České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Jan Jakeš**

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: **Shlukovací algoritmy pro nečíselná data**

Pokyny pro vypracování:

Seznamte se s algoritmy pro shlukování dat a zaměřte se na algoritmy umožnující zpracovávat také nominální (ne-numerická) data. Dále se seznamte s implementací algoritmů pro shlukovou analýzu v SW Rapidminer a možnostmi vyhodnocení a podpory interpretace výsledků shlukové analýzy v Rapidmineru. Po konzultaci s vedoucím práce implementujte uzly obsahující metriky vyhodnocující úspěšnost shlukování (například silhouette, vnitro-shlukovou variaci nebo uniformitu shluků v proměnné) a podporující interpretaci výsledků shlukové analýzy (distribuce hodnot mezi shluky, typické hodnoty, reprezentanti shluků). Implementaci vašich uzlů demonstrujte na veřejně dostupných datech - například transakční data ze soutěží "Allstate Purchase Prediction Challenge" nebo části dat "Acquire Valued Shoppers Challenge" serveru Kaggle.

Seznam odborné literatury:

Duda, Hart, Stork: Pattern Classifiaction, Wiley-Interscience, 2000, ISBN 0-471-05669-3
Hall, Witten, Frank: Data Mining, Morgan Kaufmann Publishers, 2011, ISBN: 978-0-12-374856-0
How to Extend Rapidminer, https://rapidminer.com/wp-content/uploads/2013/10/How-to-Extend-RapidMiner-5.pdf
Lukasová, Šarmanová: Metody shlukové analýzy, SNTL, 1985

Vedoucí: Ing. Miroslav Čepek, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 24. 3. 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Shlukovací algoritmy pro nečíselná data

Bc. Jan Jakeš

Vedoucí práce: Ing. Miroslav Čepek, Ph. D.

Studijní program: Otevřená informatika

Obor: Softwarové inženýrství

květen 2015

Poděkování

Tímto bych chtěl poděkovat Ing. Miroslavu Čepkovi za odborné rady a připomínky při vedení této Diplomové práce.

Taktéž bych chtěl poděkovat své přítelkyni, rodině a přátelům, kteří mě po celou dobu studia podporovali a poskytovali mi zázemí.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona £. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákon· (autorský zákon).

V Praze dne 5.5. 2015 ……

Abstract

The main goal of this diploma thesis is implementation of evaluation methods of cluster data analysis for RapidMiner software. In the opening, reader is introduced to the issue regarding cluster data analysis, including cluster methods and similarity measures. Evaluation methods of cluster analysis, specifically calculation of value of Rand index, Dunn index, silhouette and determination of typical values in clusters. Thesis also aims at software RapidMiner and implementation of new operators – Rand Index, Dunn Index, Silhouette and Density. All implemented operators were mainly designed to work with nominal data, but due to better applicability in RapidMiner, their functionality was expanded to process even numerical and mixed data. In the end of thesis, practical use of new operators in RapidMiner is described, including testing of operators on larger datasets and verification of values calculated by operators.

Abstrakt

Cílem této diplomové práce je implementace evaluačních metod shlukové analýzy dat pro software RapidMiner. V úvodu je čtenář seznámen s problematikou tykající se shlukové analýzy dat, včetně shlukovacích metod a metrik nepodobnosti. Blíže jsou popsány evaluační metody shlukové analýzy, konkrétně výpočty hodnot Rand indexu, Dunn indexu, siluety a určení typických hodnot ve shlucích. Dále se práce věnuje samotnému RapidMineru a implementací nových operátorů Rand Index operátor, Dunn Index operátor, Silhouette operátor a Density operátor. Všechny implementované operátory byly primárně určeny pro nominální data, avšak kvůli větší použitelnosti v RapidMineru byla funkcionalita rozšířena i pro zpracování numerických a smíšených dat. Na konci práce je popsáno praktické použití nových operátorů v RapidMineru, včetně testování operátorů na větších souborech dat a ověření správnosti operátory vypočítaných hodnot.

