaci, a přidělovat je zkušenějším likvidátorům. Dalším přínosem této informace může být následná hloubková analýza procesu likvidace v organizačních jednotkách a zefektivnění procesů v těch méně výkonných.



Obrázek 5.6: Důležitost atributů náhodného lesa při predikci trvání likvidace.

5.3 Dolování asociačních pravidel nižší granularity

Pro dolování asociačních pravidel byly použity dvě datové sady různé granularity. Protože pojišťovna, z jejíchž dat byla asociační pravidla dolována, poskytuje mnoho druhů pojištění a má mnoho zákazníků, bylo nutné náležitě nastavovat minimální hodnoty metrik, podle nichž měla být pravidla dolována.

Na datové sadě nižší granularity byl spuštěn algoritmus FP-stromu, z něhož byly následně dolovány asociační pravidla. Frekventované množiny, které zjistil FP-strom, vidíme v tabulce 5.3.

Skupina produktů	Podpora
POV	0.473216
LEC	0.337280
REZ	0.281293
MAJ	0.243042
MAJ, POV	0.113951
LEC, POV	0.104450
POD	0.104108
POV, REZ	0.102501
LEC, REZ	0.083664
HAV	0.082428

Tabulka 5.3: Frekventované množiny vygenerované z datové sady nižší granularity.

Skupina produktů `POV` v sobě zahrnuje povinné ručení. Vidíme tedy, že se jedná o nejčastější skupinu produktů zakoupenou zákazníky, ze všech zákazníku mělo nějakou formu povinného ručení zakoupeno 47,3 % z nich.

Další významnou skupinou produktů je `LEC`, která je tvořena produkty cestovního pojištění. Tato skupina dosahuje podpory 28,1 %. Skupina `REZ` obsahuje všechny druhy investičního pojištění, tedy například důchodové nebo kapitálové životní pojištění. Ve skupině `MAJ` jsou obsaženy různé druhy majetkového pojištění.

Z hlediska oblíbených kombinací skupiny produktů se nejčastěji vyskytuje kombinace majetkového pojištění a povinného ručení, následováno kombinací cestovního pojištění a povinného ručení. Pro úplnost zde ještě uveďme, že `POD` značí pojištění pro podnikatele a `HAV` jsou havarijní pojištění.

V tabulce 5.4 se můžeme podívat na tři pravidla s nejvyšší hodnotou spolehlivosti (confidence). Poměrně nepřekvapivě všechny tři říkají, že pokud má zákazník u pojišťovny sjednané nějaké pojištění mimo povinné ručení, měl by si ho také sjednat.

Antecedent	Konsekvent	Popdpora	Spolehlivost	Lift	Leverage	Conviction
HAV	POV	0.073802	0.895350	1.892052	0.034796	5.033745
LEC, MAJ	POV	0.044159	0.596553	1.260634	0.009130	1.305706
MAJ, REZ	POV	0.043890	0.587489	1.241482	0.008537	1.277019

Tabulka 5.4: Asociační pravidla řazená dle podpory.

Nejsilnější spolehlivost má překvapivě pravidlo, jehož levá strana je tvořena havarijním pojištěním, které oproti ostatním skupinám není příliš zastoupené. Vysoká hodnota spolehlivosti nám ale říká, že pokud bychom v databázi měli zákazníka, který má sjednané havarijní pojištění a nemá sjednané povinné ručení, měli bychom ho kontaktovat s nabídkou povinného ručení.

Dalšími poměrně spolehlivými pravidly jsou poté pravidla předpokládající dvě skupiny produktů, a to nejdříve kombinaci cestovní pojištění a majetkové pojištění, následně majetkové pojištění a investiční pojištění. Tato dvě pravidla tedy cílí na finančně zodpovědné zákazníky, kteří mají majetek k pojištění a buď jezdí do zahraničí na dovolenou, nebo si spoří finanční prostředky. Toto je v Česku silně zastoupená skupina, vlastní bydlení má 75 % domácností [12].

Méně silně je zastoupená skupina občanů, kteří podnikají cesty do zahraničí (před koronavirovou pandemií přes 30 %, v roce 2022 26 %) a občanů spořících v rámci investičního

pojištění (16 %) [12][11]. Díky tomuto vyhranění získáváme užší a přesnější popis skupiny zákazníků, na kterou je například možno cílit kampaň propagující nový nebo výhodný produkt povinného ručení.

V tabulce 5.5 můžeme vidět asociační pravidla řazená dle metriky lift. U metriky lift je důležité mít na paměti, že pravidlo hodnotí symetricky, tedy stejná hodnota liftu je pro pravidlo $HAV \implies POV$ a $POV \implies HAV$. Pro přehlednost je v tabulce uvedeno z dvojice pravidel vždy pouze jedno, hodnoceno ale bude z obou stran.

