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 DATABÁZÍ S VYUŽITÍM JAZYKA R

KNOWLEDGE DISCOVERY FROM DATABASES WITH USE OF THE R LANGUAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETER KRUTÝ**
AUTHOR

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

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Krutý Peter**
Program: Informační technologie
Název: **Získávání znalostí z databází s využitím jazyka R**
Knowledge Discovery from Databases with Use of the R Language
Kategorie: Data mining

Zadání:

1. Seznamte se s problematikou získávání znalostí z databází.
2. Seznamte se s programovacím jazykem R, prostředím nástroje R Studio a jeho podporou pro jednotlivé metody získávání znalostí, podrobněji se zaměřte na metody klasifikace.
3. Po dohodě s vedoucím zvolte alespoň dva vhodné datové vzorky, vhodné pro demonstraci metod klasifikace v jazyce R.
4. Řešení úlohy implementujte a proveďte experimenty na zvolených vzorcích dat, vyhodnoťte úspěšnost klasifikace.
5. Zhodnoťte výhody a nevýhody jazyka R pro řešení úloh klasifikace. Diskutujte další možná rozšíření vaší práce.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1.
- Torgo, L.: Data Mining with R. Learning with Case Studies. Chapman & Hall/CRC Press. 2011. 289 p.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

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. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 24. října 2019

Abstrakt

Táto práca sa zaoberá problematikou získavania znalostí z databáz. Jej cieľom je vykonať prieskum možností jazyka R a jeho podpory pre túto oblasť. Možnosti jazyka sú zisťované pomocou experimentov, pre ktoré boli vybraté vhodné dátové sady. Detailnejšia pozornosť je upriamená na metódy klasifikácie, zhlukovania a zisk asociačných pravidiel. Výstupom práce je porovnanie aplikácie jednotlivých metód v jazyku R a zhodnotenie vhodnosti použitia jazyka pre oblasť získavania znalostí z databáz.

Abstract

This thesis is focused on the field of knowledge discovery from databases. Main objective is to research possibilities of R language and its support in this area. Support is researched by experiments using appropriate data sets. More detailed attention is given to the methods of classification, clustering and association rules learning. The output of the thesis is comparison of methods application in R and defining the suitability of using language for knowledge discovery from databases.

Kľúčové slová

získavanie znalostí z databáz, dolovanie z dát, programovací jazyk R, RStudio, klasifikácia, zhlukovanie, zisk asociačných pravidiel, analýza dát, vizualizácia dát, rozhodovacie stromy, neurónové siete, bayesovská klasifikácia

Keywords

knowledge discovery from databases, data mining, R programming language, RStudio, classification, clustering, association rule learning, data analysis, data visualization, decision trees, neural networks, Bayes classification

Citácia

KRUTÝ, Peter. *Získávání znalostí z databází s využitím jazyka R.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Získávání znalostí z databází s využitím jazyka R

Prehlásenie

Prehlasujem, že túto bakalársku prácu som vypracoval samostatne pod vedením pána Ing. Vladimíra Bartíka, Ph.D. Pravdivo som uviedol všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Peter Krutý
28. mája 2020

Poďakovanie

Rád by som poďakoval vedúcemu práce, pánovi Ing. Vladimírovi Bartíkovi, Ph.D. za odbornú pomoc, trpezlivosť a cenné rady. Ďalej ďakujem Mgr. Kataríne Krutej za pravopisnú korekciu textu.

Obsah

Kapitola 1

Úvod

Žijeme v dobe dát. Extrémny pokrok v technológiách spôsobil, že každým dňom pribudne do úložísk po celom svete obrovské množstvo údajov, ktoré máme možnosť zachytávať, spracovávať, analyzovať a ukladať. Medzi zdroje generovania týchto údajov patria do veľkej miery ľudia a stroje komunikujúce v globálnej sieti internet. Existuje však aj mnoho ďalších. Množstvo elektronicky uložených údajov je dnes tak extrémne veľké, že jeho manuálne spracovávanie a analýza je časovo neefektívna a v niektorých prípadoch až nemožná. Je teda potreba tieto činnosti automatizovať.

V dnešnej dobe existuje veľké množstvo bezplatných, ale aj komerčných nástrojov pre automatizáciu. Jedným z prostriedkov automatizovaného spracovania, analýzy a vizualizácie je programovací jazyk R. Ide o jeden z najobľúbenejších a najefektívnejších jazykov pre tento účel. V súčasnosti je vďaka bezplatnej licencii považovaný v mnohých vedných oblastiach za štandard. Táto práca sa zameriava práve na prieskum a zhodnotenie jeho možností.

Text je logicky rozdelený do niekoľkých kapitol. V kapitole 2 je všeobecne popísaný teoretický úvod pre získavanie znalostí z databáz. Pre konkrétne metódy dolovacích úloh je vyhradená samostatná kapitola 3. Znalosť princípu popísaných metód je potrebná k pochopeniu ďalších kapitol. Kapitola 4 popisuje jazyk R, jeho vlastnosti, možnosti a podporu pre zisk znalostí. Táto kapitola naviac obsahuje popis integrovaného vývojového prostredia RStudio. V kapitole 5 sú prezentované experimenty, ktorými sú demonštrované možnosti jazyka. Konkrétny experiment zahŕňa predstavenie dátovej sady, ukážku implementovanej dolovacej úlohy a zhodnotenie výsledkov. Záverečné zhrnutie výhod a nevýhod jazyka R nielen v oblasti získavania znalostí z databáz sa nachádza v kapitole 6.

Kapitola 2

Získavanie znalostí z databáz

Táto kapitola je zameraná na predstavenie problematiky získavania znalostí z databáz. Ide o základný popis teórie, ktorej vysvetlenie je potrebné pre pochopenie experimentálnej časti práce.

Úvod kapitoly je vyhradený predstaveniu oblasti získavania znalostí z databáz a vysvetlením dôležitých pojmov, ktoré s touto oblasťou súvisia. Získavanie znalostí z databáz je chápané ako proces zložený z niekoľkých krokov a tieto kroky sú v jednej z podkapitol detailne popísané. Ďalšou dôležitou časťou tejto kapitoly je vymedzenie základných zdrojov dát pre dolovanie. Koniec kapitoly je venovaný popisu typických úloh pre dolovanie z dát.

2.1 Čo je to získavanie znalostí z databáz

Získavanie znalostí z databáz je extrémne rýchlo rastúci vedný odbor, ktorý poskytuje širokú podporu v analýze a rozhodovaní. Formálne ide o proces extrakcie relevantných, netriviálnych, skrytých, vopred neznámych, užitočných vzorov a modelov z obvykle veľkého množstva údajov [4]. Extrahované vzory a modely následne reprezentujú získané znalosti.

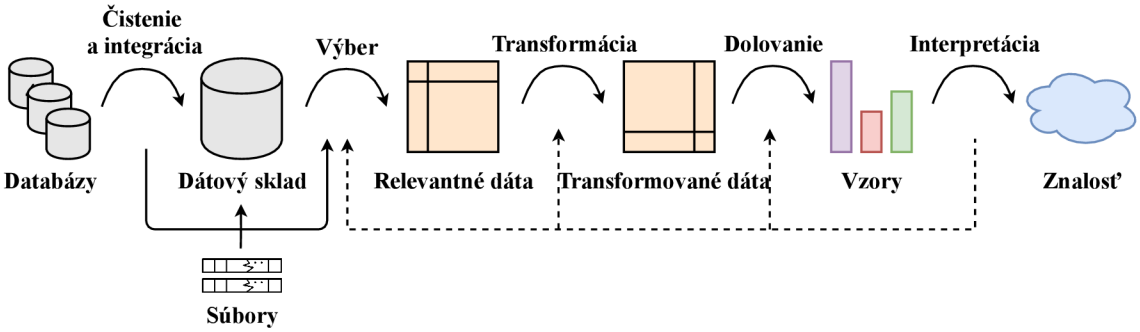
Tento vedný odbor ťaží z mnohých iných výskumných oblastí. Ide napríklad o štatistiku, strojové učenie, rozpoznávanie vzorov, databázy, algoritmy alebo vizualizáciu [9]. Medzi jeho praktické aplikácie naopak patrí napríklad biznis, bioinformatika, financie, zdravotníctvo alebo vzdelávanie [2]. Všetky tieto oblasti vďaka zisku znalostí z údajov výrazným spôsobom profitujú.

Aj keď je definícia celkom jasná, pomerne často sa stretávame s tým, že niektoré operácie nad údajmi sú do tejto oblasti nesprávne zaraďované. Je teda vhodné uviesť príklady operácií, ktoré do tejto oblasti nepatria. Napríklad obyčajné vyhľadávanie údajov alebo jednoduché príkazy SELECT nad údajmi porušujú vlastnosť netriviálnosti. Hľadanie jasne viditeľných vzorov by zas porušovalo vlastnosť skrytosti. Vo všeobecnosti sa dá povedať, že ide o operácie, ktoré istým spôsobom porušujú jednu z vlastností znalosti uvedenej v definícii.

Význam pojmu získavania znalostí z databáz je v literatúre neraz zavádzajúci. Pomerne často sa stretávame s prípadom, kedy je zamieňaný s pojmom dolovanie z dát (angl. data mining). Niektorí ľudia chápu tieto dva pojmy ako synonymá. V tejto práci je však z dôvodu jednoznačnosti dolovanie z dát vnímané ako jedna časť procesu získavania znalostí. Podrobnejší popis procesu zisku znalostí a dolovania z dát je popísaný v nasledujúcej kapitole.

2.2 Proces získavania znalostí a jeho primitíva

Tak ako bolo spomenuté v úvode kapitoly, získavanie znalostí z databáz je proces zložený z niekoľkých krokov. Kroky sa obvykle opakujú vo vopred nestanovenom poradí, pričom na predošlý krok sa vracia iba v prípade potreby. Tak ako je uvedené na obrázku 2.1, proces získavania znalostí obsahuje sedem krokov [6]:



Obr. 2.1: Proces získavania znalostí (prevzaté z [6])

1. **Čistenie dát** – úlohou čistenia dát je odstránenie šumu, vyriešenie nekonzistencie a vysporiadanie sa s chýbajúcimi údajmi.
2. **Integrácia dát** – úlohou integrácie dát je zlúčenie dát pochádzajúcich z rôznych dátových zdrojov.
3. **Výber dát** – úlohou výberu dát je vybrať dáta, ktoré sú z pohľadu dolovacej úlohy relevantné.
4. **Transformácia dát** – úlohou transformácie dát je zmena formátu dát do konsolidovanej podoby vhodnej pre dolovanie z dát. Zmenou formátu sa rozumie napríklad vykonanie sumarizácie alebo agregácie.
5. **Dolovanie z dát** – úlohou dolovania z dát je extrakcia vzorov z transformovaných dát, resp. vytvorenie modelu dát aplikovaním určitej metódy s konkrétnou implementáciou.
6. **Hodnotenie vzorov/modelu** – úlohou hodnotenia vzorov/modelu je identifikovať skutočne zaujímave vzory na základe miery užitočnosti.
7. **Interpretácia/Prezentácia znalostí** – úlohou interpretácie/prezentácie znalostí je prezentovať/interpretovať získané vedomosti z vytvorených vzorov/modelu. Obvykle sú využívané rôzne techniky vizualizácie a reprezentácie znalostí.

Kroky 1 až 4 sú označované ako fáza predspracovania pre dolovanie. Hlavný krok procesu získavania znalostí je dolovanie z dát. V tomto kroku je niekedy potrebné interagovať s používateľom alebo databázou znalostí. Proces získavania znalostí sa môže v rôznych prípadoch odlišovať. Odlišnosť procesu je definovaná na základe nasledujúcich **primitív** [6]:

- množina relevantných zdrojových údajov – výber vstupu dolovacej úlohy,
- druh dolovacej úlohy – výber typu výsledného modelu/znalosti,

- algoritmus dolovacej úlohy – výber algoritmu pre špecifický druh úlohy,
- doménová znalosť – podrobnejšie špecifikovanie problému ovplyvňujúce priebeh dolovania,
- miera a prah zaujímavosti – definícia hraničných hodnôt, napr. pre asociačné pravidlá minimálna podpora a spoľahlivosť,
- spôsob prezentácie výsledkov – výber spôsobu vizualizácie znalostí.

2.3 Typy zdrojových dát pre dolovanie

Všeobecne vzaté, dolovanie z dát je možné vykonať nad akýmkoľvek typom zmysluplných údajov. Môže ísť o údaje, ktoré sú perzistentne uložené v nejakej databáze alebo aj o tranzientné dáta, t. j. prúd dát. Najzákladnejšími typmi zdrojových dát, s ktorými sa môžeme stretávať, sú relačné databázy, dátové sklady a transakčné databázy. Existujú však aj iné typy zdrojových dát, ktoré sú v tejto podkapitole taktiež popísané. [6]

Relačné databázy

Hlavným prvkom relačných databáz je relácia. Relácia pozostáva z dvoch častí, a to schéma relácie a inštancia relácie. **Inštancia relácie** je tabuľka zložená z množiny niekoľkých záznamov. Záznam je chápaný ako jeden riadok (n-tica), pričom všetky riadky majú rovnaký počet stĺpcov (atribútov). Názov a doménu každého stĺpca (atribútu) špecifikuje **schéma relácie**. Každý záznam v tabuľke je identifikovaný primárnym kľúčom zloženým z neprázdnej množiny atribútov. Tabuľka musí takisto spĺňať podmienku prvej normálnej formy, t. j. jednotlivé stĺpce tabuľky musia obsahovať iba atomické hodnoty. [14]

Údaje v relačných databázach sú sprístupňované pomocou databázových príkazov typicky zapísaných v jazyku SQL. Pri sprístupňovaní je následne špecifický príkaz transformovaný na množinu relačných operácií, ako napríklad spojenie, projekcia alebo selekcia [6]. Tak ako bolo naznačené v podkapitole 2.1, obyčajné SQL SELECT príkazy však nepovažujeme za dolovanie z dát, nakoľko porušujú vlastnosť netriviálnosti. V prípade dolovania z dát by sme museli zisťovať komplexnejšie informácie, ako napríklad trendy, vzory a modely.

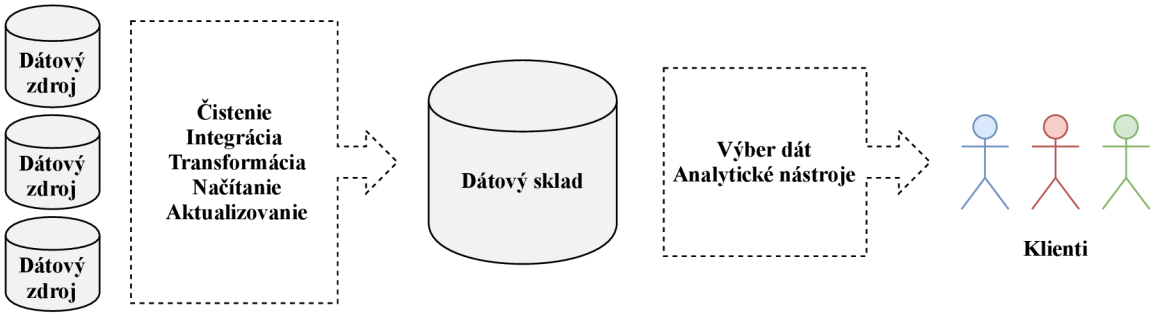
Relačné databázy môžeme považovať za najbežnejšie a najbohatšie zdroje dát a to je aj dôvod, prečo patria medzi najviac využívaný typ dát pre dolovanie. V praktickej časti tejto práce sú vykonávané experimenty práve nad týmto typom zdrojových dát.

Dátové sklady

Dátový sklad je úložisko údajov, pre ktoré platí, že jeho údaje pochádzajú z niekoľkých rozdielnych dátových zdrojov. Jeho vznik vyžaduje a predchádza procesu čistenia dát, integrácie dát, transformácie dát, načítania dát a periodického aktualizovania dát. Dátové sklady sú dôležitou platformou pre získavanie znalostí a obvykle sú využívané v organizáciách pre strategické rozhodovanie. [14]

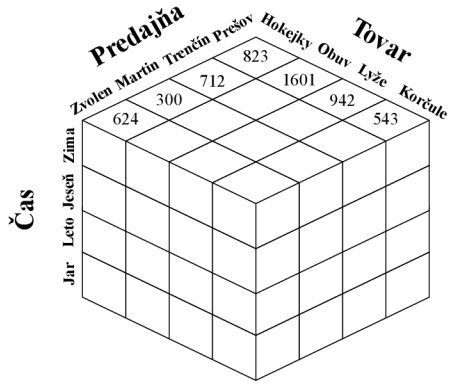
Pre uľahčenie rozhodovania sú údaje v dátovom sklade organizované okolo hlavných subjektov [6]. V prípade dátového skladu firmy predávajúcej športové potreby by hlavnými subjektmi mohli byť napr. zákazník, predávaná položka, dodávateľ atď. Pre údaje v dátovom sklade ďalej platí, že sú uložené z historickej perspektívy, navyše sú typicky sumarizované [6]. V dátovom sklade sa teda obvykle nenachádzajú jednotlivé predaje športovej potreby, ale

skôr sumarizačná hodnota predajov istého typu športovej potreby v špecifickom časovom období. Práca s dátovým skladom je demonštrovaná na obrázku 2.2.



Obr. 2.2: Tvorba a využitie dátového skladu (prevzaté z [6])

Dátový sklad je obvykle modelovaný štruktúrou, ktorá sa nazýva **multidimenzionálna kocka**. Tým, že dátový sklad poskytuje multidimenzionálny pohľad nad údajmi a obsahuje agregované údaje, je vhodný pre vykonávanie OLAP (Online Analytical Processing) operácií. Vďaka OLAP operáciám tak poskytuje omnoho väčšiu podporu pre analýzu, ako napríklad bežné relačné databázy. Grafické znázornenie multidimenzionálnej kocky je možné vidieť na obrázku 2.3. [6]



Obr. 2.3: Multidimenzionálna kocka modelujúca dátový sklad

Transakčné databázy

Transakčná databáza je úložisko organizované na základe istého druhu časovej známky. Každý záznam reprezentuje transakciu, pričom jednotlivé záznamy sú typicky uložené v obyčajnom súbore. Transakcia obyčajne zahŕňa jednoznačný identifikátor a zoznam položiek. [6]

Ako príklad je možné uviesť nákup športových potrieb zákazníkom. V tomto prípade by položkami transakcie boli identifikátor nákupu a zoznam kúpených potrieb na šport. Transakčné databázy obvykle obsahujú ďalšie tabuľky s dodatočnými informáciami o položkách transakcie. Ilustrácia transakčnej databáze je uvedená na obrázku 2.4.

Príklad 2.3.1 Analytická úloha nad transakčnou databázou
Aké športové potreby sa predávajú najčastejšie spoločne?

Na základe odpovedi na otázku z príkladu 2.3.1 by predajca mohol napr. určovať ceny produktov a zavádzať akcie tak, aby bol celkový výnos vyšší ako za normálnych okolností.

transakcia_ID	zoznam položiek
t001	P1, P3, P6
t002	P2, P3, P4
...	....

Obr. 2.4: Transakčná databáza (prevzaté z [6])

Ostatné typy zdrojových dát

Okrem relačných databáz, dátových skladov a transakčných databáz existuje množstvo ďalších typov zdrojových dát, napríklad [6]:

- **Textové databázy** – úložiská obsahujúce neštruktúrované, čiastočne štruktúrované (XML, JSON, ...) alebo štruktúrované údaje (katalógy knižníc, ...).
- **Multimediálne databázy** – úložiská obsahujúce audio údaje, obrazové údaje alebo video údaje.
- **Priestorové databázy** – úložiská obsahujúce údaje vzťahujúce sa k priestorovému usporiadaniu. Obvykle ide o geografické databázy obsahujúce údaje máp.
- **Časovo závislé databázy** – úložiská obsahujúce údaje, vzťahujúce sa k času. Môže ísť o rôzne typy udalostí, ktoré sa dejú v čase (historické záznamy, burzové údaje, ...).
- **Prúdy dát** – úložiská obsahujúce údaje, ktoré sa produkujú súvisle, pričom ich objem je obrovský a potencionálne nekonečný (údaje zo senzorov, ...). Údaje, ktoré sú najstaršie, sa typicky v čase zahadzujú.
- **Web** – veľká, rýchlo rastúca, heterogénna, široko distribuovaná databáza dostupná cez globálnu sieť internet. Obsahuje množstvo rôznych formátov (audio, video, text, ...) a disponuje veľkým potenciálom pre zisk znalostí.