Obsah

Seznam obrázků

Úvod

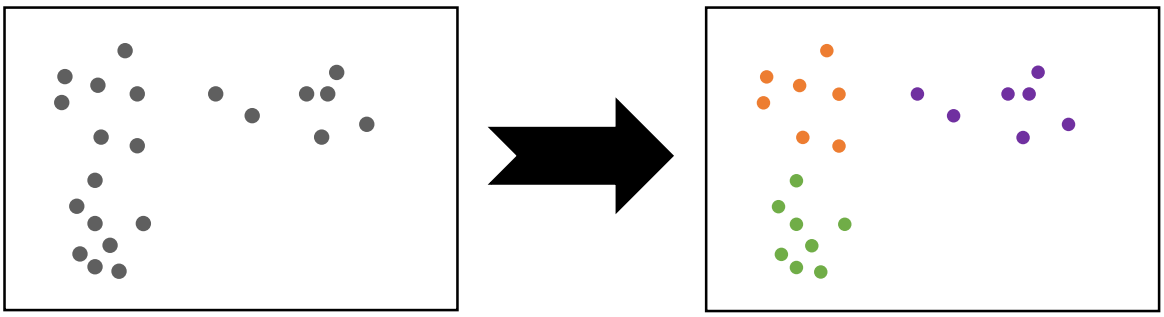
Hlavní náplní této diplomové práce je seznámit se softwarem RapidMiner, prozkoumat metody pro shlukovou analýzu dat implementovaných v RapidMineru a rozšířit RapidMiner o nové operátory pro vyhodnocení shlukové analýzy. RapidMiner je analytické softwarové prostředí použitelné pro strojové učení, vytěžování dat, prediktivní analýzu a shlukování. Shlukovacích metod je v aplikaci na výběr velké množství, stejně tak i metrik pro výpočet podobností mezi objekty. Při zkoumání implementace jsem zjistil, že všechny shlukovací metody implementované v RapidMineru počítají se stejným balíkem podobnostních metrik. Z tohoto důvodu jsem se rozhodl, že mnou implementované operátory budou pracovat se stejným balíkem podobnostních metrik jako shlukovací metody. To znamená, že implementované operátory v této práci zvládají kromě vyhodnocení nominálních dat i vyhodnocení numerických dat, a dokonce i smíšených dat. Po domluvě s vedoucím jsem vybral pro implementaci čtyři chybějící operátory pro vyhodnocení shlukovaných dat v RapidMineru. Prvním je operátor pro zjištění rozdílů mezi dvěma výsledky shlukování, a to pomocí hodnoty Rand indexu. Druhým implementovaným operátorem je operátor pro určení vnitro-shlukové evaluace vypočtením hodnoty Dunn indexu. Další operátor slouží k výpočtu hodnoty siluety ke každé instanci shlukovaných dat a výpočtu průměrné hodnoty siluety ve shlucích. Hodnoty siluet je možné zobrazit číselně nebo v grafu. Posledním implementovaným operátorem je operátor pro určení typických hodnot v jednotlivých shlucích a zobrazení hustoty zastoupení hodnot v jednotlivých shlucích. U implementace všech operátorů bylo přihlíženo na komplexnost a jednoduchost použití. Všechny implementované operátory jdou do RapidMineru přidat jednoduše jako plugin.

Diplomová práce je rozdělena do sedmi kapitol. V této části označené jako Úvod je stručně popsána hlavní náplň práce. První kapitola se věnuje popisu shluků a shlukové analýzy. Následující kapitola popisuje podobnostní metriky pro různé typy vstupních dat včetně výpočtu. Shlukovacím metodám je věnována třetí kapitola. Tato kapitola obsahuje popis jak hierarchických, tak nehierarchických metod. Pro každý typ shlukovací metody jsou zde popsány algoritmy, které je možné v RapidMineru použít. Další kapitola je věnována metrikám shlukových metod. V kapitole se vysvětluje výpočet Rand indexu, Dunn indexu, siluety a postup, jak určit typické hodnoty. V páté kapitole jsou obsaženy základní informace o softwaru RapidMiner. Kapitola šestá je věnována implementovaným operátorům. Ke každému operátoru je popis všech jeho částí, včetně vstupních a výstupních portů, nastavitelných parametrů i popis implementovaných tříd. Poslední kapitola rozebírá testování nově implementovaných operátorů v RapidMineru. Kapitola obsahuje testy na malém souboru dat s ověřením správnosti výsledků a dále výkonnostní testy na větším souboru dat Allstate Purchase Prediction Challenge, který obsahuje převážně nominální hodnoty. Po této kapitole následuje závěr celé práce shrnující její výsledky.

Při tvorbě této diplomové práce byly využity jednak tradiční prameny a vědecké statě, jednak cizojazyčná literatura, která byla často jedinou dostupnou literaturou k tématu vůbec. S ohledem na tento fakt se případným recipientům této práce omlouvám za možné drobné jazykové nedostatky, a to především v exaktním pojmenování termínů, které mnohdy v podmínkách české odborné literatury dosud nebyly odborně pojmenovány.