Antecedent	Konsekvent	Popdpora	Spolehlivost	Lift	Leverage	Conviction
HAV	POV	0.073802	0.895350	1.892052	0.034796	5.033745
POV, REZ	MAJ	0.043890	0.428190	1.761796	0.018978	1.323793
LEC, POV	MAJ	0.044159	0.422780	1.739535	0.018774	1.311385
⋮	⋮	⋮	⋮	⋮	⋮	⋮
POV	MAJ	0.113951	0.240800	0.990777	-0.001061	0.997048

Tabulka 5.5: Asociační pravidla řazená dle liftu.

Na prvním místě opět vidíme pravidlo $HAV \implies POV$. Toto pravidlo je se spolehlivostí 89,5 % a přesvědčivostí (conviction) vyšší než 5 skutečně důležité pravidlo[1]. Protože ale z hlediska subjektivních metrik hodnocení asociačních pravidel (popsáno v 2.6.5) nepřináší nečekanou informaci a akčnost tohoto pravidla byla popsána dříve, nebude zde již rozebráno.

Zajímavou hodnotu liftu má pravidlo $POV, REZ \implies MAJ$. Na rozdíl od dříve zmiňovaných pravidel, toto pravidlo nám lépe popisuje chování věrnějších zákazníků pojišťovny, kteří už dříve uzavřeli povinné ručení a investiční pojištění. Vzhledem k dříve uvedenému faktu, že vlastní bydlení má 75 % domácností, nám na základě tohoto pravidla vyvstává nová zajímavá informace o poměrně silné skupině zákazníků. Doporučení majetkového pojištění této skupině může opět vést k prostoru pro potenciálně zajímavé zvýšení příjmů pojišťovny. Ale naopak zákazníci, kteří mají pojištěný majetek, nemusejí mít zájem o investiční pojištění a povinné ručení, protože z obou pravidel toto dosahuje pouze 19,0 % spolehlivosti, oproti 42,8 % spolehlivosti pravidla uvedeného v tabulce.

Dalším zajímavým pravidlem z hlediska liftu je $LEC, POV \implies MAJ$. Toto pravidlo nám opět přináší informaci o chování věrnějších zákazníků, kteří si pojišťují cesty do zahraničí i dopravní prostředek u naší pojišťovny. I na tyto klienty má smysl mířit kampaň o případném novém produktů majetkového pojištění, protože stejně jako předešlá pravidlo dosahuje relativně vysoké spolehlivosti 42,3 %. A stejně jako u předešlého pravidla, i zde by nebylo vhodné uvažovat opačnou implikaci. Zákazník, který má u naší pojišťovny pojištěný pouze majetek, nejspíše nebude uvažovat o pojištění do zahraničí a povinném ručení, protože takové asociační pravidlo má podporu pouze 18,7 % spolehlivosti.

Tato dvě pravidla nám tedy přináší znalost o rozdílném chování zákazníků, kteří již mají povinné ručení a cestovní nebo investiční pojištění, a zákazníků, kteří mají pouze majetkové pojištění. Zákazníci s pouze majetkovým pojištění budou mít obecně nižší zájem o další nabízené produkty, zatímco zákazníci věrní naší pojišťovně spíše dopřejí případné nabídce majetkového pojištění více sluchu.

Jako poslední asociační pravidlo, které je potenciálně zajímavé pro vedení pojišťovny, je na první pohled nenápadné pravidlo $POV \implies MAJ$. Toto pravidlo ve většině metrik nedosahuje takových hodnot, jako všechna výše zmíněná pravidla. Zajímavé je především pro

[1]Pro zajímavost, pravidlo $POV \implies HAV$ má spolehlivost pouze necelých 16 % a přesvědčivost 1,08.

podporu, kterou má v datové sadě skupina produktů `POV`, tedy 47,3 %. Toto pravidlo nám tedy přináší důležitou informaci o druhu pojistného produktu, který nejpravděpodobněji bude přijat zákazníkem, který si přijde sjednat pojištění z oblasti povinného ručení. Získáváme tím důležitý poznatek z hlediska akčnosti, tedy konkrétní krok, který může pojišťovna na základě tohoto dolování přijmout, a to doporučení majetkového pojištění případnému zájemci o povinné ručení.

5.4 Dolování asociačních pravidel vyšší granularity

V tabulce 5.6 vidíme frekventované množiny vydolované z datové sady vyšší granularity[2]. Můžeme si povšimnout, že obecně je podpora mnohem nižší než u tabulky uvedené dříve (viz 5.3). Vzhledem k nízké podpoře víceprvkových frekventovaných množin (spodní část tabulky) se z těchto konkrétních frekventovaných 2-množin nepovedlo vygenerovat jakákoliv asociační pravidla. I kdyby byla případně nějaká vygenerována, možná by jejich interpretace mohla být odborníky na pojišťovnictví považována spíše za spekulaci, proto se jimi nebudeme v rámci této práce zabývat. Vzhledem k více nejasnému označení pojistného produktu na této úrovni budou jednotlivé zkratky vysvětleny spolu s pravidly, které budou tvořit.