2.4 Typy dolovacích úloh

Táto podkapitola je zameraná na vymedzenie a vysvetlenie základných typov dolovacích úloh. Inými slovami sú objasnené rôzne druhy modelov, ktoré môžeme z dát získavať. Ako prvá je popísaná technika popisovania konceptu/tried, následne hľadanie frekventovaných vzorov, klasifikácia, regresia, zhluková analýza a nakoniec ostatné menej časté úlohy. Vo všeobecnosti môžeme typy dolovacích úloh rozdeliť do dvoch kategórií [3]:

1. **Deskriptívne** – ich cieľom je charakterizovať všeobecné vlastnosti medzi údajmi. Typickým predstaviteľom tohto typu úloh je napríklad hľadanie asociačných pravidiel alebo zhlukovanie.

2. **Prediktívne** – ich cieľom je na základe analýzy súčasných údajov určiť, resp. predpokladať budúce správanie alebo hodnoty. Ako príklad je možné uviesť klasifikáciu alebo regresiu.

Pri procese dolovania je generované veľké množstvo vzorov. Typicky však platí, že iba malá časť z nich je pre koncového používateľa zaujímavá. Pred samotným popisom jednotlivých dolovacích techník je teda vhodné stanoviť si kritériá, ktoré odlišujú zaujímavé vzory od nezaujímavých [6]:

- jednoduchá zrozumiteľnosť pre človeka,
- platnosť pre nové alebo testovacie údaje s istým stupňom určitosti,
- potenciálna užitočnosť,
- novosť.

Uvedené vlastnosti sú teda kľúčové preto, aby vytvorený vzor mohol byť označený za znalosť.

Popis konceptu/triedy

Jedným zo základných typov dolovacej úlohy je popis konceptu/triedy (zovšeobecňovanie). Túto techniku zaraďujeme do kategórie deskriptívnych úloh. Jej cieľom je poskytnúť stručný, súhrnný a výstižný popis určitej množiny údajov [7]. Vo firme predávajúcej športové potreby môžu byť potreby rozdelené do tried: kopačky a hokejky, priestory firmy môžu byť rozdelené do tried: sklady a predajne. Tieto triedy je následne užitočné charakterizovať popisom, ktorý je možné získať jedným z nasledujúcich spôsobov [6]:

- **Charakterizácia dát** – cieľová trieda je popísaná sumarizovaním jej vlastností. Údaje analyzovanej/cieľovej triedy sú obvykle vyberané príkazom SELECT.
- **Diskriminácia dát** – cieľová trieda je popísaná rozdielmi medzi jednou alebo viacerými triedami. Triedy, s ktorými sa porovnáva, sa nazývajú rozdielové.

Príklad 2.4.1 Charakterizácia dát
Akí zákazníci športovej predajne utrácajú viac ako 200 eur ročne?

Výsledok príkladu 2.4.1 by mohol ukazovať, že ide o zákazníkov vo veku od 30 do 40 rokov, ktorých oblasť záujmu je zameraná na zimné športy.

Príklad 2.4.2 Diskriminácia dát
Porovnajte 2 skupín zákazníkov športovej predajne. Konkrétne zákazníkov, ktorí nakupujú športové potreby pre kolektívne športy a zákazníkov, ktorí nakupujú športové potreby pre individuálne športy.

Výsledok príkladu 2.4.2 môže ukazovať, že 75 % zákazníkov zameraných na kolektívne športy je vo veku do 27 rokov a nemajú vysokoškolské vzdelanie. Kým 50 % zákazníkov zameraných na individuálne športy je vo veku nad 27 rokov a majú dokončené vysokoškolské vzdelanie.

Dolovanie frekventovaných vzorov, asociácií a korelácií

Ďalším typom dolovacej úlohy je odhaľovanie vzťahov medzi atribútmi. Medzi tento typ úloh patrí dolovanie frekventovaných zdrojov, asociácií a korelácií.

Tak ako názov napovedá, **frekventované vzory** sú vzory, ktoré sa objavujú v množine údajov často. Poznáme mnoho rôznych typov frekventovaných vzorov: frekventované množiny, frekventované podpostupnosti alebo iné podštruktúry, ako napr. podgrafy, podstromy atď. Za frekventovanú množinu môžeme typicky označiť množinu položiek, ktoré sa spolu vyskytujú často. [6]

Ako príklad je vhodné uviesť lyže a lyžiarky, tieto dve položky sa obvykle vyskytujú v jednom nákupe veľmi často. Príkladom frekventovanej podpostupnosti by zase mohol byť sled nákupov jedného zákazníka. Prvý nákup by bol bicykel, následne stojan na bicykel a posledný náhradná duša do kolesa.

Dolovaním frekventovaných vzorov odhaľujeme v údajoch zaujímavé asociácie a korelácie. Proces odhaľovania zaujímavých asociácií je nazývaný ako **asociačná analýza**. Výsledkom analýzy sú tzv. **asociačné pravidlá**. Asociačná analýza býva pomerne často doplnená o hľadanie zaujímavých **korelácií** medzi dvojicami asociačných pravidiel. [3] Formát asociačného pravidla je v nasledujúcom tvare (2.1):

$$kupuje(X,'lyže') \Rightarrow kupuje(X,'lyžiarky')[podpora = 0.5\%, spoľahlivosť = 62\%], \quad (2.1)$$

kde premenná X reprezentuje zákazníka. Toto pravidlo informuje o tom, že zákazníci, ktorí si kupujú lyže, majú tendenciu v rovnakom nákupe kupovať aj lyžiarky. Významnosť pravidla v tomto prípade udávajú dva parametre: **podpora** a **spoľahlivosť**. Podpora 0.5 % značí, že 0.5 % nákupov obsahuje spolu položky lyže a lyžiarky. Spoľahlivosť 62 % značí, že v 62 % nákupoch si zákazník v prípade kúpy lyží kúpil aj lyžiarky. Podpora a spoľahlivosť sú dva najčastejšie používané parametre, avšak existujú aj ďalšie. Asociačné pravidlá, ktoré nepresiahnu minimálnu hranicu týchto parametrov, bývajú typicky odstránené.

Uvedené asociačné pravidlo obsahuje iba jeden predikát, a preto sa nazýva **jednodimenzionálne asociačné pravidlo**. Asociačné pravidlá však môžu obsahovať aj viac predikátov, napr. pravidlo (2.2):

$$\begin{aligned} pohlavie(X,'muž') \wedge vek(X,'15..25') &\Rightarrow kupuje(X,'korčule') \\ [podpora = 1\%, spoľahlivosť &= 40\%] \end{aligned} \quad (2.2)$$

obsahuje 3 predikáty: *pohlavie*, *vek* a *kupuje*. Takéto pravidlá sa nazývajú **multidimenzionálne asociačné pravidlá**. [6]

Dolovanie frekventovaných vzorov, asociácií a korelácií má široké uplatnenie v praxi. Ako príklad je možné uviesť hľadanie strategickej polohy skupín produktov v predajni alebo našepkávanie vhodných produktov pri nákupe v online obchode.

Klasifikácia a regresia

Medzi najčastejšie typy dolovacích úloh patrí aj klasifikácia a regresia. Úlohou a cieľom **klasifikácie** je vytvorenie modelu, ktorý popisuje a vymedzuje jednotlivé triedy dát. Model je následne využívaný pre predikciu kategorických hodnôt, t. j. hodnôt, ktoré sú diskrétne a neusporiadané. Toto je zároveň hlavná odlišnosť od regresie. **Regresia** je typ dátovej analýzy, na základe ktorej sú predikované numerické hodnoty spojitého charakteru. [3]

Príklad 2.4.3 Klasifikácia
Manažér firmy predávajúcej športové potreby potrebuje zistiť, ktorí zákazníci sú dôveryhodní a ktorí naopak rizikoví z pohľadu splácania kúpeného tovaru.

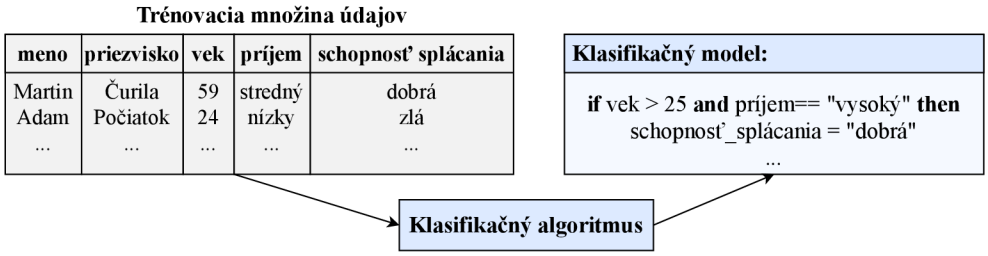
Príklad 2.4.4 Regresia
Manažér firmy predávajúcej športové potreby potrebuje zistiť, koľko peňazí minie zákazník vo veku od 20 do 30 rokov v prípade zavedenia zľavy 25 % v mesiaci december na všetok tovar.

Klasifikácia a regresia majú okrem príkladov 2.4.3 a 2.4.4 v praxi množstvo ďalších uplatnení. Pomáhajú napríklad v oblasti priemyselnej výroby, zdravotníctva, marketingu a všade inde, kde je užitočné niečo identifikovať alebo predpovedať.

Proces klasifikácie sa skladá z troch krokov: trénovanie/učenie, testovanie, aplikácia. Pred procesom klasifikácie je typicky potrebné istým spôsobom upraviť zdrojové údaje na vstupe. Medzi najzákladnejšie úpravy patria [6]:

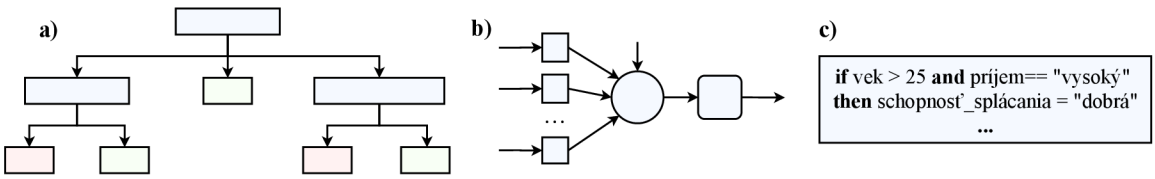
- Čistenie dát – odstránenie chýbajúcich hodnôt.
- Analýza významnosti – odstránenie atribútov objektov, ktoré sú z pohľadu klasifikácie nepotrebné.
- Transformácia dát – prevod hodnôt na iný formát (napr. kategorizácia alebo normalizácia hodnôt).

V prvej fáze, t. j. fáze **trénovania/učenia** prebieha analýza súboru dát. Tento súbor dát je označovaný ako **tréningová množina** objektov. Pre jednotlivé objekty tréningovej množiny platí, že poznáme ich triedy. Z toho vyplýva, že na základe tejto množiny dát je možné vytvoriť konkrétny klasifikačný model, ktorý dokáže predikovať triedy neznámych objektov. Priebeh tejto fázy je ilustrovaný na obrázku 2.5. [2]



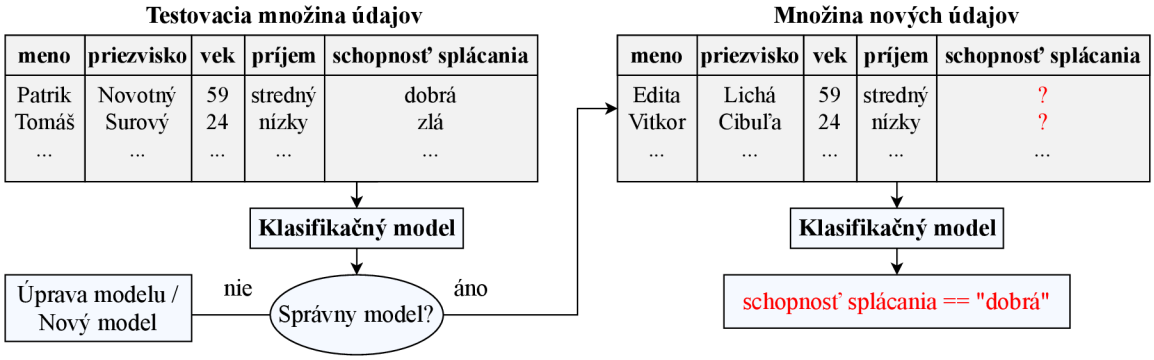
Obr. 2.5: Fáza trénovania/učenia klasifikačného modelu

Klasifikačný model môže nadobúdať rôzne podoby, napr. klasifikačné pravidlá v tvare podmienok, matematické formuly, rozhodovacie stromy alebo neurónové siete. [6] Štruktúra týchto modelov je ukázaná na obrázku 2.6.



Obr. 2.6: Reprezentácie klasifikačných modelov

Účelom druhej fázy je **testovanie** a **vyhodnotenie** klasifikačného modelu, inými slovami určenie miery úspešnosti/neúspešnosti klasifikácie. V tejto fáze sa pracuje s tzv. **validačnou množinou** objektov. Pre túto množinu údajov však platí, že ich triedy poznáme. Práve vďaka tejto vedomosti dokážeme určovať mieru úspešnosti/neúspešnosti na základe porovnania referenčných tried s výsledkami klasifikátora. [2]

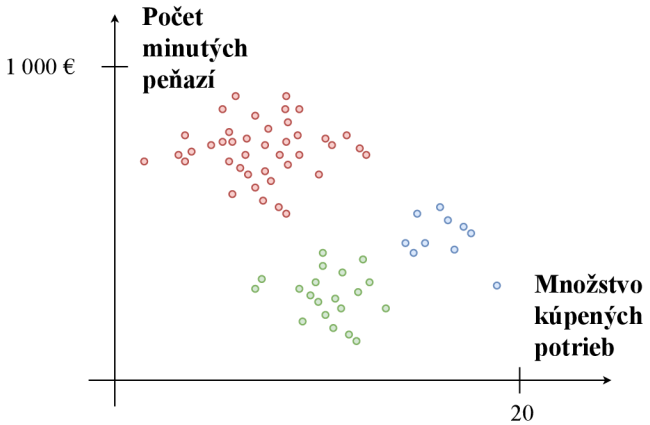


Obr. 2.7: Fázy testovania a aplikácie klasifikačného modelu

Na základe výsledkov z predošlého kroku je následne potrebné rozhodnúť, či je alebo nie je možné model použiť. V prípade, že je model dostatočné presný, je možné klasifikovať reálne údaje s neznámymi triedami. V opačnom prípade je potrebné model upraviť, prípadne vytvoriť nový. Priebeh fázy testovania a klasifikácie je uvedený na obrázku 2.7.

Zhluková analýza

Zhluková analýza je ďalším obvyklým typom dolovacej úlohy. Jej cieľom je zoskupovanie objektov do skupín na základe vopred definovaného kritéria. Výsledkom zhlukovania sú teda homogénne skupiny. Objekty v jednej skupine by mali byť v ideálnom prípade maximálne podobné a zároveň minimálne podobné s objektmi inej skupiny. [17]



Obr. 2.8: Zhlukovanie zákazníkov športového obchodu

Ako ukážku je vhodné uviesť obrázok 2.8, kde sú zhlukovaní zákazníci firmy predávajúcej športové potreby. Zhlukovanie prebieha na základe počtu minutých peňazí a množstva kúpených potrieb za jeden konkrétny mesiac.

Ostatné typy dolovacích úloh

V tejto sekcii sú popísané dolovacie úlohy, ktoré nie sú v praxi tak obvyklé ako úlohy uvedené doposiaľ. Je ale vhodné spomenúť ich.

Medzi takýto typ dolovacej úlohy patrí **analýza odľahlých objektov**. Už názov napovedá, že v tejto úlohe z pohľadu tvorby modelu nie sú zaujímave obvyklé hodnoty. Práve naopak, zaujímajú nás hodnoty, ktoré sa významne odlišujú od ostatných [6]. V praxi by mohlo ísť napríklad o detekciu zneužitia platobnej karty zákazníka. Podozrivé správanie by sa mohlo prejaviť neobvykle veľkým nákupom alebo nadmieru veľkým počtom nákupov.

Posledným typom dolovacej úlohy, ktorý je v tejto podkapitole popísaný, je **analýza evolúcie údajov**. Jej cieľom je popisovať a modelovať vývoj a trendy jednotlivých objektov v čase [3]. Konkrétnym príkladom takejto úlohy je analýza podobností alebo vyhľadávanie periodicity v čase. Výsledky by následne mohli byť využité pri predvídaní budúcnosti a podpore rozhodovania.

Kapitola 3

Metódy dolovania

Úlohou tejto kapitoly je detailnejšie predstaviť konkrétne metódy pre úlohy dolovania asociačných pravidiel, klasifikácie a zhlukovania.

Pre každú dolovaciu úlohu je vyčlenená jedna podkapitola. Prvá podkapitola je venovaná popisu metód dolovania asociačných pravidiel, konkrétne algoritmov Apriori, Eclat a metódy FP stromu. Po nej nasleduje podkapitola zameraná na metódy klasifikácie, konkrétne bayesovskú klasifikáciu, rozhodovacie stromy a neurónové siete. Trojicu dolovacích úloh uzatvára popis metód zhlukovania, konkrétne zhlukovanie založené na rozdeľovaní, hierarchické zhlukovanie a zhlukovanie založené na hustote.

3.1 Metódy dolovania asociačných pravidiel

V tejto podkapitole sú popísané a porovnané jednotlivé metódy dolovania asociačných pravidiel. V prvej časti je vysvetlený algoritmus Apriori, po ňom nasleduje stručný popis odlišností algoritmu Eclat a metódy FP stromu.

Algoritmus Apriori

Algoritmus Apriori je jednou z najznámejších metód generovania frekventovaných množín. Už meno tohto algoritmu napovedá, že princíp je postavený na využívaní predošlej (lat. prior) znalosti. Algoritmus v každej iterácií prechádza celú databázu, pričom k vytvoreniu $(k+1)$-množiny sa využíva k-množina. Algoritmus ďalej disponuje **Apriori vlastnosťou**, ktorá hovorí, že každá podmnožina frekventovanej množiny je taktiež frekventovaná. Proces použitia tejto vlastnosti sa skladá z dvoch krokov [6]:

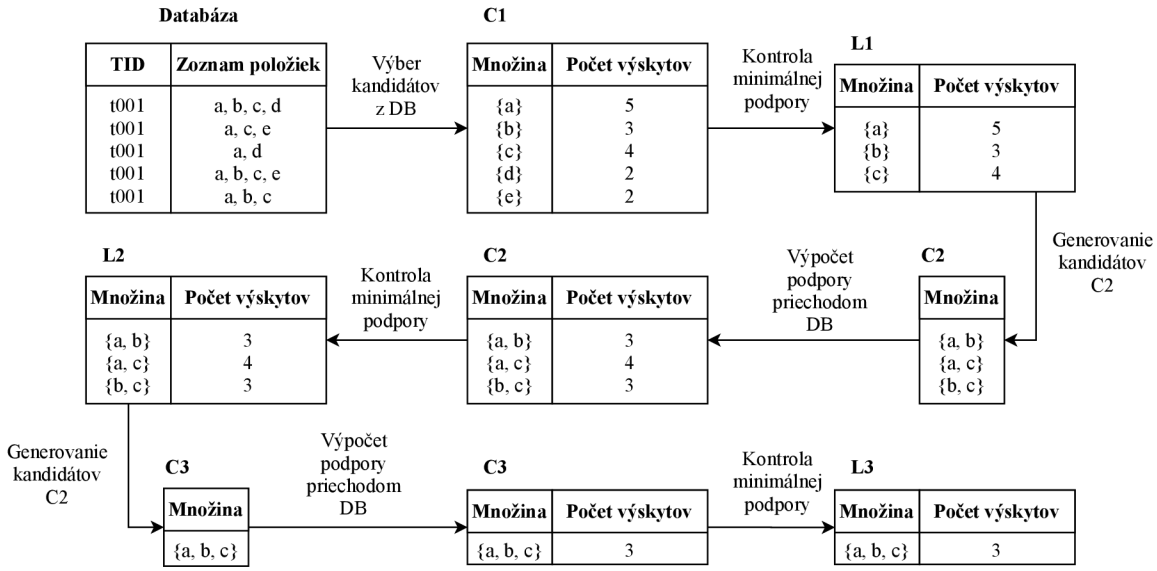
1. **Spojovací krok** – K vytvoreniu L_k (všetkých k-množín) je potrebné vygenerovať množinu kandidátov C_k spojením L_{k-1}. Nech l_1 a l_2 sú $(k-1)$-množiny z L_{k-1}. Algoritmus predpokladá, že položky v množinách sú lexikograficky zoradené. K spojeniu dôjde, ak ich prvých $k-2$ prvkov je zhodných. Výsledná množina po spojení vyzerá nasledovne: $l_1[1], l_1[2], \ldots, l_1[k-1], l_2[k-1]$, pričom musí platiť, že $l_1[k-1] < l_2[k-1]$. Tato podmienka zabezpečuje to, že sa v L_k negenerujú duplicity a zároveň sú všetky k-množiny zoradené.