1 Shluky a shluková analýza

Shluková analýza nebo též shlukování je úloha seskupení objektů do množiny a to tak, že objekty ve stejné množině jsou si podle určitého znaku podobné, naopak objekty mimo tuto množinu jsou podle určitého znaku rozdílné. Snaží se tedy seskupit data do smysluplných struktur a vytvořit taxonomii. Struktury vytvořené pomocí shlukové analýzy se nazývají shluky případně clustery.



obr. 1 - Ukázka shlukové analýzy

Jedná se o hlavní úlohu průzkumného vytěžování dat a obecnou techniku pro statickou analýzu dat. Patří mezi metody strojového učení bez učitele, není tedy třeba trénovací multimnožina.

1.1 Vstupní data pro shlukování

Vstupní data pro shlukování jsou složena z objektů, které chceme shlukovat. Jako objekt si můžeme přestavit různé rostliny, zvířata, lidi, předměty, nemoci ale i vlastnosti živočichů nebo věcí, investice, hospodaření firem, webové stránky a textové dokumenty. Prakticky se může jednat o cokoliv, co jsme schopni popsat vektorem hodnot. Jednotlivé objekty jsou seskupeny do tabulky, představující matici vstupních dat. Matice objektů tedy představuje vstupní data pro shlukovou analýzu.

Počet objektů v matici je obvykle označován písmenem n. Každý objekt představuje jeden řádek matice. Pro označení jednoho řádku matice existuje mnoho termínů, se kterými se můžeme setkat. V matematice či informatice se řádek obvykle označuje termínem vektor, v ostatních vědních oborech se můžeme setkat s termíny vzor, případ, pozorovaní, záznam apod. V této práci bude jeden řádek matice označován jako objekt nebo vektor. Sloupce vstupní matice představují jednotlivé znaky objektu neboli proměnné. Počet proměnných objektů se označuje písmenem m. Proměnné představují hodnoty vlastností nebo různé znaky jednotlivých objektů. Představme si, že máme objekty, které reprezentují

automobily. Proměnnými takového objektu mohou být například značka automobilu, spotřeba automobilu, objem motoru, barva karoserie atd.

Vstupní matice má velikost $n \times m$. V této práci bude vstupní matice objektů označována jako $\boldsymbol{X}$. Prvky této matice pak budou označovány jako x_{il}, kde $i = 1, \dots, n$ označuje konkrétní řádek matice a $l = 1, \dots, m$ označuje konkrétní sloupec matice.

1.2 Shluky

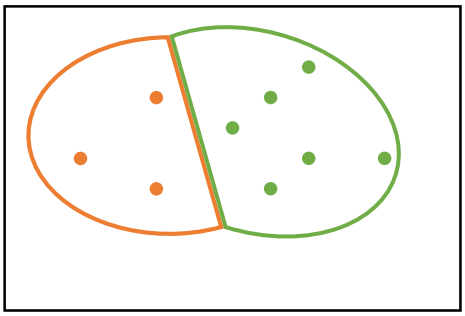
Pojmem shluk se v informatice převážně myslí množina objektů s podobnými vlastnostmi. To znamená, že ve vstupní matici dat hledáme takové vektory, které mají podobné proměnné. Objekt je více podobný ostatním objektům ve shluku, do kterého patří a rozdílný od objektů shluků, do kterých nepatří. Proces vytváření takovéto množiny se nazývá *shlukování objektů*.

Nejedná se však o jediné typy shluků. Kniha [13] popisuje další typy shluků. Jiným typem shluků jsou shluky proměnných. V informatice se jedná převážně o shlukování slov nebo termů při vyhledávání informací. Dále se shlukování proměnných používá ke snižování rozměru vektoru charakterizujícího objekty.

Další varianty jsou současné shlukování objektů a proměnných, shlukování kategorií nominálních dat. Podrobněji se těmto shlukům věnuje kniha [13].

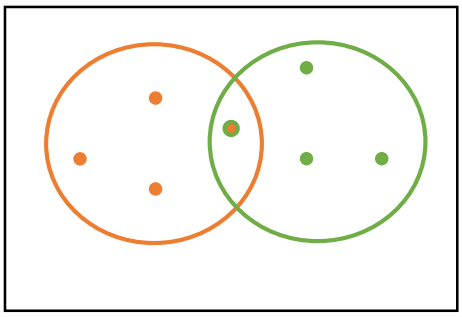
1.3 Rozdělení shluků

Shlukovací metody se dají rozdělit do dvou skupin na disjunktní shlukování a překrývající se shlukování. Disjunktivní shlukování je složeno se shluků, kde každý objekt patří do právě jednoho shluku.



obr. 2 - Disjunktivní shlukování

Překrývající se shlukování dovoluje, aby objekt patřil do více shluků. Shluky se tedy navzájem překrývají.



obr. 3 - Překrývající se shlukování

Většina metod shlukové analýzy tvoří disjunktivní shluky. Jednotlivé shluky budou v této práci označovány písmenem C_h, kde $h = 1, \ldots, k$. Hodnota k představuje počet shluků.