Skupina produktů	Podpora
prod_IN8	0.161250
prod_180	0.137621
prod_DZS	0.117322
prod_GA3	0.108509
prod_IN9	0.080765
prod_DLB	0.073007
prod_GAM	0.072214
⋮	⋮
prod_DZS, prod_180	0.041368
prod_IN8, prod_IN9	0.034369
prod_IN8, prod_DZS	0.032552

Tabulka 5.6: Frekventované množiny vygenerované z datové sady vyšší granularity.

Přestože frekventované množiny dosahují poměrně nízké spolehlivosti a celkově byla podkladní datová sada vyšší granularity řidší než předchozí datová sada. Přesto vygenerovaná asociační pravidla (zanesaná v tabulce 5.7) dosahují vyšších hodnot liftu. Hodnoty spolehlivosti také musíme brát vzhledem k velikosti datové sady s větší tolerancí.

Z interpretace výše uvedených asociačních pravidel se spíše nedá zjistit zajímavá informace o chování zákazníků na vyšší úrovni. Všechna pravidla totiž z hlediska obecných typů produktů, jako jsme uvažovali v předchozí části, míří z konkrétního druhu na ten samý druh pojištění, tedy například produkt `prod_GAM` i `prod_GA3` jsou cestovní pojištění druhu `LEC`.

Obecně tedy můžeme říci, že pokud zákazník v minulosti využil jedno z výše uvedených pojištění, je velmi pravděpodobné, že má smysl nabídnout mu nový, výhodnější nebo jinak zajímavější produkt z té samé oblasti. To už může být opět zajímavá informace, z hlediska akčnosti bychom tak mohli doporučit například sezónní kampaň na propagaci nového ces-

[2]Zkratky pojištění v této části byly pozměněny z důvodu zachování anonymity pojišťovny.

Antecedent	Konsekvent	Popdpora	Spolehlivost	Lift	Leverage	Conviction
prod_GAM	prod_GA3	0.072214	0.382524	3.525272	0.019788	1.443766
prod_GA3	prod_GAM	0.108509	0.254575	3.525272	0.019788	1.244639
prod_DZS	prod_DLB	0.117322	0.226192	3.098215	0.017972	1.197962
prod_DLB	prod_DZS	0.073007	0.363488	3.098215	0.017972	1.386743
prod_IN8	prod_IN9	0.161250	0.213138	2.638992	0.021345	1.168229
prod_IN9	prod_IN8	0.080765	0.425538	2.638992	0.021345	1.460062
prod_DZS	prod_180	0.117322	0.352600	2.562119	0.025222	1.332066
prod_180	prod_DZS	0.137621	0.300593	2.562119	0.025222	1.262037
prod_IN8	prod_DLB	0.161250	0.174224	2.386394	0.016321	1.122572
prod_DLB	prod_IN8	0.073007	0.384807	2.386394	0.016321	1.363392

Tabulka 5.7: Vygenerovaná asociační pravidla

tovního pojištění, které bychom doporučovali zákazníkům, kteří již v minulosti cestovního pojištění u naší společnosti využili.

Ostatní pojistné produkty uvedené v tabulce se týkají povinného ručení (v předchozí části druh `POV`). Zde tedy nejspíše odhalujeme praktiku, kdy je věrnému zákazníkovi po určité době nabízen výhodnější produkt z té samé kategorie. Toto zjištění nás tedy ze subjektivního hlediska nepřekvapí, ani na jeho základě nebude provedena konkrétní akce, může ovšem sloužit čtenáři jako jistá forma potvrzení této skutečnosti a jako znalost, kterou může využít při řešení svého příštího povinného ručení.

5.5 Shrnutí

V této kapitole byly na teoretickém základě a praktické implementaci popsané v této práci vystavěny experimenty, díky kterým byly dolovány znalosti z dat pojišťovny. Nejdříve byly popsány experimenty provedené nad natrénovanými klasifikátory, na jejichž základě byly získány znalosti o procesu likvidace pojistných událostí. Bylo zjištěno, jaké skutečnosti mají vliv na délku trvání likvidace a výši pojistného plnění. V druhé části této kapitoly byly na základě vydolovaných asociačních pravidel zjištěny informace o chování zákazníků pojišťovny a rovněž byla získána určitá míra vhledu do fungování pojišťovny a nabídky pojištění.