2. **Vylučovací krok** – množina kandidátov C_k je nadmnožinou L_k. To znamená, že jej prvky môžu, ale nemusia byť frekventovanými množinami. Zisťovanie výskytu každého kandidáta priechodom databáze je pri veľkom počte položiek veľmi neefektívne, preto je využitá Apriori vlastnosť. Ak existuje $(k-1)$-množina, ktorá je podmnožinou

kandidátnej k-množiny a nie je obsadená v L_{k-1} (nie je frekventovaná), tak je daná kandidátna k-množina odstránená.

Ďalšou vlastnosťou algoritmu je, že frekventované množiny sú **nadol uzavreté**. To znamená, že ak množina spĺňa požiadavku minimálnej podpory, tak ju spĺňajú aj všetky jej podmnožiny [3].

K základnému algoritmu Apriori existuje množstvo modifikácií, ktoré ho vylepšujú. Ide napríklad o zavedenie hashovania, prísnejšiu redukciu kandidátov alebo vzorkovanie. Algoritmus Apriori rovnako dokáže pracovať s transakčnou ale aj relačnou databázou. [6] Ukážka funkčnosti nad databázou transakcií je uvedená na obrázku 3.1.



Obr. 3.1: Algoritmus Apriori

Ostatné metódy

Na dolovanie asociačných pravidiel je možné využiť množstvo ďalších metód. Za spomenutie stojí algoritmus Eclat a metóda FP (Frequent pattern) stromu.

Algoritmus **Eclat** je rekurzívny algoritmus, ktorý pracuje po vzore hĺbkového vyhľadávania v grafe. Ide o rýchlejší a pamäťovo menej náročný variant algoritmu Apriori, ktorý naopak pracuje po vzore vyhľadávania do šírky. Hlavnou myšlienkou je rekurzívna práca s množinou identifikátorov položiek. V každom rekurzívnom volaní je pre každú množinu identifikátorov (kandidáta) overovaná minimálna podpora a následne sú množiny navzájom kombinované. Tento proces pokračuje, pokým existujú množiny, ktoré je možné kombinovať. [10]

Odlišnosť metódy **FP stromu** zase spočíva v konvertovaní celej databáze položiek do stromu, s ktorým sa následne pracuje po celú dobu algoritmu. Hlavnými výhodami zavedenia stromu od algoritmov Apriori a Eclat je absencia generovania kandidátov a absencia opakovaných priechodov celou databázou. [5]

3.2 Metódy klasifikácie

Už v predošlej kapitole bolo spomenuté, že pri klasifikácii je možné využiť niekoľko metód. V tejto podkapitole si jednotlivé metódy popíšeme a porovnáme. Typické kritériá pre porovnanie klasifikačných metód sú nasledujúce [6]:

- presnosť – miera správnosti klasifikovania,
- rýchlosť – výpočtová zložitosť pre vytvorenie a využívanie klasifikačného modelu,
- robustnosť – schopnosť tvorby klasifikačného modelu aj z neúplných údajov,
- stabilita – schopnosť vytvorenia správneho modelu z veľkého množstva dát,
- interpretovateľnosť – zložitosť modelu z pohľadu pochopenia.

Bayesovská klasifikácia

Bayesovská klasifikácia je metóda klasifikácie, ktorej princíp je založený na štatistike a pravdepodobnosti. Hlavnou myšlienkou tejto metódy je štatistickými metódami špecifikovať, s akou pravdepodobnosťou objekty patria do jednotlivých tried. Objekt je následne zaradený do triedy, pre ktorú má vypočítanú najvyššiu pravdepodobnosť.

Pre vysvetlenie jednoduchej bayesovskej klasifikácie je potrebné zaviesť nasledujúce označenie [6]:

- S – množina všetkých objektov (tréningových dát).
- $C_1, C_2, \ldots, C_m$ – jednotlivé triedy, do ktorých sú objekty z množiny S klasifikované. Symbol m teda udáva celkový počet týchto tried.
- s_i – počet prvkov z množiny S, ktoré sú klasifikované do triedy C_i, kde $i = 1, \ldots, m$.
- $A_1, A_2, \ldots, A_n$ – jednotlivé atribúty, pomocou ktorých je realizovaná klasifikácia.
- $X = (x_1, x_2, \ldots, x_n)$, kde x_i je hodnota atribútu A_i pre všetky $i = 1, \ldots, m$, značí testovací objekt, ktorý má byť klasifikovaný do nejakej triedy.
- $P(C_i|X)$, kde $i = 1, \ldots, m$ udáva podmienenú pravdepodobnosť toho, že ľubovoľný objekt padne do triedy C_i, ak vieme, že objekt má atribúty zhodné s objektom X.

Cieľom je teda nájsť i, pre ktoré je hodnota $P(C_i|X)$ najväčšia, pričom objekt X je potom zaradený do triedy C_i. Podľa Bayesovho vzorca (3.1) platí:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{3.1}$$

$P(X)$ je ale pre konkrétny objekt konštanta, a tak je dostačujúce hľadať triedu C_i, pre ktorú je hodnota výrazu $P(X|C_i)P(C_i)$ maximálna:

- $P(C_i)$ je pravdepodobnosť, že ľubovoľne zvolený prvok patrí do triedy C_i. Výpočet je realizovaný pomocou vzorca (3.2):

$$P(C_i) = s_i/|S| \tag{3.2}$$

  kde $|S|$ značí počet prvkov množiny (kardinalitu).

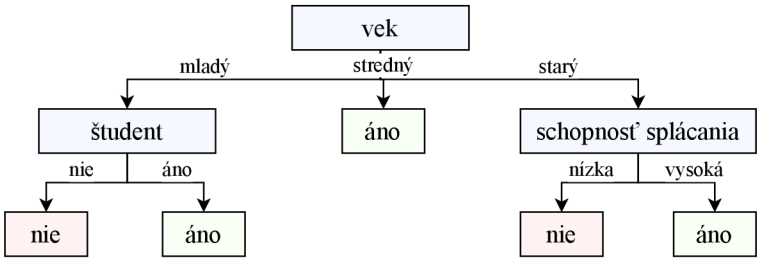
- $P(X|C_i)$ je pravdepodobnosť, že ľubovoľný objekt vybraný z triedy C_i má rovnaké hodnoty atribútov ako prvok X. Výpočet je realizovaný pomocou vzorca (3.3):

$$P(X|C_i) = \prod_{k=1}^{n} P(X_k|C_i) \tag{3.3}$$

Tento typ klasifikácie je často využívaný hlavne vďaka jeho jednoduchosti, efektívnosti, elegancii a zrozumiteľnosti. Ide o jeden z najstarších algoritmov pre klasifikáciu, pričom vďaka jeho flexibilite sa využíva v rôznych oblastiach. Konkrétnymi aplikáciami sú napr. klasifikácia textu, filtrovanie spamu či kategorizácia dokumentov. Na druhej strane dôsledkom modifikácií je odoberané z najsilnejšej vlastnosti – jednoduchosti. Najväčším negatívom tohto modelu však je, že pre objekt predpokladá nezávislosť atribútov. [12]

Klasifikácia pomocou rozhodovacích stromov

Rozhodovacie stromy sú v oblasti dolovania z dát vďaka svojim nesporným kvalitám jednou z najviac využívaných metód pre klasifikáciu. Tak ako názov napovedá, **rozhodovací strom** je model reprezentovaný grafom stromovej štruktúry. Vnútorné uzly sú označované ako rozhodovacie, nakoľko reprezentujú test na hodnoty istého atribútu, pričom každý výsledok následne predstavuje jednu vetvu. Koncové uzly (listy) reprezentujú výslednú triedu, do ktorej je objekt zaraďovaný. [12] Ukážku rozhodovacieho stromu je možné vidieť na obrázku 3.2.



Obr. 3.2: Rozhodovací strom (prevzaté z [6])

Základný algoritmus pre vytvorenie stromu je pomerne jednoduchý, avšak celková problematika klasifikácie pomocou rozhodovacieho stromu je relatívne rozsiahla a zložitá.

Pri tvorbe stromu je napríklad veľmi podstatné rozhodovať, v akom **poradí sa dopytuje na hodnoty jednotlivých atribútov**. Tento problém je riešený špeciálnym algoritmom, ktorý pracuje so zoznamom atribútov. V každom kroku rastu stromu je pre konkrétny atribút počítaný tzv. **informačný zisk**, ktorý určuje, ako dobre daný atribút rozdeľuje údaje vzhľadom na ich cieľovú klasifikáciu. K výpočtu informačného zisku je však potrebné poznať hodnotu entropie. **Entropia** je označovaná ako miera (ne)určitosti informácie. Po výpočte entropie a informačného zisku je možné určiť, ktorý atribút má najvyššiu rozhodovaciu schopnosť a bude v procese tvorby stromu vybraný ako aktuálny. [3]

Problém, ktorý je takisto vhodné uviesť, je **odstraňovanie prebytočných vetiev**. Prebytočné vetvy okrem zložitosti zhoršujú aj samotnú presnosť predpovedi. Vzniká tak potreba ich odstránenia. Základné metódy pre odstránenie sú nasledujúce [6]:

- **prepruning** – odstránenie prebytočných vetiev prebieha pri vytváraní stromu. Táto metóda je rýchlejšia.

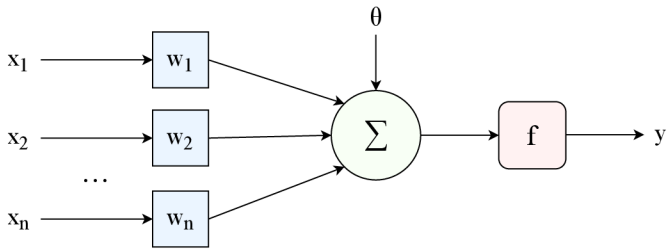
- **postpruning** – odstránenie prebytočných vetiev prebieha po úplnom vytvorení stromu. Táto metóda je spoľahlivejšia.

Rozhodovacie stromy so sebou prinášajú **ďalšie problémy**, ktoré je potrebné riešiť, napr. určovanie hĺbky, do ktorej má rozhodovací strom narásť, spracovanie spojitých atribútov, spracovanie atribútov s rozdielnym ohodnotením alebo zvyšovanie efektívnosti výpočtu. Nad rozhodovacími stromami rovnako existuje množstvo zložitejších metód, akými je napr. metóda **Náhodný les**. Táto metóda je postavená na vytvorení viac stromov, pričom pri samotnej klasifikácii je vrátený modus (najčastejšia hodnota) tried od stromov [6].

Medzi silné stránky rozhodovacích stromov patrí jednoduchá prevoditeľnosť na klasifikačné pravidlá, nízka náročnosť na množstvo výpočtov pri klasifikácii, schopnosť pracovať so spojitými aj diskrétnymi hodnotami a jasná indikácia najdôležitejších atribútov pre predikciu. Slabými stránkami sú náchylnosť k chybám v prípade klasifikácie do väčšieho množstva tried alebo potencionálna časová náročnosť vytvorenia stromu. [12]

Klasifikácia pomocou neurónových sietí

Neurónové siete sú klasifikačným modelom, ktorý je inšpirovaný činnosťou biologických štruktúr, konkrétne neurónmi. Neurónová sieť typicky obsahuje hneď niekoľko **umelých neurónov**, pričom ich zloženie je nasledovné: **telo** (biol. ozn. soma), **vstupy** (biol. ozn. dendrity), **výstup** (biol. ozn. axon) [9]. Štruktúra umelého neurónu je znázornená na obrázku 3.3.



Obr. 3.3: Umelý neurón

Činnosť neurónu spočíva v prijímaní niekoľkých **vstupných hodnôt** $x_1, x_2, \ldots, x_n$, ktoré sú získané zo vstupných dát alebo výstupov iných neurónov. Vstupné hodnoty sú následne násobené **váhovými hodnotami** $w_1, w_2, \ldots, w_n$, (biol. ozn. synapsie), ktoré prijatý signál zosilnia alebo zoslabia. Z týchto hodnôt sa následne počíta suma podľa vzorca (3.4):

$$\sum_{i=1}^{n} w_i x_i + \theta, \tag{3.4}$$

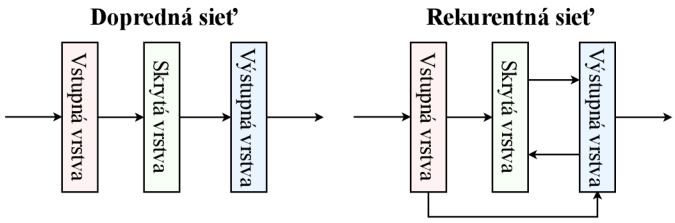
kde θ je vnútorná konštanta neurónu označovaná ako **bias**. Suma týchto hodnôt je následne privedená na vstup **aktivačnej funkcie**, ktorá generuje skokový alebo spojitý výstup y. Výstup je nakoniec privedený na vstup iných neurónov alebo môže reprezentovať výslednú hodnotu, na základe ktorej je vykonaná klasifikácia konkrétneho objektu. Uvedený popis činnosti platí pre každý neurón, ktorý sa v sieti nachádza. [6]

Neurónové siete majú mnoho odlišných topológií, pričom konkrétny typ obvykle rozdeľuje sieť do niekoľkých vrstiev. Konkrétna vrstva následne definuje typ neurónu. Medzi dve najhlavnejšie topológie zaraďujeme [3]:

1. **Dopredná sieť** – Tento typ siete sa využíva v prípade, kedy je možné predložiť kompletne celú vstupnú informáciu. Sieť reaguje na okamžitý stav vstupov, pričom informácia sa spracováva jedným smerom od vstupov k výstupom.

2. **Rekurentná sieť** – Takýto typ siete sa používa v situácii, kedy je pre výsledok dôležitá časová postupnosť vstupných údajov. Výstupy vrstiev sú typicky poprepájané so vstupmi, čím vzniká spätná väzba – pamäť. Signál prichádza cez vstupnú vrstvu a cirkuluje, kým sa stav siete neustáli. V momente ustálenia stavu je možné prečítať výstup. Z tohto typu siete je odvodená aj ďalšia, tzv. čiastočne rekurentná sieť. Jej odlišnosť spočíva v tom, že obmedzuje zavedenie rekurentných väzieb iba medzi niektoré vrstvy.

Grafické znázornenie oboch typov topológií je možné vidieť na obrázku 3.4.



Obr. 3.4: Topológie neurónových sietí

Kombinácia topológie, typu učenia a učiaceho algoritmu je označovaná ako **model neurónovej siete**. Najbežnejšími modelmi neurónových sietí sú siete so spätným šírením, Kohonenove mapy alebo RBF (Radial Basic Function) siete.

Model **siete so spätným šírením** využíva topológiu doprednej siete, učenie s učiteľom a algoritmus spätného šírenia chýb. Proces učenia tejto siete je nasledovný. Na začiatku sú hodnoty váh a biasov nastavené na malé náhodné čísla. Na vstup je privedený prvý záznam z tréningovej množiny. Nakoľko ide o učenie s učiteľom, tak záznam musí obsahovať aj požadovaný výstup. Signály sa následne zo vstupu šíria cez celú neurónovú sieť až na výstupnú vrstvu, ktorá vygeneruje výstup. Vygenerovaný výstup sa porovná s požadovaným výstupom a vypočíta sa chyba. Chyba je následne z výstupu šírená celou sieťou naspäť na vstup, pričom sa pre jednotlivé neuróny menia hodnoty váh a biasov tak, aby bola chyba minimálna. Tento proces sa opakuje pre každý záznam z tréningovej množiny. Výhodou tohto algoritmu je jeho jednoduchosť, univerzálnosť a robustnosť. Nevýhodou časová náročnosť trénovania. [6]

Neurónové siete sú dnes hlavne vďaka univerzálnosti jeden z najpopulárnejších klasifikačných modelov. Na vstupe a výstupe umožňujú pracovať s diskrétnymi, ale aj spojitými hodnotami. V porovnaní s ostatnými metódami sú vo väčšine prípadov efektívnejšie. Najväčšou nevýhodou neurónových sietí je ich netransparentnosť – nedokážeme vysvetliť ich výsledky. Ďalším mínusom je fakt, že vstupné údaje by mali byť v ideálnom prípade obmedzené v intervale 0 až 1. V istých prípadoch môže byť problematický aj proces trénovania/učenia. [9]

Ostatné typy klasifikácie

V tejto sekcii sú stručným spôsobom popísané princípy metód, ktoré sú menej obvyklé.

Prvou z nich je klasifikačný model, ktorý je založený na **k-najbližších susedoch**. Princíp tejto metódy spočíva vo výpočte euklidovskej vzdialenosti medzi dvoma objektami reprezentovanými n-ticou atribútov: $X = (x_1, x_2, ..., x_n)$ a $Y = (y_1, y_2, ..., y_n)$. Euklidovská vzdialenosť je počítaná podľa vzorca (3.5):

$$d(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.5}$$

Pri klasifikácii konkrétneho objektu je vybraných k najbližších objektov z tréningovej množiny, pričom objekt je zaradený do triedy, ktorá je pri týchto k objektoch najčastejšia. [6]

Klasifikátor založený na **fuzzy množinách** funguje na princípe určovania miery príslušnosti objektu/atribútu k jednotlivým triedam. Ide tak o zjemnenie diskrétneho rozhodovania na určovanie koeficientu z intervalu 0 až 1. [9]

Klasifikácia pomocou **genetických algoritmov** je založená na princípe prírodného vývoja a genetických vlastností. Jedinci, ktorí sú označovaní ako potomkovia, dedia od svojich rodičov gén. V prípade, že má byť potomok silnejší, vezme si od každého rodiča ten „lepší“ gen. Táto myšlienka je prenesená aj do genetických algoritmov. Gén je v organizmoch zapísaný pomocou chemického vzorca. V oblasti strojového učenia sa však zápis typicky zjednodušuje obyčajným reťazcom znakov. [3]

3.3 Metódy zhlukovania

Úlohou tejto podkapitoly je stručným spôsobom predstaviť a porovnať najznámejšie metódy/algoritmy zhlukovania. Typické kritériá pre porovnanie metód zhlukovania sú nasledujúce [6]:

- schopnosť spracovania rozsiahlych databáz,
- schopnosť spracovania rôznorodých databáz,
- schopnosť spracovania zašumených databáz,
- schopnosť spracovania vysokodimenzionálnych databáz,
- schopnosť vytvárania zhlukov rôzneho tvaru.

Všeobecne sa metódy zhlukovania delia do viacerých skupín. Zo skupiny metód založených na rozdeľovaní sú konkrétne popísané metódy k-means a k-medoids. Následne je ukázaný princíp hierarchického zhlukovania a zhlukovania založeného na hustote. Záver podkapitoly je venovaný stručnému uvedeniu ďalších, menej známejších skupín zhlukovacích metód.

Zhlukovanie založené na rozdeľovaní

Táto skupina metód je založená na rozdeľovaní n objektov do k skupín, pričom musí platiť $k \leq n$. Ďalšou podmienkou je, že každá skupina musí obsahovať aspoň jeden objekt. Samotný algoritmus týchto metód spočíva v náhodnom vybraní k objektov, ktoré na počiatku reprezentujú jednotlivé zhluky. Následne sa iteratívne hľadajú objekty, ktoré najlepšie reprezentujú daný zhluk. Pre dosiahnutie konečného výsledku sa vyžijú rôzne heuristiky, akými sú napríklad k-means alebo k-medoids. [9]

Princíp zhlukovania **k-means** je založený na nájdení stredového bodu zhluku. Konkrétne objekty sú ďalej rozdelené a priradené k stredovému bodu, ktorý je k nim najbližšie. Pri zhlukovaní týmto algoritmom treba brať do úvahy, že výsledky sa môžu každým spustením líšiť, nakoľko výber stredu zhlukov je vykonávaný náhodne. [9]

Zhlukovanie **k-medoids** je veľmi podobné zhlukovaniu k-means. Zásadná odlišnosť spočíva v tom, že v algoritme k-medoids nie je zhluk reprezentovaný svojím stredom, ale objektom, ktorý je najbližšie k stredu. Tento fakt zaručuje to, že algoritmus je v prípade prítomnosti odľahlých objektov robustnejší. [6]

Použitie týchto metód je vhodné pre hľadanie zhlukov guľatého tvaru s menším až stredným počtom objektov. Nevýhodou týchto metód je, že je potrebné vopred stanoviť počet zhlukov.