2 Podobnost a nepodobnost objektů

Podobnost objektů je jedním z velmi důležitých faktorů ovlivňujících výsledek shlukovacího algoritmu. Výběrem různých výpočtů podobností můžeme dosáhnout u stejné shlukovací metody různých výsledků. Je tedy nutné zvolit správnou míru podobnosti, případně nepodobnosti objektů. Volbu ovlivňuje jak zvolená metoda shlukování, tak samotný charakter dat.

2.1 Měření podobnosti a nepodobnosti

Ve shlukové analýze se snažíme seskupovat objekty, které mají podobné vlastnosti. Je tedy nutné mezi objekty hledat ty, které jsou si podobné. Jak jsou dva objekty podobné zjistíme díky měření míry podobnosti objektů nebo též častěji používané míry nepodobnosti objektů.

Míra podobnosti představuje číselnou hodnotu reprezentující, jak jsou si dva objekty podobné. V literatuře [13] a [12] se míra podobnosti objektů x_i a x_j označuje jako $S(x_i,\ x_j)$, případně zkráceně jako $S_{i,j}$. V této práci budeme používat stejné označení. Míra podobnosti by přinejmenším měla splňovat tyto podmínky:

1. $S_{i,j} \geq 0$
2. $S_{i,j} = S_{j,i}$
3. $S_{i,i} = maximální\ hodnota\ z\ oboru\ hodnot\ S$

Míra nepodobnosti představuje číselnou hodnotu reprezentující, jak moc jsou si dva objekty nepodobné, tedy jak moc se od sebe liší. Pro objekty x_i a x_j označujeme míru nepodobnosti jako $D(x_i, x_j)$, případně zkráceně jako $D_{i,j}$. Měla by splňovat přinejmenším tyto podmínky:

1. $D_{i,j} \geq 0$
2. $D_{i,j} = D_{j,i}$
3. $D_{i,i} = 0$

2.2 Kvantitativní data

Objekt složen z kvantitativních dat představuje vektor číselných hodnot. U kvantitativních dat se míra nepodobnosti počítá jako vzdálenost objektů v prostoru. Proto se často místo pojmu míra nepodobnosti používá míra vzdálenosti. Souřadnice pro výpočet vzdálenosti jsou určeny proměnnými objektů. V případě, kdy je splněna trojúhelníková nerovnost mezi objekty, tj.

$$D_{iy} + D_{yi} \geq D_{ij}\ (i, j, y \in \langle 1; n \rangle)\ ,$$

pak hovoříme o *metrice*.

Podle knihy [13] k nejznámějším typům vzdáleností patří:

- **Euklidovská vzdálenost D_E**

$$D_E(x_i, x_j) = \sqrt{\sum_{l=1}^{m} (x_{il} - x_{lj})^2} = \|x_{il} - x_{lj}\|$$

- **Vážená euklidovská vzdálenost D_{EW}**

$$D_{EW}(x_i, x_j) = \sqrt{\sum_{l=1}^{m} w_l\ (x_{il} - x_{jl})^2}$$

- **Čtvercová euklidovská vzdálenost D_{ES}**

$$D_{ES}(x_i, x_j) = \sum_{l=1}^{m} (x_{il} - x_{jl})^2$$

- **Manhattanská vzdálenost D_B**

$$D_B(x_i, x_j) = \sum_{l=1}^{m} |x_{il} - x_{jl}| = |x_l - x_j|$$

- **Čebyševova vzdálenost D_C**

$$D_C(x_i, x_j) = max_l(|x_{il} - x_{jl}|)$$

- **Minkowského vzdálenost D_M**

$$D_M(x_i, x_j) = \sqrt[q]{\sum_{l=1}^{m} |x_{il} - x_{jl}|^q}$$

- **Lanceyho-Williamsova vzdálenost D_{LW}**

$$D_{LW}(x_i, x_j) = \sum_{l=1}^{m} \frac{|x_{il} - x_{jl}|}{|x_{il}| + |x_{jl}|}$$

pro $|x_{il}| + |x_{jl}| \neq 0$ (jinak je vzdálenost 0)

Podrobnější popis vzorců je možné nalézt v knihách [13] a [12].

V případech, kdy vektory hodnot jednotlivých objektů obsahují pouze čísla větší nebo rovny nule a zároveň kladné hodnoty mají větší význam než hodnota 0, mohou být místo měr nepodobnosti použity míry podobnosti. Jako nejznámější míry podobnosti jsou v literaturách uváděny kosinova míra, Jaccardův koeficient, Diceho koeficient a Czekanowského koeficient.