Kapitola 6

Závěr

V této práci jsme byli nejdříve seznámeni s problematikou získávání znalostí z databází v kapitole 2. V kapitole 3 jsme prozkoumali prostředí jazyka Python, které sloužilo jako implementační rozhraní pro vypracování práce. Dále jsme se v kapitole 4 seznámili také s daty poskytnutými pojišťovnou, na jejichž základě byly navrženy úlohy získávání znalostí z databází, konkrétně dolování asociačních pravidel a klasifikace pojistných událostí. Tyto úlohy byly implementovány v prostředí jazyka Python a jeho knihoven (popsáno v kapitole 4) a byly nad nimi provedeny experimenty, které jsou popsány v kapitole 5. Ty byly na závěr této práce vyhodnoceny a na jejich základě byly získány zajímavé a užitečné znalosti z fungování pojišťovny.

Nejdůležitějším přínosem této práce jsou právě znalosti, které má dolování z dat přinášet. V průběhu práce jsme díky spojení více druhů a zdrojů dat a provedení více druhů dolovacích úloh prozkoumali zajímavé souvislosti, které se týkají pojišťovnictví. Ukázali jsme si také, jak je možné z komplexních a nepřehledných dat získat jednoduché a snadno pochopitelné znalosti. A v neposlední řadě jsme na vlastní kůži zažili, jak je datová analýza krásná a zajímavá oblast informatiky, která má potenciál pozorného a zvídavého datového analytika v mnohém obohatit.

V průběhu tvorby této práce nebyla možnost konzultovat výsledky s odborníky z oblasti pojišťovnictví. Tato limitace je velkým prostorem pro zlepšení při případném pokračování práce, protože odborné konzultace by vyústily ve větší počty asociačních pravidel a klasifikačních atributů k diskuzi. Celkově by tím byla tato práce významně obohacena a zjištěné skutečnosti by mohly být přesněji formulovány.

Možných pokračování této práce je mnoho. V průběhu této práce byly prozkoumány pouze některé oblasti pojišťovnictví, je proto nasnadě zaměřit se na další oblasti, jako jsou například výkony pojišťovacích poradců, spojit použitá data s volně dostupnými geoprostorovými daty a odhalovat další souvislosti pojišťovnictví v širším kontextu. Během dalšího pokračování by také mohly být prozkoumány složitější algoritmy, konkrétně hluboké neuronové sítě, které by mohly být použity pro robustnější a přesnější klasifikaci i predikci v stávajících dolovacích úlohách.

Literatura

[1] *Class Imbalance Problem.* 2013. Accessed: 22. 4. 2024. Dostupné z: http://www.chioka.in/class-imbalance-problem/.

[2] Breiman, L. Random forests. *Machine learning.* Springer. 2001, sv. 45, s. 5–32.

[3] Chawla, N. V., Bowyer, K. W., Hall, L. O. a Kegelmeyer, W. P. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of artificial intelligence research.* 2002, sv. 16, s. 321–357.

[4] Han, J., Kamber, M. a Pei, J. *Data Mining: Concepts and Techniques.* 3. vyd. Amsterdam: Morgan Kaufmann Publishers, 2012. Morgan Kaufmann Series in Data Management Systems. ISBN 978-0-12-381479-1.

[5] Han, J., Pei, J., Yin, Y. a Mao, R. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery.* Springer. 2004, sv. 8, s. 53–87.

[6] Ng, A. a Jordan, M. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems.* 2001, sv. 14.

[7] Sagiroglu, S. a Sinanc, D. Big data: A review. In: *2013 International Conference on Collaboration Technologies and Systems (CTS).* 2013, s. 42–47. DOI: 10.1109/CTS.2013.6567202.

[8] Shearer, C. The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing.* 2000, sv. 5, č. 4, s. 13–22.

[9] Silberschatz, A. a Tuzhilin, A. On Subjective Measures of Interestingness in Knowledge Discovery. In:. Leden 1995, s. 275–281.

[10] ZENDULKA, J., BARTÍK, V., LUKÁŠ, R. a RUDOLFOVÁ, I. *Získávaní znalostí z databází. Studijní opora* [Fakulta informačních technologií VUT v Brně]. 2009.

[11] Český statistický úřad. *Cestování Čechů v roce 2022.* Accessed: 22. 4. 2024. Dostupné z: https://www.czso.cz/csu/xl/cestovani-cechu-v-roce-2022.

[12] Český statistický úřad. *Finanční situace domácností - 2021 a 2022.* Accessed: 22. 4. 2024. Dostupné z: https://www.czso.cz/csu/czso/financni-situace-domacnosti-2021-a-2022#.

[13] Český statistický úřad. *Český statistický úřad.* Accessed: 22. 4. 2024. Dostupné z: https://www.czso.cz/csu/xa.