Hierarchické zhlukovanie

Tento typ zhlukovania počíta na základe vopred definovanej metriky vzdialenosť medzi jednotlivými objektmi. Na základe tejto vzdialenosti sú následne objekty hierarchicky rozdelené, pričom vzniká strom zhlukov. Rozklad je možné vykonať buď zhora nadol, alebo zdola nahor, t. j. objekty sa buď postupne rozdeľujú, alebo spájajú do zhlukov. Medzi konkrétne príklady patrí rozdeľujúca metóda AGNES a spájajúca metóda DIANA. [3]

Výhodou týchto metód je, že sú rýchlejšie ako metódy založené na rozdeľovaní. Nevýhodou je menšia presnosť a fakt, že ak niektoré zhluky rozdelíme alebo naopak spojíme, tak už ich nikdy nie je možné naspäť spojiť alebo rozdeliť.

Zhlukovanie založené na hustote

Hlavnou myšlienkou tohto typu zhlukovania je zhlukovať objekty, ktoré sú navzájom prepojené vysoko a oddelené nízko zahustenými oblasťami. Konkrétnym príkladom sú metódy DBSCAN a DENCLUE. [6]

Pre vysvetlenie metódy **DBSCAN** je potrebné stanoviť dva kľúčové parametre: veľkosť okolia a minimálny počet bodov. Ak je vo vzdialenosti okolia bodu B viac bodov ako stanovené minimum, tak sa bod B prehlási za tzv. hustý bod, t. j. bod, ktorý je spolu s bodmi jeho okolia súčasťou jedného zhluku. [3]

Metóda **DENCLUE** je založená na využití distribučných funkcií hustoty. Dátový priestor je reprezentovaný ako súčet funkcií vplyvu na okolie, pričom jedna funkcia zastupuje konkrétny objekt. Zhluky sú následne reprezentované miestami, v ktorých sa nachádzajú lokálne maximá celkovej funkcie hustoty. [6]

Výhodou týchto metód je, že umožňujú vytvárať zhluky rôznych tvarov. Ďalším plusom je, že sú schopné pracovať s odľahlými hodnotami a šumom v údajoch.

Ostatné typy zhlukovania

Okrem uvedených skupín zhlukovania existuje aj mnoho ďalších. Ide napr. o zhlukovanie založené na modeloch, zhlukovanie založené na mriežke alebo zhlukovanie vysoko dimenzionálnych údajov.

Cieľom **metód založených na modeloch** je nájsť špecifický model, ktorý do čo najväčšej miery odpovedá zdrojovým údajom. Zhluky sú následne vytvárané na základe nájdeného, optimálneho modelu. **Metódy založené na mriežke** fungujú na princípe rozdelenia priestoru objektov do buniek v tvare mriežky. Všetky operácie zhlukovania následne prebiehajú nad mriežkou, čo robí tieto metódy rýchlejšími. **Metódy pre zhlukovanie**

vysokodimenzionálnych údajov sú špeciálne metódy, ktoré sú navrhnuté s cieľom odstránenia nedokonalosti väčšiny typov zhlukovania, a to neschopnosť práce s väčším počtom atribútov. [6]

Kapitola 4

Jazyk R a jeho podpora pre dolovanie z dát

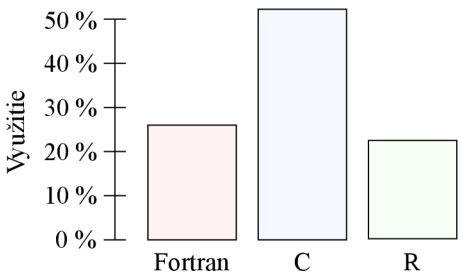
Úlohou tejto kapitoly je predstaviť jeden z mnohých prostriedkov pre získavanie znalostí z databáz, a to jazyk R.

Prvá časť kapitoly je venovaná všeobecnému popisu jazyka R a prezentácii jeho vlastností. Nasleduje popis a ukážka integrovaného vývojového prostredia RStudio. Záver kapitoly je zameraný na popis podpory jazyka R v oblasti dolovania z dát.

4.1 Základný popis jazyka R

Jazyk R je multiparadigmatický jazyk[1] určený pre štatistické výpočty a vizualizáciu údajov. Ide o projekt, ktorý vznikol ako implementácia jazyka S pod slobodnou licenciou GNU GPL[2] [8]. Nakoľko je jazyk R v súčasnosti bezplatný, tak predbehol v používaní komerčný jazyk S a stal sa tak štandardom v mnohých oblastiach výskumu. Komunita TIOBE[3] ohodnotila jazyk R vo februári 2020 ako trinásty najpopulárnejší.

Jazyk R ponúka viacero pracovných prostredí. Najbežnejším je konzola, t. j. príkazová riadka. Tak ako ukazuje obrázok 4.1, prostredie jazyka R je napísané prevažne v jazyku C, Fortran a R samotnom. Existujú však aj grafické používateľské prostredia, ako napr. integrované vývojové prostredie RStudio. Detailnejší popis tohto prostredia je uvedený v nasledujúcej podkapitole 4.2.



Obr. 4.1: Využitie jazykov v jadre R (prevzaté z [16])

[1] Multiparadigmatický jazyk – jazyk umožňujúci viacero štýlov písania zdrojového kódu programu
[2] GNU General Public License – licencia verejného a slobodného softvéru
[3] TIOBE – komunita hodnotiaca popularitu programovacích jazykov, pozri https://www.tiobe.com/

História a jazyk S

Už v úvode podkapitoly bolo uvedené, že jazyk R je odvodený zo staršieho jazyka S. Jazyk S bol vytvorený Johnom Chambersom v roku 1976, v Bell Labs, čisto v jazyku Fortran a jeho veľkou nevýhodou bolo, že bol súčasťou komerčného balíka S-PLUS. Fakt, že jazyk nebol dostupný zdarma, bol zároveň aj motiváciou pre vznik jazyka R, bezplatnú alternatívu, ktorá bola vytvorená v roku 1991 na univerzite v Aucklande. Na vzniku jazyka R sa podieľali Ross Ihaka a Robert Genteleman. Na základe ich krstných mien je aj zároveň odvodený samotný názov jazyka. [13]

Medzi týmito dvoma jazykmi existujú isté rozdiely, avšak väčšina zdrojového kódu napísaného v jazyku S funguje aj v jazyku R. Jazyk R je v dnešnej dobe považovaný za modernú implementáciu jazyka S a súčasťou tímu, ktorý sa na ňom podieľa, je aj pôvodný autor jazyka S, John Chambers. [1]

Programovacie možnosti

Táto sekcia je parafrázou textu z [15]. Tak ako bolo uvedené na začiatku podkapitoly, jazyk R je multiparadigmatický programovací jazyk. Syntax jazyka je podobná jazyku S, sémantika je zas veľmi blízka jazyku Scheme. Z jazyka LISP je odvodená reprezentácia údajov a kódu v podobe tzv. s-výrazov[4].

Za jazykom R sú skryté dva základné koncepty: objekty a funkcie. **Objekt** je možné chápať ako pomenované úložné miesto špecifickej veľkosti. V jazyku R sa všetko ukladá ako objekt. Dáta, premenné, funkcie, všetko je chápané ako objekt uložený niekde v pamäti. Špecifickým typom objektu sú **funkcie**. Ide o typ objektu, ktorý je navrhnutý tak, aby niesol funkcionalitu rôznych operácií. Na vstupe obvykle využívajú argumenty, na základe ktorých produkujú výstup.

Skalárne dátové typy sú reprezentované neobvyklým spôsobom, a to ako vektor dĺžky jeden. Jazyk R okrem vektorov zahŕňa aj množstvo iných **dátových štruktúr**: polia, dátové rámce, matice, zoznamy a ďalšie špeciálne, využívané v rôznych typoch analýz. Jazyk R, tak ako MATLAB, APL, Octave a ostatné podobné jazyky, podporuje maticovú aritmetiku. Premenné sú dynamicky typované s globálnym rozsahom. Argumenty funkcie sú predávané hodnotou a ich vyhodnocovanie je založené na princípe lazy evaluation. To znamená, že nie sú vyhodnocované v momente volania funkcie, ale až v momente, keď sú použité. Jazyk R takisto podporuje procedurálne a objektovo orientované programovanie.

Jazyk R je zaraďovaný medzi **interpretované jazyky**. Používateľovi je vykonávanie príkazov typicky sprístupnené cez interpret príkazovej riadky. Používateľ môže v príkazovom riadku napr. priradiť výraz `20 / 5` do premennej `x`. Výsledok je možné následne zobraziť jednoduchým zápisom názvu premennej, pričom interpret jazyka R zobrazí hodnotu uloženú v objekte:

```
> x <- 20 / 5
> x
[1] 4
```

Táto kalkulácia je interpretovaná ako súčet dvoch jednorozmerných vektorov, pričom výsledkom je opäť jednorozmerný vektor. Prefix `[1]` značí, že na danom riadku je vypísaný vektor, ktorý začína od prvého prvku. Toto značenie je užitočné pri výstupoch, ktoré presiahnu viac ako jeden riadok.

[4] s-výrazy (symbolický výraz) – špecifický typ notácie pre dáta so štruktúrou vnoreného zoznamu alebo stromu

Podpora dátovej analýzy a vizualizácie

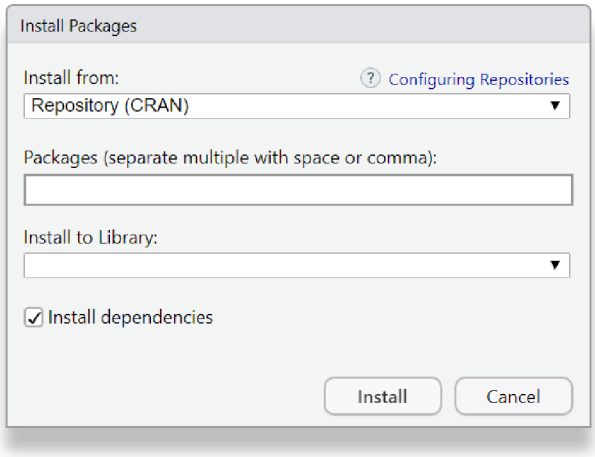
Jazyk R a jeho balíčky poskytujú širokú podporu pre rôzne typy analýzy, štatistiky a strojového učenia. Ide napríklad o nasledujúce oblasti: lineárne/nelineárne modelovanie, zhlukovanie, klasifikácia, hľadanie frekventovaných vzorov alebo analýza časových radov. Podrobnejší popis podpory v týchto oblastiach je uvedený v podkapitole 4.3.

Ďalšou silnou stránkou jazyka R je vizualizácia údajov. Jazyk spolu s jeho balíčkami ponúka širokú podporu pre vytváranie statických, ale aj dynamických grafov. Najelegantnejším a najpoužívanejším riešením pre vizualizáciu je balíček `ggplot2`[5]. Tento balíček vytvorený Hadleym Wickhamom umožňuje vytvárať grafy, ktoré dokážu pracovať s jednoduchými a štrukturovanými hodnotami spojitého, ale aj diskrétneho charakteru. Balíček ďalej podporuje širokú škálu symbolov, zmenu farby, veľkosti, priehľadnosti objektov a kvantum ďalšej užitočnej funkcionality. Okrem `ggplot2` existuje veľa ďalších balíčkov, často aj takých, ktoré sú vytvorené pre vizualizáciu výstupu konkrétnej analytickej úlohy.

Rozšírenia

Jazyk R poskytuje pre používateľov možnosť rozšírenia funkcionality v podobe balíčkov. Nielenže ich môžu slobodne využívať, môžu ich aj slobodne vytvárať. Vďaka tejto možnosti dnes existuje veľké množstvo balíčkov, konkrétne 15 462. Väčšina z nich je napísaná v jazyku R, avšak existujú aj prípady, kedy bol využitý iný jazyk, ako napr. Java, C, C++ alebo Fortran. Balíčky je možné inštalovať dvomi spôsobmi:

1. Využitím príkazu `install.packages(<názov balíka>)` v konzole.
2. Využitím nástrojov vo vývojovom prostredí RStudio, ukážka na obrázku 4.2.



Obr. 4.2: Inštalácia balíčkov v RStudio

Inštalácií balíčka predchádza stiahnutie z komplexného online repozitára **CRAN**[6]. Po stiahnutí a následnej inštalácií je potrebné zavolať príkaz `library(<názov balíčka>)`, ktorý balíček importuje do pracovného prostredia. Následne je možné využívať jeho celú funkcionalitu.

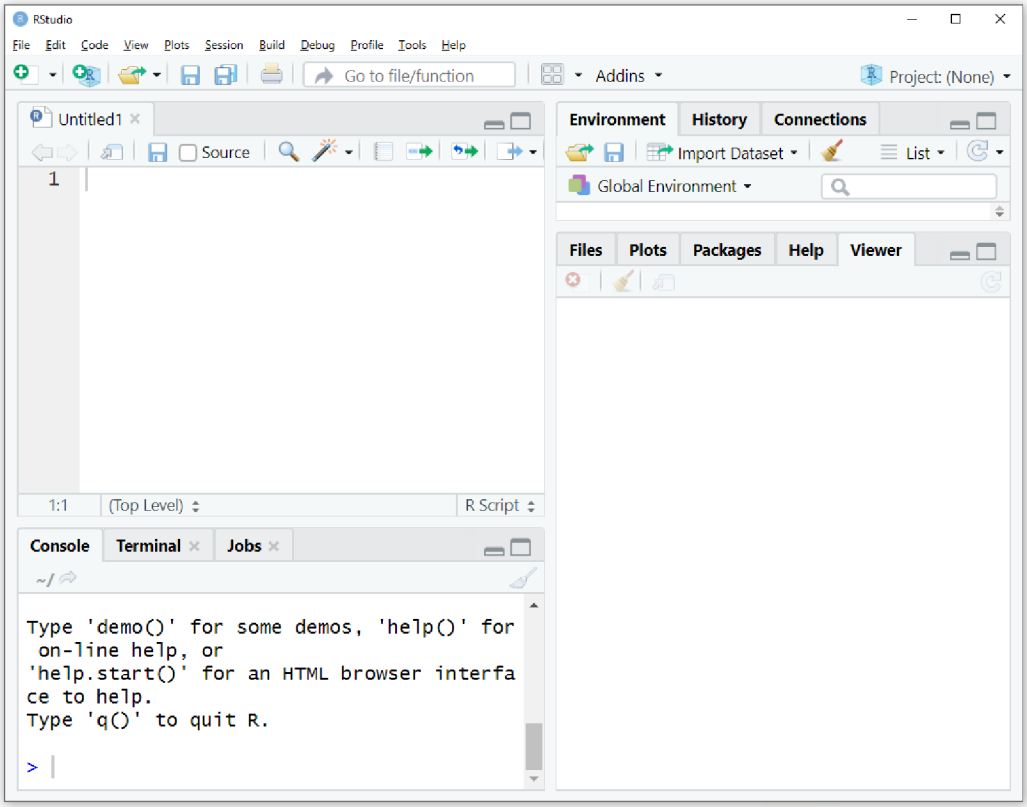
[5] `ggplot2` – balíček pre vizualizáciu údajov, pozri https://ggplot2.tidyverse.org/
[6] CRAN – komplexný online repozitár balíčkov, pozri https://cran.r-project.org/

4.2 RStudio

Na začiatku kapitoly bolo spomenuté, že jazyk R ponúka viac pracovných prostredí. Jedným z nich je integrované vývojové prostredie **RStudio**[7]. Ide o grafické používateľské rozhranie, ktoré bolo prvýkrát predstavené 28.2.2011 spoločnosťou RStudio, Inc. Tento program je teda vytvorený treťou stranou, ktorá nemá žiaden formálny vzťah s tvorcami jazyka R. Je napísaný v jazyku C++ kombinovanom s Qt, ktorý implementuje používateľské rozhranie. Okrem týchto dvoch jazykov je čiastočne použitá aj Java a Javascript.

RStudio je obľúbené hlavne kvôli tomu, že výrazným spôsobom sprehľadňuje, uľahčuje a urýchľuje programovanie v jazyku R. Je možné využívať ho bezplatne v dvoch odlišných formátoch:

1. **RStudio desktop** – bežná desktopová aplikácia, ukážka na obrázku 4.3.
2. **RStudio server** – aplikácia bežiaca na vzdialenom serveri. Používateľovi je sprístupnená cez webový prehliadač.



Obr. 4.3: Pracovné prostredie RStudio desktop

Štruktúra vývojového prostredia je flexibilná, používateľ si ju môže upraviť podľa vlastných preferencií. Jeho základné zobrazenie je však zložené z panela nástrojov a zo štyroch okien s možnosťou prepínania funkcionality:

[7] RStudio – integrované vývojové prostredie pre jazyk R, pozri https://rstudio.com/

1. **Editor** – Okno editoru umožňuje písať, ukladať, spúšťať zdrojový kód a mnoho ďalších obvyklých operácií. Takisto podporuje zvýrazňovanie syntaxe jazyka.

2. **Konzola, terminál, úlohy** – Okno v režime konzoly umožňuje spúšťať príkazy jazyka R. V režime terminálu je zas možné spúšťať príkazy konkrétneho operačného systému. Režim úloh slúži na správu spustených skriptov.

3. **Prostredie, história, pripojenia** – Okno v režime prostredia zobrazuje aktívne premenné a ich hodnoty. História ponúka možnosť zobraziť históriu spustených príkazov. Pripojenia slúžia k správe pripojených zdrojov dát.

4. **Súbory, nákresy, balíčky** – Okno v režime súborov umožňuje prehľadávať adresové štruktúry v konkrétnom systéme. Nákresy umožňujú zobrazovať výstupy vizualizačných príkazov. Balíčky slúžia k správe balíčkov.

4.3 Dolovanie z dát v jazyku R

Úlohou tejto podkapitoly je predstaviť možnosti použitia jazyka R v procese dolovania z dát. Postupne sú prezentované jednotlivé funkcie a balíčky, ktoré jazyk R ponúka. Všetky uvedené balíčky sú súčasťou komplexného online repozitára CRAN. Ich použitie je demonštrované v nasledujúcej kapitole 5.

Asociačné pravidlá

Jazyk R pre dolovanie asociačných pravidiel ponúka balíček `arules`. Balíček poskytuje dva algoritmy dolovania pravidiel. Prvým algoritmom je algoritmus Apriori. Druhým algoritmom je algoritmus Eclat. Implementáciu algoritmu Apriori predstavuje funkcia `apriori()`. Algoritmus Eclat je reprezentovaný funkciou `eclat()`. Balíček `arules` okrem týchto dvoch algoritmov ponúka aj funkcie: `summary()`, ktorá zobrazí základné štatistiky nad pravidlami a `inspect()`, ktorá vypíše jednotlivé pravidlá spolu s dodatočnými údajmi miery zaujímavosti. V prípade potreby vizualizovať asociačné pravidlá, je možné využiť balíček `arulesViz`, ktorý ponúka množstvo rôznych typov grafov.

Klasifikácia

Prediktívne modely nad údajmi je možné v jazyku R vytvárať viacerými spôsobmi. Tvorbu klasifikačných modelov a následnú predikciu diskrétnych hodnôt je možné vykonať s využitím nasledujúcich balíčkov: `naivebayes`, `party`, `rpart`, `randomForest`, `neuralnet`, ...

Bayesovskú klasifikáciu je v jazyku R možné realizovať napríklad s využitím balíčka `naivebayes`. Balíček obsahuje niekoľko rôznych implementácií tohto typu klasifikácie, pričom najzákladnejšou je funkcia `naive_bayes()`. Alternatívou k tomuto balíčku je balíček `e1071`.

Balíček `party` poskytuje podporu pre tvorbu klasifikačného modelu vo forme **rozhodovacieho stromu**. Na samotnú tvorbu stromu je určená funkcia `ctree()`, ktorá na vstupe ponúka množstvo dodatočných argumentov pre možnosť úpravy procesu tvorby. Na základe vytvoreného stromu je následne možné klasifikovať triedy pre neznáme objekty pomocou funkcie `predict()`.