- **Kosinova míra S_K**

$$S_K(x_i, x_j) = \frac{\sum_{l=1}^{m} x_{il} x_{jl}}{\sqrt{\sum_{l=1}^{m} x_{il}^2 \sum_{l=1}^{m} x_{jl}^2}} = \frac{\sum_{l=1}^{m} x_{il} x_{jl}}{\|x_i\| \cdot \|x_j\|}$$

- **Jaccardův koeficient S_J**

$$S_J(x_i, x_j) = \frac{\sum_{l=1}^{m} x_{il} x_{jl}}{\sum_{l=1}^{m} x_{il}^2 + \sum_{l=1}^{m} x_{jl}^2 - \sum_{l=1}^{m} x_{il} x_{jl}}$$

- **Diceho koeficient S_D**

$$S_D(x_i, x_j) = \frac{2 \sum_{l=1}^{m} x_{il} x_{jl}}{\sum_{l=1}^{m} x_{il}^2 + \sum_{l=1}^{m} x_{jl}^2}$$

- **Czekanowského koeficient S_C**

$$S_C(x_i, x_j) = \frac{2 \sum_{l=1}^{m} \min\{x_{il} x_{jl}\}}{\sum_{l=1}^{m} (x_{il} + x_{jl})}$$

2.3 Nominální data

V případě, že je objekt složen z nominálních dat, tedy z dat, které popisují vlastnosti, kvalitu nebo informace o objektu, máme velice omezenou množinu možností, jak určit podobnost či nepodobnost dvou objektů. Pro ukázku si představme množinu objektů reprezentující automobily. Jednotlivé nominální hodnoty těchto objektů následně mohou být například barva auta, značka auta, typ

karoserie atd. U takovýchto jediné co lze zkoumat je to, zda jsou hodnoty shodné nebo rozdílné. Pro výpočet se používají dva koeficienty, koeficient prosté shody a koeficient nesouhlasu.

- **Koeficient prosté shody**

Jedná se o podíl všech nominálních hodnot objektu, u nichž jsou hodnoty shodné, a celkového počtu proměnných.

$$S_{ij} = \frac{\sum_{l=1}^{m} S_{ijl}}{m},$$

kde $S_{ijl} = 1 \Leftrightarrow x_{il} = x_{jl}$ a $S_{ijl} = 0$ jinak.

- **Koeficient nesouhlasu**

Koeficient nesouhlasu se spočítá obdobně jako koeficient prosté shody. Rozdíl ve vzorci je v tom, že jmenovatel ve zlomku nepředstavuje suma shodných hodnot, ale suma rozdílných hodnot.

Mezi koeficientem prosté shody a koeficientem nesouhlasu platí vztah

$$D_{ij} = 1 - S_{ij}.$$

2.4 Dichotomická data

Tento typ dat se často používá ve vědních disciplínách, jako jsou společenské vědy, medicína apod. Data tedy mohou nabývat hodnot například muž/žena, ano/ne atd. U dichotomických dat jsou vždy jen dvě možnosti hodnot, které lze zvolit. Je tedy velmi jednoduché převést tato data do binárních hodnot. Při výpočtu podobnosti dvou objektů složených z dichotomických dat je důležité určit, zda jsou proměnné symetrické nebo asymetrické.

Symetrické proměnné se vyznačují tím, že obě hodnoty jsou stejně důležité. Může se jednat o proměnné, již zmíněné výše, tedy například muž/žena. Pro výpočet podobnosti objektů, se symetrickými proměnnými se používají stejné vzorce, které se používají pro výpočet podobnosti u objektů s nominální daty (viz oddíl 3.1.2).

V případě asymetrických proměnných je jedna hodnota důležitější než hodnota druhá. Může se jednat například o data typu: je důležitější pacienta uzdravit než neuzdravit. Podle knihy [13] se pro výpočet podobnosti objektů s asynchronními dichotomickými daty používají míry pro asymetrické binární proměnné, jimiž jsou Jaccardův koeficient, Diceho koeficient a Czakanowského koeficient.

Obecný vzorec, který předpokládá kódy 0 (absence) a 1 (výskyt), je pro danou třídu měr následující (viz [13]):

$$S_{PA}(x_i, x_j) = \frac{\Theta \sum_{l=1}^{m} x_{il} x_{jl}}{\Theta \sum_{l=1}^{m} x_{il} x_{jl} + \sum_{l=1}^{m} |x_{il} - x_{jl}|}.$$

Jestliže $\Theta = 1$, jde o Jaccardův koeficient, pro $\Theta = 2$, dostáváme o Czakanowského koeficient (též Diceho koeficient).