Alternatívou k tvorbe modelu vo forme rozhodovacieho stromu je balíček `rpart`. Pre tvorbu stromu sa v tomto prípade využíva funkcia `rpart()`. Nevýhodou tejto funkcie je, že

nezabezpečuje automatické orezanie prebytočných vetiev a základný strom je tak častokrát zložitý [18]. Klasifikácia neznámych objektov je vykonaná rovnako ako v balíčku `party` pomocou funkcie `predict()`.

Balíček `randomForest` je využívaný pre vytváranie modelu v podobe **náhodného lesa**. Vytvorenie modelu je realizované funkciou `randomForest()`. Táto funkcia má však dve obmedzenia, pričom sa obe týkajú zdrojových dát. Prvým obmedzením je, že funkcia nedokáže pracovať s množinou dát, v ktorej chýbajú hodnoty. Druhým je, že funkcia nedokáže pracovať s množinou dát, v ktorej kategorický atribút nadobúda viac ako 32 hodnôt. [18] Alternatívna cesta pre vytvorenie tohto typu modelu je využitie funkcie `cforest()` z balíčka `party`, ktorá na rozdiel od `randomForest()` neobsahuje obmedzenie na počet hodnôt atribútu.

Balíček `neuralnet` poskytuje podporu pre klasifikáciu v podobe **neurónovej siete**. Samotné vytvorenie klasifikačného modelu je realizované funkciou `neuralnet()`, ktorá môže na vstupe prijímať množstvo konfiguračných parametrov, ako napr. výber modelu siete alebo počet neurónov v skrytej vrstve. Klasifikácia je vykonaná pomocou funkcie `compute()`.

Zhlukovanie

Jazyk R poskytuje podporu pre všetky druhy zhlukovania popísaných v podkapitole 3.3, konkrétne pre zhlukovanie založené na rozdeľovaní, hierarchické zhlukovanie a zhlukovanie založené na hustote.

Jedným z najvyužívanejších typov zhlukovania v jazyku R je zhlukovanie **k-means** realizované funkciou `kmeans()` z balička `stats`.

Predstaviteľmi zhlukovania **k-medoids** sú balíčky `cluster` a `fpc`. Balíček `cluster` ponúka algoritmy PAM a CLARA v podobe funkcií `pam()` a `clara()`. Algoritmus CLARA rozširuje algoritmus PAM a využíva sa v prípade väčšieho objemu dát na vstupe. Rozšírená implementácia algoritmu PAM sa nachádza aj v balíčku `fpc`. Ide o funkciu `pamk()`. Jej výnimočnosť spočíva v tom, že na rozdiel od predošlých funkcií nevyžaduje na vstupe počet výsledných zhlukov. Algoritmus si ich zistí sám tak, aby bol výsledok čo najviac optimálny.

Hierarchicke zhlukovanie je v jazyku R reprezentované funkciou `hclust()` z balíčka `stats`. Funkcia vytvorí hierarchiu rôznych počtov zhlukov. Samotný výber počtu zhlukov spolu so zaradením objektov je vykonaný pomocou funkcie `cutree()`.

Balíček `fpc` poskytuje podporu aj pre **zhlukovanie založené na hustote**. Konkrétne v podobe algoritmu DBSCAN reprezentovaným funkciou `dbscan()`. Alternatívou k balíčku `fpc` je balíček `dbscan`.

Podpora ostatných dolovacích úloh

Jazyk R a jeho balíčky poskytuje podporu pre množstvo ďalších dolovacích úloh, akými sú analýza odľahlých zdrojov, analýza časových radov, dolovanie z textu, ...

Kapitola 5

Riešenie experimentov v jazyku R

Táto kapitola sa venuje popisu experimentov, ktoré boli vykonané pre demonštrovanie možností jazyka R v oblasti získavania znalostí z databáz. Celkovo ide o päť experimentov:

1. **Dolovanie asociačných pravidiel** v súvislosti s prežitím pasažierov plavby lode Titanic.
2. **Bayesovská klasifikácia** výpovedí manželských/rozvedených párov.
3. **Klasifikácia** dĺžky absencie zamestnancov v práci pomocou **rozhodovacích stromov**.
4. **Klasifikácia** dizajnových prvkov mostov v meste Pittsburgh pomocou **neurónových sietí**.
5. **Zhlukovanie** semien pšenice.

Popis každého experimentu má jasnú štruktúru, ktorá sa skladá z predstavenia experimentu, popisu a ukážky dátovej sady, realizácie experimentu a zhodnotenia experimentu. Všeobecná teória k prezentovaným dolovacím úlohám je uvedená v podkapitole 2.4, popis jednotlivých metód a algoritmov sa nachádza v kapitole 3.

5.1 Asociačné pravidlá

Prvý experiment prezentuje príklad dolovania asociačných pravidiel. Experiment je realizovaný nad dátovou sadou, ktorá obsahuje údaje o pasažieroch neúspešnej plavby loďou Titanic. Cieľom experimentu je s využitím týchto údajov nájsť vzťahy medzi pasažiermi v závislosti na tom, či prežili/neprežili.

V prvej časti podkapitoly je popísaná a ukázaná dátová sada pasažierov. Následne je na ukážkach kódu popísané samotné dolovanie pravidiel, spolu s ich interpretáciou, vizualizáciou a redukciou.

Dátová sada pasažierov lode Titanic

Neúspešná plavba loďou Titanic sa v roku 1912 zapísala do histórie ako jedna z najväčších plavebných katastrof. Loď, ktorá bola považovaná za nepotopiteľnú, klesla počas svojej prvej plavby na dno oceána po zrážke s ľadovcom. Na palube nebolo dostatok záchranných člnov, čo malo za následok smrť 1502 cestujúcich z celkového počtu 2224. Prežitie pasažierov

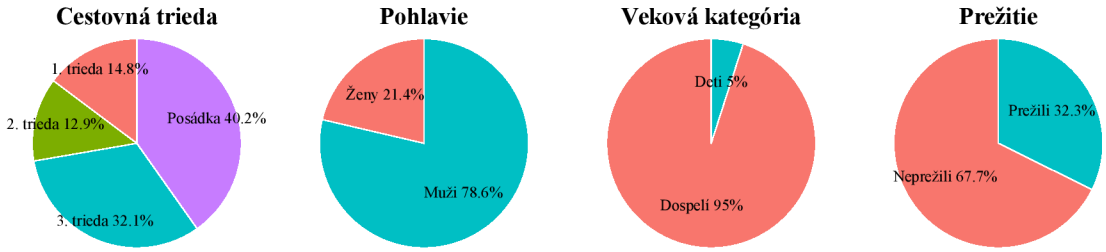
ovplyvnilo do istého stupňa aj šťastie, avšak vo veľkej miere podmieňovalo prežitie osoby to, do akej skupiny patrí.

Dátová sada je dostupná v online repozitári na webovej stránke rdatamining[1]. Obsahuje 2201 záznamov, kde jeden záznam reprezentuje pasažiera v podobe 4 atribútov. Konkrétne ide o cestovnú triedu, pohlavie, vek a prežitie. Štruktúra dátovej sady je nasledovná:

```
> str(titanic.raw)
'data.frame': 2201 obs. of 4 variables:
$ Class : Factor w/ 4 levels "1st","2nd","3rd",..: 3 3 3 3 3 3 3 3 3 3 ...
$ Sex : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
$ Age : Factor w/ 2 levels Ädult","Child": 2 2 2 2 2 2 2 2 2 2 ...
$ Survived: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

Databáza neobsahuje žiadne chýbajúce hodnoty. Zastúpenie jednotlivých atribútov, je vizualizované na obrázku 5.1 s využitím balíčka `ggplot2`.

```
> summary(titanic.raw)
  Class         Sex          Age       Survived
 1st :325   Female: 470   Adult:2092   No :1490
 2nd :285   Male  :1731   Child: 109   Yes: 711
 3rd :706
 Crew:885
```



Obr. 5.1: Vizualizácia zastúpenia atribútov s balíčkom `ggplot2`

Dolovanie pravidiel

Tak ako je uvedené v podkapitole 4.3, medzi najvyužívanejšie algoritmy pre dolovanie asociačných pravidiel v jazyku R patria algoritmy Apriori a Eclat z balíčka `arules`. V tejto podkapitole je ukázané použitie algoritmu Apriori. Algoritmus Eclat je pre tento experiment nevhodný, nakoľko nepodporuje možnosť obmedzenia na výskyt atribútu a parameter spoľahlivosti.

Algoritmus Apriori je reprezentovaný funkciou `apriori()`, ktorá na vstupe môže prijímať niekoľko argumentov. Argumentom `paramater` je napríklad možné ovplyvniť proces dolovania tak, že budú vyberané iba pravidlá, ktoré spĺňajú podmienky významnostných parametrov. Predvolené nastavenie funkcie `apriori()` pre významnostné parametre je nasledovné: `support = 0.1` – minimálna podpora pravidla, `confidence = 0.8` – minimálna spoľahlivosť pravidla. V prípade tohto experimentu je však vhodné parametre upraviť.

[1]Dátová sada pasažierov lode Titanic, pozri http://rdatamining.com/data

```
> library(arules))
> rules <- apriori(titanic.raw,
    parameter = list(minlen=2, supp=0.0008, conf=0.8),
    appearance = list(rhs=c("Survived=No", "Survived=Yes"), default="lhs")
```

Minimálna podpora je nastavená na veľmi malú hodnotu. Dôvodom je, že je snaha získať informácie o prežití aj tej najmenšej skupiny pasažierov, v tomto prípade dvojčlennej skupiny (`=ceiling(0,0008*2201)`). Parameter minimálnej dĺžky pravidla je nastavený na hodnotu `2` z dôvodu, aby ľavá strana pravidla nemohla byť prázdna. Okrem argumentu `parameter` je vo volaní funkcie využitý aj argument `appearance`. Ten obmedzuje dolovanie pravidiel po stránke využitia atribútov. V tomto prípade boli pravidlá obmedzené na to, aby sa na pravej strane vyskytoval iba atribút `Survived`.

K výpisu asociačných pravidiel je potrebne využiť funkciu `inspect()`, nakoľko výsledok funkcie `apriori()` túto informáciu neposkytuje. Pred samotným výpisom je ešte obvykle vhodné zoradiť pravidlá podľa zvoleného významnostného parametru.

```
> rules.sorted <- sort(rules, by="lift")
> inspect(rules.sorted)
lhs                                  rhs             support confid lift
{Class=2nd,Age=Child}             => {Survived=Yes} 0.011   1.000  3.096
{Class=1st,Age=Child}             => {Survived=Yes} 0.003   1.000  3.096
{Class=2nd,Sex=Female,Age=Child} => {Survived=Yes} 0.006   1.000  3.096
{Class=2nd,Sex=Male,Age=Child}   => {Survived=Yes} 0.005   1.000  3.096
{Class=1st,Sex=Male,Age=Child}   => {Survived=Yes} 0.002   1.000  3.096
{Class=1st,Sex=Female}           => {Survived=Yes} 0.064   0.972  3.010
{Class=1st,Sex=Female,Age=Adult} => {Survived=Yes} 0.064   0.972  3.010
...
```

Pre výpis podrobnejších štatistik nad pravidlami sa používa funkcia `summary()` .

```
> summary(rules.sorted))
set of 12 rules
 summary of quality measures:
    support           confidence          lift            count
 Min.   :0.0020   Min.   :0.8270   Min.   :1.222   Min.   : 5.0
 Median :0.0360   Median :0.9170   Median :2.716   Median :80.0
 Mean   :0.0506   Mean   :0.9242   Mean   :2.490   Mean   :111.3
 Max.   :0.1920   Max.   :1.0000   Max.   :3.096   Max.   :422.0
 ...
```

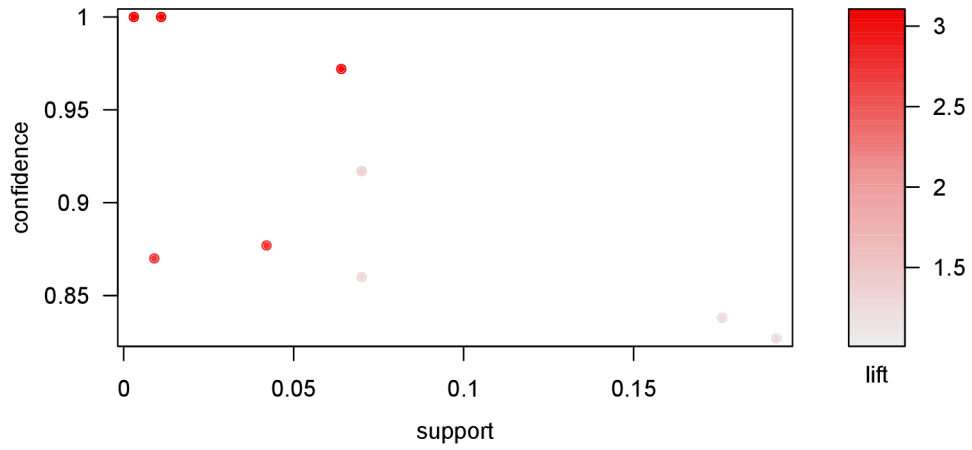
Odstránenie redundantných pravidiel

Z predošlého výpisu funkcie `inspect()` je možné vidieť, že niektoré z pravidiel poskytujú takmer totožnú informáciu. Príkladom takýchto pravidiel sú pravidlá 1 a 3. Všeobecne platí, že pravidlo (pravidlo 3) je považované za redundantné v prípade, keď je špecifickejšie ako iné (pravidlo 1) a nadobúda menšiu hodnotu významnostného parametru `lift`. Redundantné pravidlá je potrebné odstrániť. K nájdeniu redundantných pravidiel je možné využiť funkciu `is.redundant()`.

```
> rules.pruned <- rules.sorted[!is.redundant(rules.sorted)]
> inspect(rules.pruned)
lhs                                   rhs             support confidence lift
{Class=2nd,Age=Child}              => {Survived=Yes} 0.011   1.000      3.096
{Class=1st,Age=Child}              => {Survived=Yes} 0.003   1.000      3.096
{Class=1st,Sex=Female}             => {Survived=Yes} 0.064   0.972      3.010
{Class=2nd,Sex=Female}             => {Survived=Yes} 0.042   0.877      2.716
{Class=Crew,Sex=Female}            => {Survived=Yes} 0.009   0.870      2.692
{Class=2nd,Sex=Male,Age=Adult}     => {Survived=No}  0.070   0.917      1.354
{Class=2nd,Sex=Male}               => {Survived=No}  0.070   0.860      1.271
{Class=3rd,Sex=Male,Age=Adult}     => {Survived=No}  0.176   0.838      1.237
{Class=3rd,Sex=Male}               => {Survived=No}  0.192   0.827      1.222
```

Vizualizácia pravidiel

Po odstránení redundancií je možné finálne asociačné pravidlá vizualizovať pomocou funkcie `plot()` z balíčka `arulesViz`.



Obr. 5.2: Vizualizácia asociačných pravidiel s balíčkom `arulesViz`

Na obrázku 5.2 je graf typu `scatterplot`, funkcia však ponúka ďalšie odlišné typy, ako napríklad `grouped`, `graph`, `matrix`, `paracoord`, ...

Zhodnotenie experimentu

V tomto experimente bolo ukázané dolovanie asociačných pravidiel s balíčkom `arules`, redukcia pravidiel a vizualizácia s balíčkom `arulesViz`. Cieľom bolo zistiť informácie o prežití pasažierov plavby loďou Titanic. Z finálnych asociačných pravidiel a analýzy dátovej sady je možné vyvodiť nasledujúce závery:

- S najvyššou pravdepodobnosťou prežili deti prvej a druhej triedy (prežili všetky).
- S najnižšou pravdepodobnosťou prežili dospelí muži z druhej triedy (prežilo 8,3 %).
- Ženy prežili s vyššou pravdepodobnosťou ako muži (73 % oproti 21 %).

- Deti prežili s vyššou pravdepodobnosťou ako dospelí (52 % oproti 31 %).
- Pravdepodobnosť prežitia na základe cestovnej triedy: prvá trieda (62 %), druhá trieda (41 %), tretia trieda (25 %), posádka (24 %).

5.2 Bayesovská klasifikácia

Nasledujúci experiment je zameraný na ukážku jednoduchej bayesovskej klasifikácie nad dátovou sadou partnerských rozvodov. Dátová sada obsahuje výpovede manželských/rozvedených párov, ktoré sú klasifikované do dvoch tried: výpoveď rozvedeného páru a výpoveď manželského páru. Ide teda o binárnu klasifikáciu.

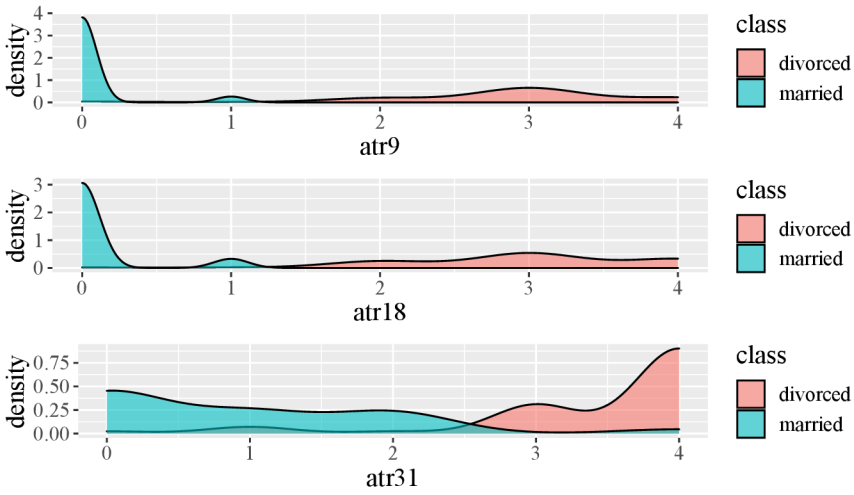
Prvá časť podkapitoly je venovaná podrobnejšiemu popisu a ukážke dátovej sady. Nasleduje popis procesu vytvorenia a využitia klasifikačného modelu. V závere podkapitoly sú zhrnuté výsledky experimentu.

Dátová sada partnerských rozvodov

Dátová sada partnerských rozvodov pozostáva zo 171 záznamov. Každý záznam je reprezentovaný jednou výpoveďou manželského/rozvedeného páru, konkrétne ide o výpovede 84 rozvedených a 86 manželských párov. Pár odpovedal na 54 otázok, akými sú napríklad:

- Užívate si cestovanie s partnerom?
- Máte s partnerom rovnakú predstavu o tom, ako by mal vyzerať život v manželstve?
- Pociťujete pri vymieňaní názorov agresivitu voči partnerovi?

Vizualizácia odpovedí na tieto otázky je uvedená na obrázku 5.3.



Obr. 5.3: Vizualizácia zastúpenia hodnôt atribútov

Všetky zozbierané odpovede nadobúdajú hodnoty v stupnici 0 až 4, kde konkrétna hodnota reprezentuje závažnosť problému pre potencionálny rozvod konkrétneho páru. Dátová sada

neobsahuje žiadne chýbajúce hodnoty. Viac informácií o databáze je možné nájsť v repozitári UCI Machine Leaning Repositary[2].

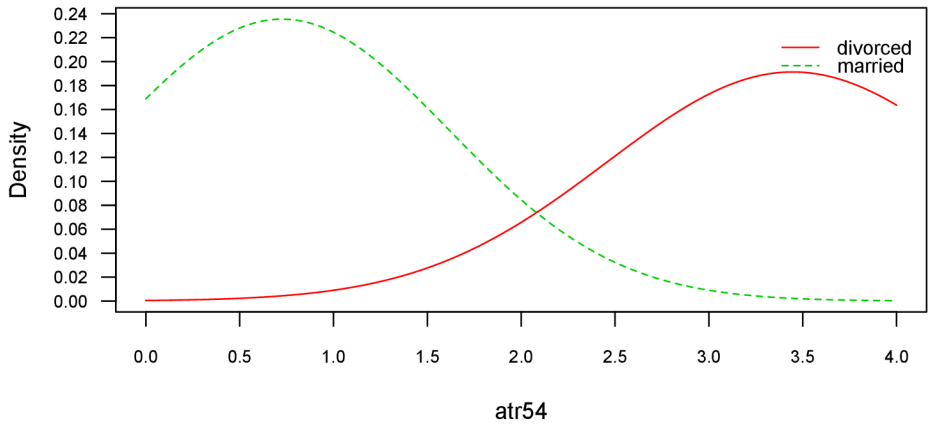
Jednoduchá bayesovská klasifikácia

V tejto sekcii je ukázané vytvorenie a použitie jednoduchého bayesovského klasifikátora. Model je vytvorený nad tréningovou množinou údajov (7/10 záznamov) pomocou funkcie `naive_bayes()` z balíčka `naivebayes`. V prípade preskúmania modelu je možné zistiť viacero informácií, akými sú napr. pravdepodobnosť zaradenia záznamu do jednotlivých tried alebo zastúpenie hodnôt atribútov pre jednotlivé triedy.

```
> divorces.bayes <- naive_bayes(class ~ ., data = divorces.train)
> divorces.bayes
A priori probabilities:
 divorced   married
0.4745763 0.5254237

atr1    divorced   married
  mean 3.1964286 0.4516129
  sd   0.7958904 0.9175416
...
```

Informácia o zastúpení hodnôt atribútov pre jednotlivé triedy je odlišná pre kategorické atribúty a kvantitatívne atribúty. Pre kvantitatívne atribúty s hodnotami spojitého charakteru je uvedená stredná hodnota a smerodajná odchýlka pre danú triedu.



Obr. 5.4: Vizualizácia zastúpenia hodnôt kvantitatívneho atribútu

Zastúpenie kategorických atribútov s hodnotami diskrétneho charakteru je vyjadrené pravdepodobnosťou výskytu hodnoty pre danú triedu. V tejto dátovej sade sa však atribúty tohto typu nevyskytujú. Rozloženie hodnôt atribútov v jednotlivých triedach je možné vizualizovať pomocou funkcie `plot()`. Ukážka je na obrázku 5.4.

[2]Dátová sada partnerských rozvodov, pozri https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set

Použitie modelu, t. j. predikcia je vykonaná s využitím funkcie `predict()`. Na vstupe prijíma testovaciu množinu údajov (3/10 záznamov). Výstupom je vektor tried klasifikovaných záznamov.

```
> divorces.predict <- predict(divorces.bayes, divorces.test)
> divorces.predict
[1] divorced married married divorced divorced divorced married divorced
[9] ...
```

Vektor tried je možné využiť ako vstup pre funkciu `confusionMatrix` z balíčka `caret` a získať tak informácie o presnosti klasifikátora.

```
> confusionMatrix(divorces.predict, divorces.test$class)
Confusion Matrix and Statistics
          Reference
Prediction divorced married
  divorced       27       0
   married        1      24
                Accuracy : 0.9808
...
```

Zhodnotenie experimentu

Cieľom tohto experimentu bolo vytvoriť jednoduchý bayesovský klasifikátor, ktorý dokáže klasifikovať výpovede na výpoveď manželského a výpoveď rozvedeného paru. Z výpovedí vyplynulo, že dôvod rozvodu alebo udržania vzťahu je často podobný. Prediktívny model nad touto dátovou sadou by teda bolo napr. možné využiť pre predpoveď rozvodov a zároveň tak poskytovať pomocnú radu párom, ktoré sa nachádzajú v kritickom období. K vytvoreniu modelu bol využitý balíček `naivebayes`. Z celkového počtu 52 testovaných záznamov bolo presne klasifikovaných 51, z čoho vyplýva, že presnosť jednoduchého bayesovského klasifikátora je 98,08 %. Pre porovnanie je uvedená presnosť ostatných typov modelov:

- Rozhodovací strom s balíčkom `party` – presnosť 96,15 %.
- Náhodný les s balíčkom `randomForest` – presnosť 98,08 %.
- Neurónová sieť so spätným šírením s balíčkom `neuralnet` – presnosť 98,08 %.

Zaujímavosťou tejto dátovej sady je to, že ak by bolo trénovanie modelu vykonané nad atribútom s odpoveďami na otázku, či má pár rovnakú predstavu o živote v manželstve, tak by presnosť klasifikátora nad validačnou množinou vyšla 100 %.

5.3 Klasifikácia pomocou rozhodovacích stromov

V tejto podkapitole je zdokumentovaný experiment prezentujúci vytvorenie a použitie prediktívnych modelov v podobe rozhodovacieho stromu. Vytvorený model pracuje nad databázou absencií v práci, pričom úlohou je správne predikovať dĺžku neprítomnosti zamestnanca.

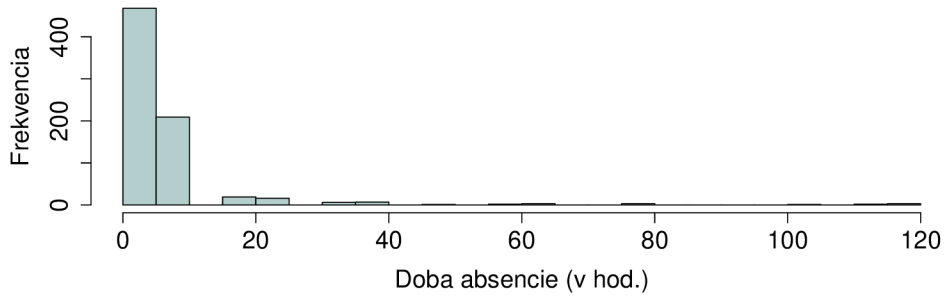
Prvá časť podkapitoly sa venuje popisu, ukážke, analýze a úprave zdrojových dát. Následne je prezentované vytvorenie, použitie a vizualizácia rozhodovacieho stromu s využitím balíčkov `party`, `rpart`, `randomForest`. Koniec podkapitoly je určený pre zhodnotenie modelu a výsledkov experimentu.

Dátová sada absencií v práci

Databáza absencií v práci bola vytvorená zo záznamov Brazílskej kuriérskej spoločnosti. Zber bol vykonaný v období troch rokov, konkrétne od júla 2007 do júla 2010. Za toto obdobie sa vyzbieralo 740 záznamov. Každý záznam reprezentuje neprítomnosť zamestnanca v podobe 21 atribútov:

```
> colnames(absent.data)
[1]  "id"                "reason.for.absence"  "month.of.absence"
[4]  "day.of.the.week"   "seasons"             "transportation.expense"
...
[19] "height"            "body.mass.index"     "absenteeism.time"
```

Jedným z atribútov je aj `absenteeism.time` (dĺžka absencie) vizualizovaný histogramom na obrázku 5.5. Ide o atribút predikovanej triedy, ktorý je potrebné upraviť do podoby vhodnej pre klasifikáciu. Ďalšia potrebná úprava sa týka záznamov, ktoré obsahujú neplatné hodnoty. Proces obidvoch úprav je popísaný v nasledujúcej sekcii.



Obr. 5.5: Histogram dĺžky absencie

Detailnejšie informácie o dátovej sade sú dostupné na webových stránkach UCI Machine Leaning Repositary[3].

Predspracovanie dátovej sady

V tejto sekcii je zdokumentovaný proces odstránenia nedokonalostí zdrojových dát a ich následná príprava pre vytvorenie a použitie klasifikátora.

Už v predošlej sekcii bolo uvedené, že dátová sada obsahuje neplatné hodnoty, ktoré môžeme považovať za chýbajúce. Prítomnosť chýbajúcich hodnôt je možné zistiť s využitím funkcie `anyNA()`. S funkciou `is.na()` je zase možné zistiť zastúpenie chýbajúcich hodnôt pre konkrétny atribút.

```
> anyNA(absent.data)
[1] TRUE
> as.matrix(colSums(is.na(absent.data)))
                        [,1]
reason.for.absence        43
absenteeism.time          44
...
```

[3]Dátová sada absencií v práci, pozri https://archive.ics.uci.edu/ml/datasets/Absenteeism+at+work

Chýbajúce hodnoty je ďalej potrebné odhadnúť a doplniť. K tomuto poslúži balíček `mice`, ktorý obsahuje množstvo zaujímavých funkcií pre prácu s chýbajúcimi hodnotami. Jednou z nich je funkcia `mice()`, ktorej funkcionalita spočíva v odhadovaní chýbajúcich hodnôt. Funkcia na vstupe môže prijímať niekoľko argumentov. V tomto prípade ide o údaje s chýbajúcimi hodnotami a inicializátor generátora náhodných čísel. Výstupom funkcie je implicitne 5 odhadov pre každú chýbajúcu hodnotu. Počet odhadov je možné zmeniť vstupným argumentom `m`.

```
> impute <- mice(absent.data, seed=123)
```

Z výsledných odhadov je potrebné vybrať ten najvhodnejší a následne s pomocou funkcie `complete()` doplniť chýbajúce hodnoty v dátovej sade.

```
> absent.data <- complete(impute, 3)
```

Ďalším problémom zdrojových dát je formát atribútu predikovanej triedy. Dĺžka absencie zamestnanca je vyjadrená v hodinách, to znamená, že ide o numerické hodnoty spojitého charakteru. V prípade predikovania takýchto hodnôt je vhodné namiesto klasifikácie využiť regresné metódy, to však nie je cieľom experimentu. Problému sa dá vyhnúť jednoduchou kategorizáciou hodnôt do 4 tried: poldenná absencia (1 až 4 hod.), celodenná absencia (4 až 8 hod.), absencia do 1 týždňa (8 až 40 hod.), absencia nad 1 týždeň (nad 40 hod.). Kategorizáciu triedy je možné vykonať nasledovne:

```
> more.than.week <- absent.data$absenteeism.time > 40
> absent.data$absenteeism.time[more.than.week] <- "more than week"
```

Nad dátovou sadou bolo vykonaných ešte niekoľko úprav, akými sú napr. odstránenie nepotrebných atribútov a rozdelenie dát na tréningovú a validačnú množinu v pomere 7 ku 3.

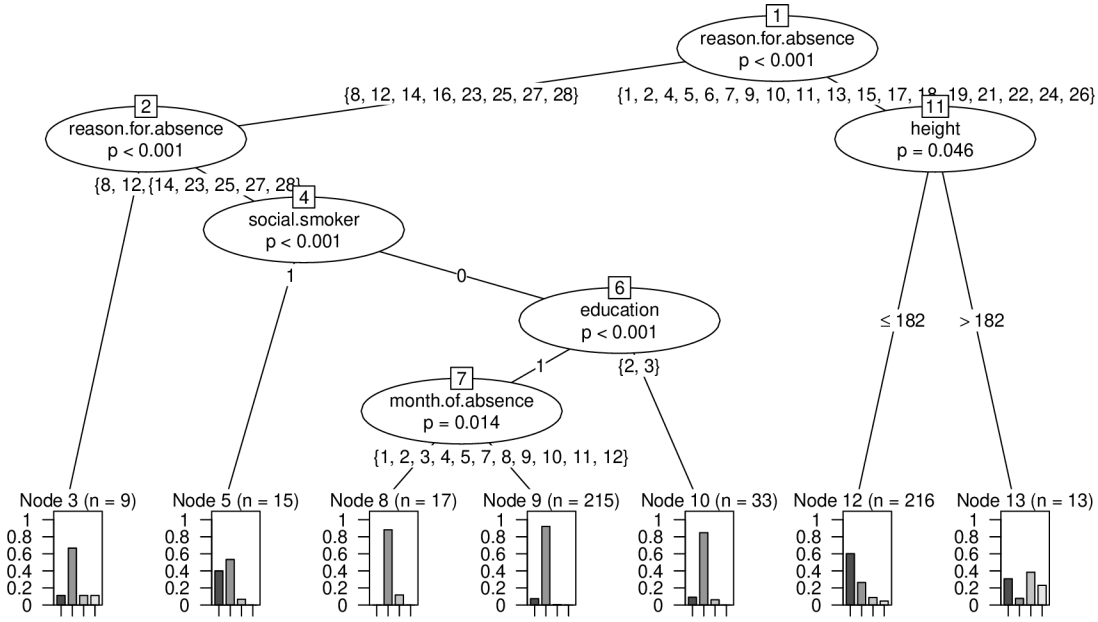
Rozhodovacie stromy s balíčkom `party`

Táto sekcia prezentuje, ako vytvoriť a použiť klasifikačný model v podobe rozhodovacieho stromu s využitím balíčka *party*. Balíček *party* poskytuje k vytvoreniu modelu funkciu `ctree()`. Funkcia na vstupe prijíma minimálne predikčnú formulu a tréningovú množinu údajov.

```
> absent.ctree <- ctree(absenteeism.time ~ ., data = absent.train)
```

Vytvorený model je možné preskúmať vypísaním pravidiel alebo vykreslením rozhodovacieho stromu s pomocou funkcie `plot()`, tak ako je uvedené na obrázku 5.6.

```
> absent.ctree
         Conditional inference tree with 7 terminal nodes
1) reason.for.absence == 8, 12, 14, 16, 23, 25, 27, 28; criterion = 1,
statistic = 318.956
  2) reason.for.absence == 8, 12, 16; criterion = 1, statistic = 80.625
    3)* weights = 9
  2) reason.for.absence == 14, 23, 25, 27, 28
    4) social.smoker == 1; criterion = 0.999, statistic = 43.381
...
> plot(absent.ctree)
```



Obr. 5.6: Vizualizácia rozhodovacieho stromu z balíčka `party`

Klasifikácia je následne vykonaná s funkciou `predict()`. Tá na vstupe prijíma vytvorený klasifikačný model a validačnú množinu údajov. Výstupom je vektor tried pre jednotlivé záznamy, ktorý je možné využiť ako vstup pre funkciu `confusionMatrix()` z balíčka `caret`. Funkcia následne vypíše informácie o presnosti klasifikácie.

```
> absent.ctree.predict <- predict(absent.ctree, absent.test)
> confusionMatrix(absent.ctree.predict, absent.test$absenteeism.time)
Confusion Matrix and Statistics
                Reference
Prediction       full day  half day  less than week  more than week
  full day             51        20              15               1
  half day             13       115               4               0
  less than week        1         1               1               0
  more than week        0         0               0               0
Overall Statistics
               Accuracy : 0.7523
...
```

Rozhodovacie stromy s balíčkom `rpart`

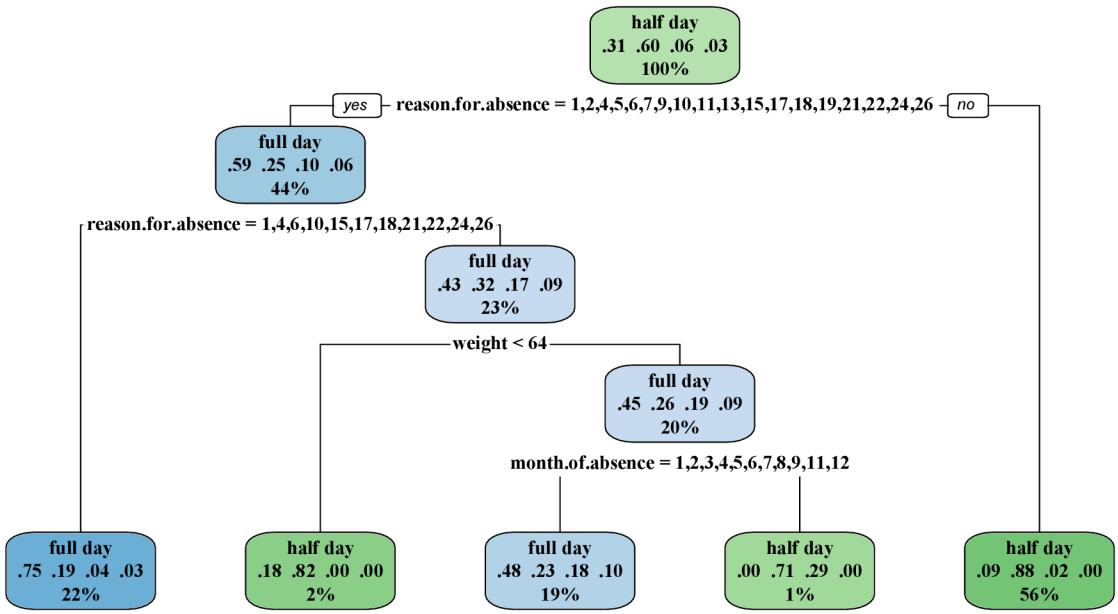
Ďalším balíčkom, ktorý je možné využiť pre tvorbu rozhodovacích stromov, je balíček `rpart`. V tomto balíčku je možné vytvoriť rozhodovací strom pomocou funkcie `rpart()`. Vstupy sú s výnimkou argumentu `control` rovnaké ako v prípade volania funkcie `ctree()`. Argument `control` je využitý pre zmenu tvorby stromu. V tomto prípade ide o zmenu minimálneho počtu záznamov potrebného na rozdelenie uzla.

```
> absent.rpart <- rpart(
    absenteeism.time ~ .,
    data = absent.train,
    control = rpart.control(minsplit = 10))
```

Pri použití balíčka `rpart` je v porovnaní s balíčkom `party` potrebné vykonať jeden krok naviac. Funkcia `rpart()` neposkytuje kontrolu generovania prebytočných vetiev, a tak je ich orezanie nutné vykonať manuálne s využitím funkcie `prune()`. Funkcia na vstupe prijíma strom s prebytočnými vetvami a tzv. parameter zložitosti, ktorý definuje mieru orezania.

```
absent.rpart.pruned <- prune(absent.rpart, cp = minerror.cp)
```

Rozhodovací strom je rovnako ako v balíčku `party` možné preskúmať výpisom pravidiel. Odlišnosť je pri vykreslení, kde je potrebné použiť funkciu `rpart.plot()` z balíčka `rpart.plot`. Výsledok funkcie je uvedený na obrázku 5.7.



Obr. 5.7: Vizualizácia rozhodovacieho stromu z balíčka `rpart`

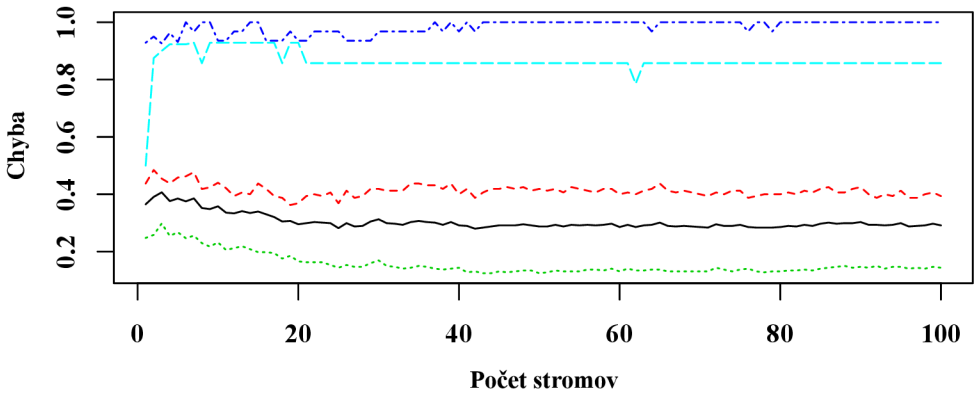
Klasifikáciu a overenie jej presnosti je možné realizovať rovnako ako v prípade balíčka `party`, a to s využitím funkcie `predict()` a funkcie `confusionMatrix()` z balíčka `caret`.

```
> absent.rpart.predict <- predict(absent.rpart.pruned, absent.test)
> confusionMatrix(absent.rpart.predict, absent.test$absenteeism.time)
Confusion Matrix and Statistics
              Reference
Prediction     full day  half day  less than week  more than week
full day             51        19              14               1
half day             14       117               6               0
less than week        0         0               0               0
more than week        0         0               0               0
Overall Statistics
               Accuracy : 0.7568
...
```

Rozhodovacie stromy s balíčkom `randomForest`

V tejto sekcii je prezentovaná klasifikácia pomocou metódy náhodného lesa. Ide o metódu, ktorá je postavená na rozhodovacích stromoch a v jazyku R je reprezentovaná funkciou `randomForest()` z balíčka `randomForest`. Funkcia na vstupe prijíma okrem obvyklej predikčnej formuly a tréningovej množiny údajov aj počet rozhodovacích stromov, ktoré budú les tvoriť. Implicitná hodnota pre tento argument je 500.