2.5 Kombinace datových typů

V případě, že máme m objektů takových, že proměnné objektu jsou různého typu, pak je vhodné použít Gowerův koeficient podobnosti (viz [24] a [13]).

$$S_{ij} = \frac{\sum_{l=1}^{m} w_{ijl} S_{ijl}}{\sum_{l=1}^{m} w_{ijl}},$$

kde

- S_{ijl} míra podobnosti mezi objekty $\boldsymbol{x}_i$ a $\boldsymbol{x}_j$ na základě l-té proměnné.
- w_{ijl} nabývá hodnoty 0 (v případě, že hodnota x_{il} nebo x_{jl} chybí nebo jsou obě tyto hodnoty rovny nule a l-tá proměnná je binární) nebo hodnoty 1 v ostatních případech.

Míra podobnosti S_{ijl} závisí na typu l-té proměnné:

- $\boldsymbol{X}_l$ je binární nebo nominální: $S_{ijl} = 1 \Leftrightarrow x_{il} = x_{jl}$ a $S_{ijl} = 0$ v ostatních případech.
- $\boldsymbol{X}_l$ je kvantitativní: $S_{ijl} = 1 - \frac{|x_{il} - x_{jl}|}{R_l}$, kde R_l je variační rozpětí.

3 Dělení metod shlukové analýzy

Metody shlukové analýzy se dají dělit do různých skupin podle různých kritérií. Nejčastěji se používá dělení metod shlukové analýzy podle výsledné struktury shluků, a to na hierarchické a nehierarchické. Tohoto rozdělení se budeme držet i v této práci. Dále se dají metody shlukové analýzy dělit podle pevně zadaného počtu shluků nebo nalezení optimálního počtu shluků za běhu vybrané metody.

3.1 Hierarchické metody

Hierarchické metody shlukové analýzy jsou metody, jejichž výsledkem jsou shluky seřazeny v hierarchické struktuře. Každý shluk je tedy reprezentován jedním hierarchickým stromem. Jednotlivé uzly stromu pak představují shluky.

U hierarchických metod shlukové analýzy můžeme rozlišovat několik přístupů. Jedním přístupem ke shlukování je:

- **Monotetický** – shluky se vytvářejí na každé úrovni podle pouze jedné proměnné
- **Polytetický** – shluky se vytvářejí s ohledem na všechny proměnné současně

Jiným přístupem může být:

- **Aglomerativní** - vychází se ze stavu, že každý objekt je samostatný shluk. Při každé iteraci se dva nejpodobnější shluky spojí v jeden shluk. Tento postup může pokračovat do doby, kdy jsou všechny objekty v jednom shluku.
- **Divizní** - vychází ze stavu, že všechny objekty patří do stejného shluku. Při každé iteraci se dvě nejméně podobné skupiny objektů ve shluku rozdělí. Postupné rozdělování končí stavem, kdy každý objekt je samostatným shlukem.

3.1.1 Monotetické shlukování

Monotetické shlukování je primárně určeno pro binární data. Pro shlukování se obvykle používá divizní přístup. Vycházíme tedy z toho, že všechny objekty jsou v jednom shluku, který bude rozdělen na dva. Každý objekt má *m* proměnných. Z těchto *m* proměnných je vybrána jedna a podle této hodnoty se objekty rozdělí do dvou shluků. Protože proměnné obsahují pouze binární hodnoty, rozdělení se provede podle hodnot vybraných proměnných tak, že jedna skupina má proměnnou nastavenou na hodnotu 0 a druhá skupina na hodnotu 1. V každé iteraci se tedy snižuje počet použitých parametrů o 1.

V první iteraci tedy vybíráme z *m* proměnných, podle kterých budou data dělena, v druhé iteraci *(m - 1)* atd.

Výběr proměnné pro dělení se určí na základě porovnání intenzit závislostí všech dvojic proměnných. Pro zjištění intenzity závislosti mezi proměnou *k* a proměnnou *l* může být použita míra

$$q_{kl} = |a_{kl}d_{kl} - b_{kl}c_{kl}|,$$

kde a_{kl}, b_{kl}, c_{kl}, d_{kl} jsou četnosti vytvořené v kontingenční tabulce pro proměnné *k* a *l*. Pro každou *l*-tou proměnnou spočítáme hodnotu

$$q_l = \sum_{k \neq l} q_{kl}\,, k = 1,2,3, \dots, m\,.$$

Podle proměnné, u níž bylo dosaženo maxima z těchto hodnot, tj. $max_l(q_l)$, se následně provede zařazení objektů. [13]

Velkou výhodou tohoto přístupu shlukování je, že po dokončení shlukování je vytvořen model dat. Podle tohoto modelu je možné velice snadno zařadit nový objekt, který nebyl v původních datech, do správného shluku. Každý shluk je reprezentován binárními hodnotami proměnných, díky čemuž jsou vytvořena alokační pravidla. Díky těmto pravidlům je možné objekt jednoznačně zařadit. Monotetická analýza je označována též jako jeden z přístupů konceptuálního shlukování, viz [13] a [25].