```
> absent.rf <- randomForest(absenteeism.time ~ ., data = absent.train,
    ntree = 100)
>absent.rf
...
        OOB estimate of error rate: 30.5 %
Confusion matrix:
               full day half day less than week more than week class.error
full day             95       58              5               2   0.4062500
half day             49      263              1               0   0.1597444
less than week       22        9              0               0   1.0000000
more than week       11        1              0               2   0.8571429
...
```



Obr. 5.8: Chybovosť náhodného lesa z balíčka `randomForest`

V prípade vypísania modelu sú poskytnuté informácie o chybovosti zaradenia záznamov z tréningovej množiny. Chybovosť zaradenia v závislosti na počte použitých stromov je možné vizualizovať pomocou funkcie `plot()`. Výsledok volania funkcie je uvedený na obrázku 5.8.

Samotná klasifikácia je realizovaná obdobne ako v predošlých prípadoch, a to pomocou funkcie `predict()`. Presnosť klasifikácie je následne zistená s využitím funkcie `confusionMatrix()` z balíčka `caret`.

```
> absent.rf.predict <- predict(absent.rf, absent.test)
> confusionMatrix(absent.rf.predict, absent.test$absenteeism.time)
Confusion Matrix and Statistics
              Reference
Prediction     full day  half day  less than week  more than week
full day             47        18              11               1
half day             17       118               6               0
less than week        1         0               3               0
more than week        0         0               0               0
Overall Statistics
               Accuracy : 0.7568
...
```

Zhodnotenie experimentu

Úlohou tohto experimentu bolo demonštrovať vytvorenie a použitie klasifikačných modelov postavených na rozhodovacom strome. V experimente boli postupne prezentované možnosti balíčkov `party`, `rpart` a `randomForest`.

Klasifikácia bola vykonaná nad dátovou sadou absencií v práci, pričom cieľom bolo správnym spôsobom klasifikovať dĺžku absencie zamestnanca. Pred samotnou klasifikáciou bolo potrebné údaje predspracovať. Medzi úpravy patrilo nájdenie a doplnenie nevalidných hodnôt a kategorizácia atribútu dĺžky absencie do 4 tried: poldenná absencia (30 %), celodenná absencia (61 %), absencia menej ako týždeň (7 %), absencia viac ako týždeň (2 %).

Nízky počet záznamov s dlhodobou absenciou zapríčinil, že modely nedokážu klasifikovať objekty do triedy absencia viac ako týždeň. Rozhodovací strom s balíčkom `rpart` naviac nedokáže klasifikovať ani do triedy absencia menej ako týždeň. Presnosť klasifikácie nad validačnou množinou vyšla pri jednotlivých modeloch nasledovne:

- Rozhodovací strom s balíčkom `party` – 75,23 %.
- Rozhodovací strom s balíčkom `rpart` – 75,68 %.
- Náhodný les s balíčkom `randomForest` – 75,68 %.

Atribútom s najväčšou rozhodovacou schopnosťou vyšiel vo všetkým prípadoch atribút dôvodu absencie. Pre porovnanie presnosti je ešte možné uviesť presnosť iných typov modelov:

- Jednoduchá bayesovská klasifikácia s balíčkom `naivebayes` – 59,01 %.
- Neurónová sieť so spätným šírením s balíčkom `neuralnet` – 60,36 %.

5.4 Klasifikácia pomocou neurónových sietí

V tomto experimente sú predstavené postupy vytvorenia a použitia klasifikačného modelu v podobe neurónovej siete. Ako zdroj dát pre experiment je využitá dátová sada mostov v meste Pittsburgh. Cieľom je predikovať 5 atribútov popisujúcich dizajn mostu.

Úvod podkapitoly je zameraný na detailnejšie predstavenie zdrojových dát a ich následné predspracovanie do podoby vhodnej pre klasifikáciu. Jadrom podkapitoly je vytvorenie a použitie neurónovej siete pre úlohu klasifikácie. V závere podkapitoly sú zhrnuté výsledky experimentu a záverečné poznatky.

Dátová sada mostov Pittsburghu

Mesto Pittsburgh je vo svete preslávene veľkou pestrosťou najrôznejších typov mostov. Táto dátová sada ich obsahuje celkovo 108. Každý most je reprezentovaným konkrétnym záznamom v podobe 13 atribútov. Atribúty sú rozdelené do dvoch skupín:

- popisné atribúty, na základe ktorých sa bude predikovať,

```
> colnames(bridges.data)[2:8]
[1] "river" "location" "erected" "purpose" "length" "lanes" "clear"
```

- predikované atribúty.

```
> colnames(bridges.data)[9:13]
[1] "t.or.d" "material" "span" "rel.l" "type"
```

S databázou mostov sa však spája niekoľko problémov, ktoré je potrebné eliminovať. Ide napríklad o prítomnosť záznamov s chýbajúcimi hodnotami. Riešenie tohto, ale aj ďalších problémov je podrobnejšie rozobrané v ďalšej sekcii. Viac informácií o dátovej sade je dostupných na webovej stránke UCI Machine Learning Repositary[4].

Predspracovanie zdrojových dát

Tak ako bolo spomenuté v úvode podkapitoly, dátová sada obsahuje niekoľko nedokonalostí. V tejto sekcii je popísaný proces odstránenia týchto nedokonalostí a následná úprava zdrojových dát do podoby vhodnej pre klasifikáciu pomocou neurónovej siete.

Prvou a najzásadnejšou úpravou, ktorú je potrebné vykonať, je identifikácia a doplnenie chýbajúcich hodnôt. Tento problém je riešený rovnako ako v experimente z predošlej podkapitoly 5.3. Pre identifikáciu chýbajúcich hodnôt sú využité funkcie `anyNa()` a `is.na()`, pre doplnenie funkcie `mice()` a `complete()` z balíčka `mice`.

Dátová sada je po predošlom kroku kompletná, ďalšie úpravy sa týkajú existujúcich hodnôt. Pre zrýchlenie učenia neurónovej siete je vhodné vstupné hodnoty normalizovať. V rámci tohto experimentu budú normalizované do intervalu 0 až 1. Pred samotnou normalizáciou je však potrebné zmeniť kategorické atribúty na kvantitatívne tak, ako je demonštrované na trojhodnotovom atribúte `river`.

```
> bridges.data$river[bridges.data$river == "A"] <- 0
> bridges.data$river[bridges.data$river == "M"] <- 1
> bridges.data$river[bridges.data$river == "O"] <- 2
> bridges.data$river <- as.numeric(bridges.data$river)
```

[4]Dátová sada mostov Pittsburghu, pozri https://archive.ics.uci.edu/ml/datasets/Pittsburgh+Bridges

Normalizácia je vykonaná aplikovaním jednoduchej funkcie na hodnoty popisných atribútov.

```
> bridges.data <- as.data.frame(apply(
    bridges.data[, 2:8], 2,
    function(x) (x - min(x))/(max(x)-min(x))))
```

Posledným krokom v procese predpracovania je rozdelenie zdrojových dát na tréningovú a validačnú množinu. Pomer záznamov medzi týmito dvoma množinami je 7 ku 3.

Neurónové siete

Po príprave zdrojových dát je možné pristúpiť k vytvoreniu klasifikačného modelu v podobe neurónovej siete. K vytvoreniu je využitá funkcia `neuralnet()` z balíčka `neuralnet`.

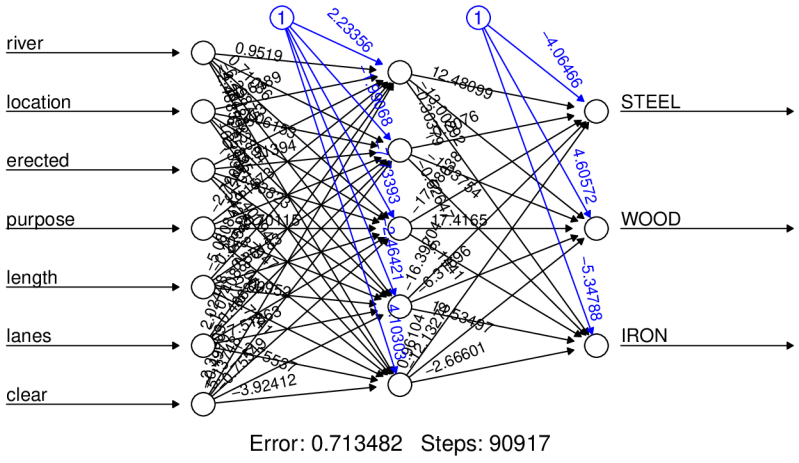
Funkcia na vstupe prijíma niekoľko argumentov. Prvým argumentom je formula, ktorá udáva, nad akými atribútmi bude vykonaná predikcia. Predikovaným atribútom je atribút `material`, volanie funkcie by však vyzeralo obdobne aj pre ostatné 4 predikované atribúty. Ďalším vstupným argumentom je samotný dátový zdroj. Vstupný argument `hidden` definuje, koľko neurónov bude použitých v skrytej vrstve. Hodnota 5 bola zvolená na základe predpisu (5.1):

$$(m + n)/2 \tag{5.1}$$

kde m je počet neurónov vo vstupnej vrstve a n je počet neurónov vo výstupnej vrstve. Na vytvorenie siete je použitý algoritmus spätného šírenia s krokom učenia 0,01. Vstupný argument `linear.output = FALSE` určuje, že nepôjde o neurónovú sieť zameranú na regresiu, ale na klasifikáciu.

```
> bridges.neuralnet <- neuralnet(
    material~river+location+erected+purpose+length+lanes+clear,
    data = bridges.train, hidden = 5, algorithm = "backprop",
    learningrate = 0.01, linear.output = FALSE)
```

Vytvorenú neurónovú sieť je možné vizualizovať použitím funkcie `plot()`. Výsledná vizualizácia je zobrazená na obrázku 5.9.



Obr. 5.9: Vizualizácia neurónovej siete z balíčka `neuralnet`

Klasifikácia je následne vykonaná s funkciou `compute()`, ktorá na vstupe prijíma model v podobe neurónovej siete a validačnú množinu údajov.

```
> bridges.predict <- compute(bridges.neuralnet, bridges.test[,1:7])
```

Výstupom je matica zastúpenia tried pre každý záznam. Matica je následne upravená na vektor tried, ktorý je použitý na zistenie presnosti modelu.

```
> confusionMatrix(as.factor(bridges.predict), bridges.test$material)
Confusion Matrix and Statistics
          Reference
Prediction IRON STEEL WOOD
IRON          0     1    0
STEEL         2    21    1
WOOD          2     0    6
Overall Statistics
               Accuracy : 0.8182
...
```

Zhodnotenie experimentu

V tomto experimente bolo demonštrované vytvorenie a použitie klasifikačného modelu v podobe neurónovej siete so spätným šírením. K tomuto účelu poslúžil balíček `neuralnet`.

Klasifikácia bola realizovaná nad údajmi o mostoch v Pittsburghu, pričom cieľom bolo na základe 7 popisných atribútov predikovať ďalších 5 vlastností mostu. Databáza vyžadovala niekoľko úprav. Prvou úpravou bolo doplnenie chýbajúcich hodnôt s balíčkom `mice`. Po kompletizácii dátovej sady nasledovala normalizácia hodnôt do intervalu 0 až 1.

Klasifikácia bola negatívne ovplyvnená nevyváženosťou a nízkym počtom zdrojových dát. Dátová sada obsahovala iba 108 záznamov, pričom zastúpenie jednotlivých tried nebolo rovnomerné. To malo za následok, že modely ignorovali niektoré triedy s nízkym počtom záznamov. Presnosť vyšla pre jednotlivé atribúty nasledovne:

- Atribút `t.or.d` (vrchná/spodná podpera) – binárna klasifikácia s presnosťou 90,91 %.
- Atribút `material` (materiál) – klasifikácia do 3 tried s presnosťou 81,82 %.
- Atribút `span` (rozpätie) – klasifikácia do 3 tried s presnosťou 66,67 %.
- Atribút `rel.l` – klasifikácia do 3 tried s presnosťou 75,76 %.
- Atribút `type` (typ) – klasifikácia do 7 tried s presnosťou 27,27 %.

Pre porovnanie presnosti je možné nad atribútom `material` uviesť presnosť ostatných typov modelov:

- Jednoduchá bayesovská klasifikácia s balíčkom `naivebayes` – 84,85 %.
- Rozhodovací strom s balíčkom `party` – 84,85 %.
- Náhodný les s balíčkom `randomForest` – 87,88 %.

5.5 Zhlukovanie

Posledný experiment je zameraný na ukážku zhlukovania založeného na rozdeľovaní, hierarchického zhlukovania a zhlukovania založeného na hustote. Ako zdroj dát pre zhlukovanie je vybraná databáza semien pšenice. Semená sú zhlukované na základe 7 popisných atribútov.

Detailnejšia špecifikácia dátovej sady a jej položiek je uvedená v prvej časti podkapitoly. Po nej nasleduje ukážka jednotlivých techník zhlukovania. Záver podkapitoly je venovaný porovnaniu a zhodnoteniu výsledkov experimentu.

Dátová sada semien pšenice

Dátová sada semien pšenice bola vytvorená v rámci výskumu na agrofyzikálnom ústave Poľskej akadémie vied v Lubline. Vizualizácia vnútornej štruktúry semena bola realizovaná pomocou röntgenových doštičiek a techniky soft X-ray.

Databáza obsahuje celkovo 210 záznamov. Jednotlivé záznamy reprezentujú konkrétne semeno pšenice v podobe 8 atribútov: plocha, obvod, kompaktnosť, dĺžka, šírka, koeficient asymetrie, dĺžka drážky, typ semena.

```
> head(seeds.data, n=5)
   area perim compact kern.len kern.wid asym.coef kern.gro.len seed
1 15.26 14.84  0.8710    5.763    3.312     2.221        5.220    1
2 14.88 14.57  0.8811    5.554    3.333     1.018        4.956    1
3 14.29 14.09  0.9050    5.291    3.337     2.699        4.825    1
4 13.84 13.94  0.8955    5.324    3.379     2.259        4.805    1
5 16.14 14.99  0.9034    5.658    3.562     1.355        5.715    1
```

Atribút `seed` (typ semena) nadobúda 3 hodnoty: Kama, Rosa a Canadian. Každé z týchto semien sa v databáze vyskytuje 70-krát. V rámci experimentu zhlukovania je tento atribút potrebné odstrániť. Podrobnejší popis dátovej sady je možné nájsť v repozitári UCI Machine Leaning Repositary[5].

Predspracovanie zdrojových dát

Databáza neobsahuje žiadne nevalidné a chýbajúce hodnoty. V prípade zhlukovania je však nutné odstrániť atribút cieľovej triedy.

```
> seeds.features <- seeds.data
> seeds.features$seed <- NULL
```

Pre zlepšenie výsledku zhlukovania je ďalej vhodné údaje normalizovať. Výsledkom normalizácie je, že pri počítaní vzdialeností medzi objektami a následnom vytváraní zhlukov má každý atribút rovnakú váhu. Normalizácia je vykonaná rovnako ako v predošlom experimente, atribúty nadobúdajú hodnoty z intervalu 0 až 1.

```
> seeds.features <- as.data.frame(apply(
    seeds.features,
    2,
    function(x) (x - min(x))/(max(x)-min(x))))
```

[5] Dátová sada semien pšenice, pozri https://archive.ics.uci.edu/ml/datasets/seeds

Zhlukovanie založené na rozdeľovaní

Najbežnejšou technikou zhlukovania založenom na rozdeľovaní je zhlukovanie k-means. Táto technika je v jazyku R reprezentovaná funkciou `kmeans()` z balíčka `stats`. Funkcia `kmeans()` prijíma na vstupe minimálne dátovú sadu a počet cieľových zhlukov, ktorý je potrebné vopred zistiť.

V jazyku R je ideálny počet cieľových zhlukov možné zistiť práve s využitím funkcie `kmeans()`. Výsledok funkcie obsahuje atribút `withins`, ktorý informuje o tom, ako blízko sú k sebe objekty v zhluku. Čím menšia je táto hodnota, tým sú objekty bližšie k sebe. Súčet týchto hodnôt pre každý zhluk je možné interpretovať ako nepresnosť zhlukovania.

```
for (i in 1:20) wcss[i] <- sum(kmeans(seeds.features,centers = i)$withins)
```

Nepresnosť zhlukovania je otestovaná pre 1 až 20 zhlukov, pričom výsledok je vizualizovaný na obrázku 5.10. Optimálny počet cieľových zhlukov je vybraný ako hodnota 3, nakoľko ide o hraničnú hodnotu, od ktorej sa nepresnosť zhlukovania takmer vôbec neznižuje.

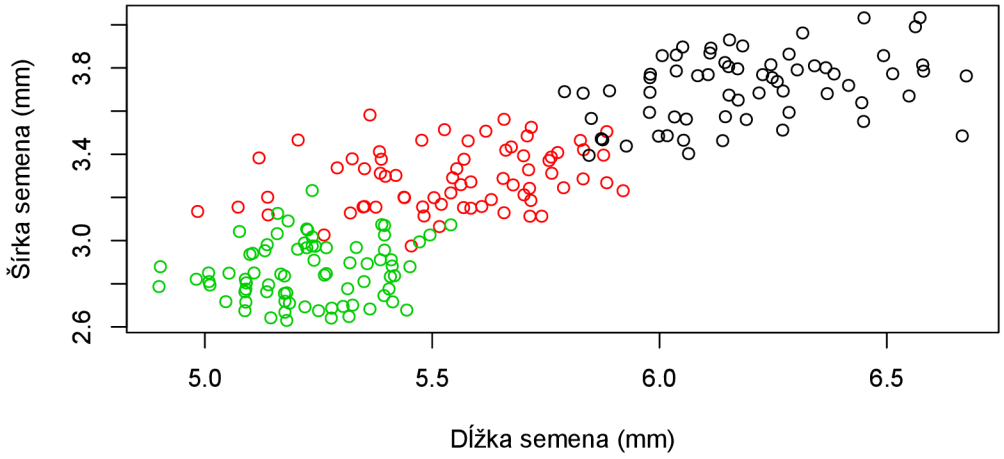
Obr. 5.10: Nepresnosť zhlukovania v závislosti na počte zhlukov

Po výbere výsledného počtu zhlukov je možné pristúpiť k samotnému zhlukovaniu.

```
> seeds.kmeans <- kmeans(seeds.features, 3)
> seeds.kmeans
K-means clustering with 3 clusters of sizes 82, 67, 61
Cluster means:
        area    perim   compact kern.len kern.wid asym.coef kern.gro.len
  1 0.757333  0.793743 0.694192 0.730038 0.769501  0.367577     0.757093
  2 0.383490  0.419841 0.671204 0.364685 0.468499  0.264177     0.318384
  3 0.123334  0.175137 0.378179 0.186710 0.162527  0.498569     0.279288
Clustering vector:
   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 1 2 2 1 1 2 2 2 2 2 2
...
```

Z výpisu vidno, že výsledok funkcie poskytuje množstvo komplexných informácií, akými sú napr. počet objektov v zhlukoch, stredové hodnoty atribútov pre zhluky alebo vektor

zaradenia objektov. Výsledné zhluky je možné vizualizovať na atribútoch dĺžky a šírky semena, ukážka na obrázku 5.11.



Obr. 5.11: Výsledok zhlukovania k-means

Správnosť zhlukovania je možné overiť s využitím pôvodných hodnôt atribútu `seed` a funkcie `table()`.

```
> table(seeds.data$seed, seeds.kmeans$cluster)
          1  2  3
canadian 67  3  0
kama     10 58  2
rosa      0  8 62
```

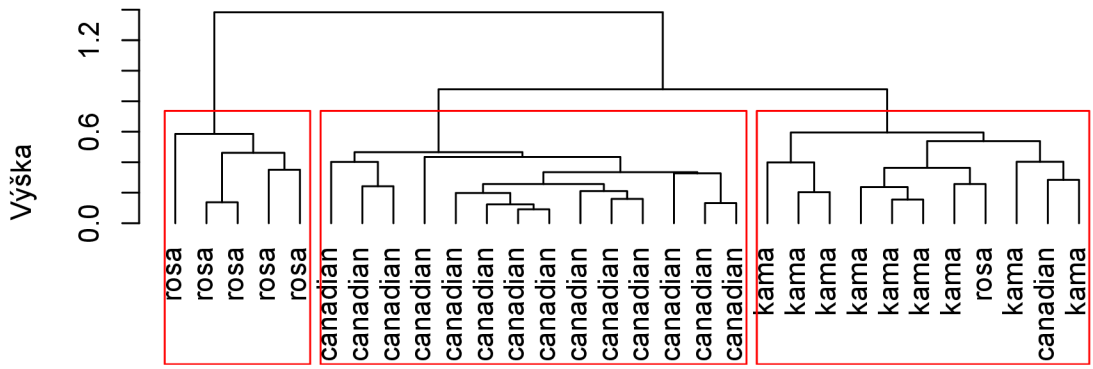
Výpis informuje o tom, že jednotlivé semená pšenice majú v istej miere podobné vlastnosti. Jedinou výnimkou je dvojica typu Canadian a Rosa. Ich vlastnosti podľa výsledných zhlukov nemajú žiadne prekrytie. K zisku podrobnejších informácií o výsledku zhlukovania je možné použiť funkciu `cluster.stats()` z balíčka `fpc`.

Hierarchické zhlukovanie

V tejto sekcii je demonštrované hierarchické zhlukovanie s využitím funkcie `hclust()` z balíčka `stats`. Funkcia v rámci tohto experimentu prijíma na vstupe 2 argumenty, a to vzájomné euklidovské vzdialenosti záznamov a kritérium spájania `average`. Toto kritérium definuje, že vzdialenosť medzi dvoma zhlukmi je vypočítaná ako priemer vzdialeností medzi objektami jedného a druhého zhluku.

```
> seeds.hclust <- hclust(dist(seeds.features), method="average")
```

Vytvorenú hierarchiu zhlukov je možné vizualizovať v podobe dendrogramu s využitím funkcie `plot()`. Z dôvodu prehľadnosti vykreslenia je vybraná náhodná vzorka 30 zástupných záznamov. Ukážka je uvedená obrázku 5.12.



Obr. 5.12: Dendrogram zhlukov

Správnosť zhlukovania je opäť možné overiť pomocou funkcie `table()`. Pred overením je však potrebné stanoviť hierarchii zhlukov pevný počet 3. Túto činnosť zaisťuje funkcia `cutree()`.

```
> seeds.result <- cutree(seeds.hclust.com, 3)
> table(seeds.data$seed, seeds.result)
          1  2  3
canadian 68 15  0
kama      2 52  2
rosa      0  3 68
```

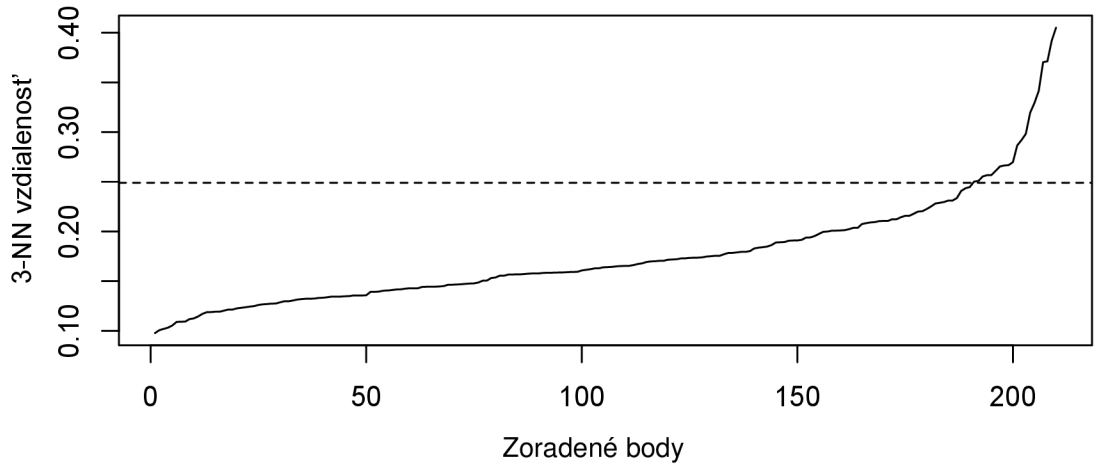
Z výsledku je rovnako ako pri zhlukovaní k-means vidno, že semená s výnimkou dvojice typu Canadian a Rosa majú do istej miery podobné vlastnosti. K zisku podrobnejších informácií o výsledku zhlukovania je možné použiť funkciu `cluster.stats()` z balíčka `fpc`.

Zhlukovanie založené na hustote

Posledným typom zhlukovania, ktoré je ukázané v tejto podkapitole je zhlukovanie založené na hustote. Typickým predstaviteľom tejto skupiny zhlukovania je metóda DBSCAN. V jazyku R je reprezentovaná funkciou `dbscan()` z balíčka `dbscan`.

Rozdiel tejto metódy oproti metódam z predošlých sekcií spočíva v tom, že na vstupe je nutné zadať namiesto počtu zhlukov dva argumenty: `eps` (veľkosť okolia) a `minPts` (minimálny počet bodov v okolí). Tieto argumenty je potrebné vopred zistiť.

K zisteniu optimálnej hodnoty argumentov je možné v balíčku `dbscan` využiť funkciu `kNNdistplot()`, ktorá počíta vzdialenosť k-najbližších susedov v matici bodov. Funkcia na vstupe prijíma dátovú sadu semien a počet k-najbližších susedov. Hodnota k je nastavená na vopred vybranú hodnotu argumentu `minPts`. Výstupom funkcie je graf uvedený na obrázku 5.13.



Obr. 5.13: Výstup funkcie `kNNdistplot()`

Optimálna hodnota `eps` je stanovená na základe miesta v grafe, kde vzdialenosť k-najbližších susedov začína prudko rásť. V rámci tohto experimentu však funkcia `kNNdistplot()` vracia hodnoty, ktorých použitie je pri zhlukovaní neuspokojivé. K zisteniu hodnôt je teda potrebné použiť menej sofistikovaný postup, a to opakované volanie funkcie `dbscan()`. Výsledné volanie s optimálnymi hodnotami argumentov `eps` a `minPts` je nasledovné:

```
> seeds.dbscan <- dbscan(seeds.features, eps=0.262, minPts=22)
```

Správnosť výsledkov je možné overiť s využitím pôvodných hodnôt atribútu `seed` a funkcie `table()`.

```
> table(seeds.data$seed, seeds.dbscan$cluster)
          0  1  2  3
canadian  8 61  1  0
kama     19  2 49  0
rosa     35  0  1 34
```

Z výpisu je vidno, že výsledný model obsahuje 4 zhluky, pričom nultý zhluk je chápaný ako šum. Výpis ďalej informuje o tom, že semená, rovnako ako v prípade predošlých metód, majú s výnimkou dvojice typu Candian a Rosa do istej miery podobné vlastnosti. K zisku podrobnejších informácií o výsledku zhlukovania je možné použiť funkciu `cluster.stats()` z balíčka `fpc`.

Zhodnotenie experimentu

Cieľom posledného experimentu bolo prezentovať rôzne typy zhlukovania v jazyku R, konkrétne zhlukovanie založené na rozdeľovaní, hierarchické zhlukovanie a zhlukovanie založené na hustote. K tomuto účelu boli využité balíčky `stats` a `dbscan`.

Zhlukovanie bolo realizované nad dátovou sadou semien pšenice. Nad týmito údajmi bolo potrebné vykonať dve úpravy. Prvou bolo odstránenie atribútu triedy semena, druhou normalizácia hodnôt s cieľom priradiť pri počítaní euklidovskej vzdialenosti objektov a následnom zhlukovaní každému atribútu rovnakú váhu.

Z výsledkov všetkých typov zhlukovania vzišlo, že jednotlivé druhy semien pšenice sú z pohľadu vlastností mierne podobné. Výnimkou je dvojica druhu Canadian a Rosa, ktoré nemali žiadne prekrytie. K porovnaniu metód zhlukovania sú uvedené tri metriky: porovnanie zaradenia s referenčnou triedou semena a atribúty `within.cluster.ss`, `avg.silwidth` z výstupu funkcie `cluster.stats()`.

- Zhlukovanie k-means:
  - 187 z 210 (89,05 %) záznamov bolo zaradených správne,
  - `within.cluster.ss` nadobúda hodnotu 22,0244,
  - `avg.silwidth` nadobúda hodnotu 0,4221.
- Hierarchické zhlukovanie:
  - 188 z 210 (89,52 %) záznamov bolo zaradených správne,
  - `within.cluster.ss` nadobúda hodnotu 23,7921,
  - `avg.silwidth` nadobúda hodnotu 0,3944.
- Zhlukovanie založené na hustote:
  - 144 z 210 (68,57 %) záznamov bolo zaradených správne,
  - `within.cluster.ss` nadobúda hodnotu 36,6882,
  - `avg.silwidth` nadobúda hodnotu 0,2423.

Atribút `within.cluster.ss` informuje o tom, ako blízko sú pri sebe objekty v zhlukoch. Čím je hodnota menšia, tým sú k sebe objekty bližšie a zhlukovanie je presnejšie. Atribút `avg.silwidth` informuje o tom, ako blízko sú pri sebe objekty v zhlukoch a ako veľmi sú od seba zhluky oddelené. Atribút nadobúda hodnoty 0 až 1, pričom hodnoty bližšie k 1 znamenajú presnejšie zhlukovanie.

Kapitola 6

Zhodnotenie jazyka R

Táto kapitola je zameraná na zhrnutie a zhodnotenie vlastností jazyka R. Cieľom je sprostredkovať dôvody, prečo je vhodné alebo naopak nevhodné používať jazyk R pre úlohy z oblasti získavania znalostí z databáz. V prvej časti kapitoly sú diskutované výhody, v druhej zase nevýhody.

6.1 Výhody

Medzi hlavné výhody jazyka R v oblasti získavania znalostí z databáz patria:

- **Otvorenosť** – Jazyk R je softvér s otvoreným zdrojovým kódom pod licenciou GNU GPL. To znamená, že ide o verejný a slobodný softvér, ktorý je dostupný bezplatne a môže ho používať ktokoľvek, kdekoľvek a kedykoľvek.
- **Popularita** – Jazyk R patrí medzi najvyužívanejšie jazyky. Tak ako bolo spomenuté v kapitole 4.1, komunita TIOBE ho ohodnotila vo februári 2020 ako trinásty najpopulárnejší. Vysoká popularita má za následok širokú online podporu v rámci riešenia vzniknutých problémov a štúdia jazyka.
- **Rozšírenia** – V jazyku R existuje množstvo najrôznejších balíčkov. Momentálne sa ich v online repozitári CRAN nachádza presne 15462. Ich počet a kvalita sú veľkým prínosom pre rôzne odvetvia vedy a priemyslu.
- **Rast** – Jazyk R nepatrí medzi dosluhujúce jazyky. Na jeho vývoji sa podieľa skupina programátorov, ktorá aj v dnešnej dobe zabezpečuje integráciu rôznych noviniek. Okrem vývojárskej skupiny prispievajú k rastu v podobe balíčkov aj samotní používatelia.
- **Kompatibilnosť** – Jazyk R umožňuje integráciu s mnohými inými programovacími jazykmi, akými sú napr. C, C++, Java alebo Python. Rovnako podporuje rozličné formáty zdrojových dát a kompatibilitu s množstvo databázových systémov.
- **Nezávislosť** – Jazyk R je multiplatformový jazyk, ktorý môže bežať v akomkoľvek operačnom systéme: Windows, Linux alebo Mac. Je nezávislý od platformy.
- **Efektívnosť programovania** – Rýchle naprogramovanie požadovanej funkcionality je jednou z najväčších predností jazyka R. Jazyk obsahuje množstvo vysoko abstraktných funkcií, ktorými je možné v krátkom úseku kódu vykonať častokrát aj veľmi zložité výpočty.

- **Podpora práce s údajmi** – Jazyk R podporuje vykonávanie operácií s vektormi, poliami, maticami a rôznymi ďalšími dátovými objektmi rôznej veľkosti. Ďalej zabezpečuje nástroje pre čistenie údajov a opravu poškodených záznamov. Takisto ponúka konštrukcie a balíčky, ktoré umožňujú transformovať neštrukturované a chaotické údaje do štruktúrovanej podoby.
- **Podpora dátovej analýzy** – Primárnym účelom jazyka R je poskytnúť rozsiahlu funkcionalitu v oblasti štatistiky a dátovej analýzy. Logicky teda ide o jednu z hlavných predností tohto jazyka.
- **Podpora vizualizácie** – Jazyk R spolu s balíčkami, akými sú napríklad `ggplot2` alebo `plotly`, poskytuje príkladnú podporu pre vizualizaciu údajov v podobe rôznych grafov.
- **Podpora interpretácie** – Jazyk R obsahuje množstvo balíčkov určených na interpretáciu výsledkov v rôznych formátoch, akými sú napr. interaktívne webové aplikácie.
- **Pracovné prostredia** – Nad jazykom R existuje množstvo vývojových prostredí a doplnkov do vývojových prostredí, ktoré pre programátorov uľahčujú a zefektívňujú prácu. Najvyužívanejším je RStudio popísané v podkapitole 4.2.

6.2 Nevýhody

Najväčšími nevýhodami jazyka R v oblasti získavania znalostí z databáz sú:

- **Zložitosť** – Jazyk patrí medzi náročnejšie jazyky. Jeho učenie môže prebiehať veľmi pomaly, a tak nie je vhodný pre začínajúcich programátorov.
- **Bezpečnosť** – Jazyk R v porovnaní s konkurenčnými jazykmi nemá zabudovanú bezpečnosť. V tomto smere tak so sebou prináša množstvo obmedzení, akými je napríklad nevhodnosť použitia vo webových aplikáciach. [11]
- **Rýchlosť** – Jazyk R je omnoho pomalší v porovnaní s konkurenčnými jazykmi. [11]
- **Rozloženie funkcionality** – V jazyku R je veľká časť funkcionality rozdelená do množstva balíčkov, čo v prípade ich neznalosti spôsobuje problémy pri programovaní.
- **Záruka kvality** – V online repozitári CRAN je možné nájsť množstvo nekvalitných a redundantných balíčkov.
- **Správa pamäte**– Jazyk R v porovnaní s konkurenčnými jazykmi využíva viac pamäte. Objekty sa ukladajú vo fyzickej pamäti, pričom ju môžu v prípade práce s veľkou databázou celú vyčerpať. [11]

Kapitola 7

Záver

Táto práca sa zaoberala problematikou získavania znalostí z databáz v jazyku R. Jej cieľom bolo implementovať experimenty, ktoré demonštrujú poskytovanú podporu jazyka pre túto oblasť. K dosiahnutiu cieľu bolo potrebné vybrať vhodné dátové sady pre daný typ dolovacej úlohy. Pozornosť bola venovaná úlohám klasifikácie, zhlukovaniu a dolovaniu asociačných pravidiel.

Prvý experiment bol zameraný na dolovanie asociačných pravidiel nad prežitím pasažierov plavby lode Titanic. Nasledoval experiment jednoduchej bayesovskej klasifikácie výpovedí manželských a rozvedených párov s výslednou presnosťou 98,08 %. Ďalším experimentom bola klasifikácia dĺžky absencie zamestnancov pomocou rozhodovacích stromov a metódy náhodného lesa. Priemerná presnosť modelov v tomto prípade vyšla 75,53 %. V experimente klasifikácie dizajnových prvkov mostov pomocou neurónových sietí vyšla priemerná presnosť vytvorených modelov na 68,49 %. Posledný experiment bol zameraný na zhlukovanie semien pšenice.

Okrem samotnej implementácie experimentov boli v rámci tejto práce objasnené základné myšlienky, princípy, postupy a metódy, ktoré so ziskom znalostí a jazykom R súvisia. Výstupom práce bolo zhodnotenie jazyka a jeho podpory pre uvedené dolovacie úlohy.

V priebehu práce som rozšíril svoje poznatky v oblasti získavania znalostí z databáz, osvojil si prácu s jazykom R, jeho balíčkami a integrovaným vývojovým prostredím RStudio. V práci však bol popísaný a implementovaný iba zlomok z toho, čo jazyk R poskytuje. Prácu by teda bolo možné v budúcnosti rozšíriť o ukážku ďalších dolovacích úloh a činností spojených s procesom zisku znalostí. Ďalším prípadným rozšírením by mohlo byť porovnanie podpory jazyka R s konkurenčnými jazykmi.

Literatúra

[1] *The R Project for Statistical Computing* [online]. The R Foundation, 2020 [cit. 2020-03-15]. Dostupné z: https://www.r-project.org/about.html.

[2] Bharati, M. a Ramageri, M. Data mining techniques and applications. *Indian Journal of Computer Science and Engineering*. Citeseer. 2010, zv. 1, č. 4, s. 301–305. ISSN 0976-5166.

[3] Dunham, M. H. *Data mining: Introductory and advanced topics*. 1. vyd. New Jersey: Pearson Education PTR, 2006. ISBN 0-13-088892-3.

[4] Gullo, F. From patterns in data to knowledge discovery: what data mining can do. *Physics Procedia*. Elsevier. 2015, č. 62, s. 18–22. ISSN 1875-3892.

[5] Gupta, B. a Garg, D. FP-tree based algorithms analysis: FPGrowth, COFI-Tree and CT-PRO. *International Journal on Computer Science and Engineering*. Engg Journals Publications. 2011, zv. 3, č. 7, s. 2691–2699.

[6] Han, J., Pei, J. a Kamber, M. *Data mining: concepts and techniques*. 3. vyd. Waltham: Morgan Kaufmann Publishers, 2012. ISBN 978-0-12-381479-1.

[7] Hanna, M. Data mining in the e-learning domain. *Campus-wide information systems*. Emerald Group Publishing Limited. 2004, zv. 21, č. 1, s. 29–34. ISSN 1065-0741.

[8] Hornik, K. et al. *Frequently Asked Questions on R* [online]. Február 2020 [cit. 2020-03-14]. Dostupné z: https://cran.r-project.org/doc/FAQ/R-FAQ.html.

[9] Kantardzic, M. *Data mining: concepts, models, methods, and algorithms*. 2. vyd. New Jersey: John Wiley & Sons, 2011. ISBN 978-0-470-89045-5.

[10] Kaur, M. ECLAT Algorithm for Frequent Itemsets Generation. *International Journal of Computer Systems*. 2014, zv. 1, č. 3, s. 82–84. ISSN 1065-0741.

[11] Krill, P. *Why R? The Pros and Cons of the R Language* [online]. Jún 2015 [cit. 2020-03-27]. Dostupné z: https://www.infoworld.com/article/2940864/r-programming-language-statistical-data-analysis.html.

[12] Paralič, J. *Objavovanie znalostí v databázach*. Vedecký spis. Košice: Fakulta elektrotechniky a informatiky TU, 2003. Dostupné z: https://spu.fem.uniag.sk/cvicenia/ksov/zach/Data%20mining/Literatura/ObjavovanieZnalostivDB.pdf.

[13] Peng, R. D. *R programming for data science* [online]. September 2019 [cit. 2020-03-31]. E–kniha v procese tvorby. Dostupné z: https://bookdown.org/rdpeng/rprogdatascience.

[14] Ramakrishnan, R. a Gehrke, J. *Database management systems.* 3. vyd. New York: McGraw Hill, 2003. ISBN 0-07-246563-8.

[15] Torgo, L. *Data mining with R: learning with case studies.* 1. vyd. Minneapolis: Chapman & Hall/CRC press, 2011. ISBN 978-1-4398-1018-7.

[16] wrathematics. *How Much of R is Written in R?* [online]. August 2011 [cit. 2020-03-15]. Dostupné z: https://www.r-bloggers.com/how-much-of-r-is-written-in-r.

[17] Zaki, M. J. a Wong, L. Data mining techniques. In: *Selected Topics in Post-Genome Knowledge Discovery.* 3. vyd. London: World Scientific, 2004, s. 125–163. ISBN 981-238-780-3.

[18] Zhao, Y. *R and data mining: Examples and case studies.* 1. vyd. San Diego: Academic Press, Elsevier, 2013. ISBN 978-0-12-396963-7.

Príloha A

Obsah priloženého pamäťového média

Súčasťou tejto práce je priložený dátový nosič obsahujúci:

- `BP_xkruty00` – koreňový adresár
  - `experimenty` – adresár obsahujúci experimenty
    - `experiment1` – experiment dolovania asociačných pravidiel
    - `experiment2` – experiment bayesovskej klasifikácie
    - `experiment3` – experiment klasifikácie s rozhodovacími stromami
    - `experiment4` – experiment klasifikácie s neurónovou sieťou
    - `experiment5` – experiment zhlukovania
  - `technicka_sprava` – adresár obsahujúci súbory potrebné na vytvorenie technickej správy (preklad otestovaný na školskom serveri merlin a v prostredí Overleaf)
  - `README.md` – dokumentácia projektu
  - `bp.pdf` – technická správa vo formáte `pdf` (preložené v prostredí Overleaf)
  - `bp_print.pdf` – technická správa vo formáte `pdf` určená na tlač (preložené v prostredí Overleaf)