3.1.2 Polytetické shlukování

U tohoto typu shlukování se shluky vytvářejí s ohledem na všechny proměnné. Využívá se jak *aglomerativní* přístup, tak *divizní* přístup.

3.1.3 Aglomerativní metody

U aglomerativních metod hierarchického shlukování se vychází ze stavu, že každý objekt přestavuje jeden samostatný shluk. Mějme tedy n objektů reprezentujících právě n shluků. Tento počáteční rozklad se označuje Ω_0. V prvním cyklu se vyberou dva nejvíce podobné shluky a spojí se v jeden. Zda jsou si dva shluky podobné, se určí z hodnot matice podobností. Hodnoty matice podobností jsou vypočteny v každém cyklu podle typu aglomerativních algoritmů. Nově vzniklý shluk a všechny zbývající shluky z Ω_0 tvoří nový rozklad Ω_1. V dalších krocích jsou tedy vždy vybrány dva shluky nižšího stupně rozkladu s největší podobností a spojí se v nový shluk. Jinými slovy z rozkladu Ω_{i-1} vybereme dva nejpodobnější shluky a spojíme je. Tím vznikne rozklad množiny Ω_i. Počet cyklů pro zpracování všech objektů je tedy $n - 1$. Pokud zpracujeme všechny objekty, dostaneme pouze jeden shluk, což většinou nechceme. Pro ukončení se proto používají různé ukončující podmínky, které

zkracují běh algoritmu. Mezi často používané podmínky pro ukončení patří dosažení předem stanoveného počtu shluků nebo pokud podobnost dvou nejpodobnějších shluků překročí nastavenou mez.

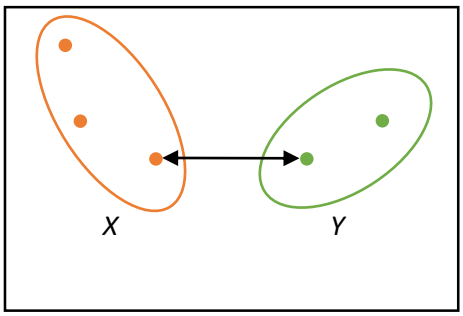
Pokud jsou si dva objekty nebo shluky podobné, znamená to, že jsou si v grafickém zobrazení velice blízko. Používá se tedy místo pojmu podobnost termín vzdálenost. Vzorce použité níže jsou převzaty z literatury [13], [12] a [9].

Metoda nejbližšího souseda (Simple linkage)

Metoda nejbližšího souseda hledá pro spojení shluky, které mají objekty nejblíže u sebe. To znamená, že hledáme dva objekty z různých shluků, které mezi sebou mají nejkratší vzdálenost. Shluky obsahující dva nejbližší prvky se spojí v jeden. Matematicky může být metoda nejbližšího souseda popsána vzdáleností $D(X,Y)$ pokud platí podmínka $X \neq Y$ jako

$$D(X,Y) = min_{x \in X, y \in Y} d(x,y),$$

kde X,Y jsou dvě skupiny objektů považované za shluky a $d(x,y)$ značí vzdálenost mezi dvěma objekty x a y.



obr. 4 - Metoda nejbližšího souseda

Výsledné shluky nemají sférický charakter. Dochází k řetězení objektů. Nevýhodou této metody je brzké spojování dobře rozpoznatelných shluků do jednoho z důvodů, že se některé objekty obou shluků příliš přiblížily, viz [12],[3],[26] a [27].

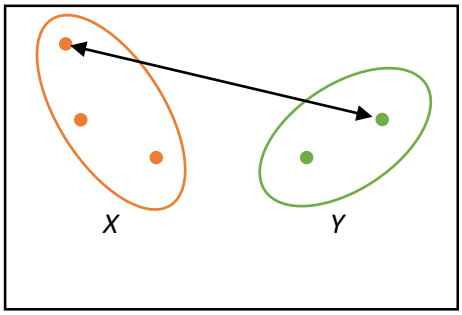
Metoda nejvzdálenějšího souseda (Complete linkage)

Metoda nejvzdálenějšího souseda hledá pro spojení shluky, které mají nejvzdálenější objekty ve shluku nejblíže u sebe. Tedy vybereme dva shluky, nalezneme dva od sebe nejvzdálenější objekty

těchto shluků a vypočítáme jejich vzdálenost. Tento postup provedeme pro všechny dvojice shluků a z výsledných vzdáleností vybereme tu nejkratší. Matematicky může být metoda nejvzdálenějšího souseda popsána vzdáleností $D(X,Y)$ pokud platí podmínka $X \neq Y$ jako

$$D(X,Y) = max_{x \in X, y \in Y} d(x,y),$$

kde X,Y jsou dvě skupiny objektů považované za shluky a $d(x,y)$ značí vzdálenost mezi dvěma objekty x a y.



obr. 5 - Metoda nejvzdálenějšího souseda

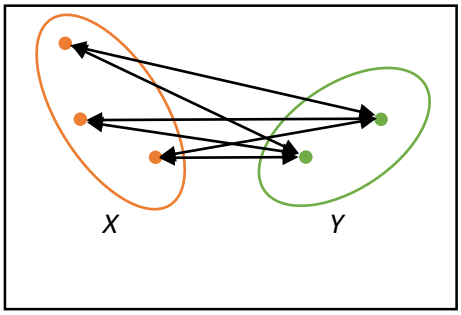
Metoda nejvzdálenějšího souseda je zbavena nevýhod, které měla metoda nejbližšího souseda. Nedochází k řetězení objektů a shluků. Shluky vytvořené touto metodou mají přibližně stejnou velikost. Má tendenci tvořit sférické shluky. Viz [12],[3] a [28].

Metoda průměrné vazby (Average linkage)

U metody průměrné vazby probíhá postup výběru shluků pro sjednocení následovně. Postupně se vybírají všechny dvojice shluků. Následně ze všech dvojic shluků metoda vytvoří dvojice objektů, kde jeden objekt patří do prvního shluku a druhý do druhého. Mezi každou dvojicí objektů se spočítá vzdálenost. Ze všech vypočtených vzdáleností mezi dvěma shluky se spočítá průměrná vzdálenost. Následně se porovnají všechny průměrné vzdálenosti mezi všemi shluky a nalezne se ta nejkratší. Shluky, které budou mít mezi sebou nejkratší průměrnou vzdálenost, se spojí. Matematicky muže být metoda průměrné vazby popsána vzdáleností $D(X,Y)$, pokud platí podmínka $X \neq Y$ jako

$$D(X,Y) = \frac{1}{n_x n_y} \sum_{x \in X} \sum_{y \in Y} d(x,y)\,,$$

kde X,Y jsou dvě skupiny objektů považované za shluky, $d(x,y)$ značí vzdálenost mezi dvěma objekty x a y. Hodnoty n_x a n_y představují počet objektů ve shlucích X a Y.



obr. 6 - Metoda průměrné vazby

Spojování shluků probíhá ve větších vzdálenostech než u metody nejbližšího souseda. Nedochází tak ke spojování velkých shluků, které mají pouze dva objekty velice blízko sebe. Přidání dalších objektů většinou nenaruší výsledek shlukování. U metody nedochází tak často k nejednoznačnosti výsledků díky tomu, že průměrná vzdálenost mezi shluky bývá jedinečná. Podrobněji viz [12],[3] a [9].

Centroidní metoda

Metoda ke spojení dvou shluků používá euklidovskou vzdálenost centroidů těchto shluků. Postup výběru shluků pro spojení probíhá následovně: Nejprve se spočtou centroidy všech shluků, které představují těžiště shluku. Jedná se o hypotetický bod se souřadnicemi získanými jako průměr všech souřadnic objektů daného shluku. Mezi těmito centroidy se spočítá euklidovská vzdálenost. Mezi shluky, kde je vypočtená vzdálenost nejmenší, dojde ke spojení. Matematicky muže být centroidní metoda popsána následovně: Nechť $\overline{C_X}$ a $\overline{C_Y}$ jsou těžiště shluků $X = \{ X_1, X_2, \dots X_k \}$ a $Y = \{ Y_1, Y_2, \dots Y_l \}$, kde $X_i = \{ x_{i1}, x_{i2}, \dots x_{ip} \}, i = 1, 2, \dots, k$, jsou objekty shluku X, $Y_i = \{ y_{i1}, y_{i2}, \dots y_{ip} \}, i = 1, 2, \dots, l$, jsou objekty shluku Y.

$$\overline{C_X} = \left(\overline{x_1}, \overline{x_2}, \dots, \overline{x_p}\right), \overline{x_j} = \frac{1}{k} \sum_{i=1}^{k} x_{i,j},$$

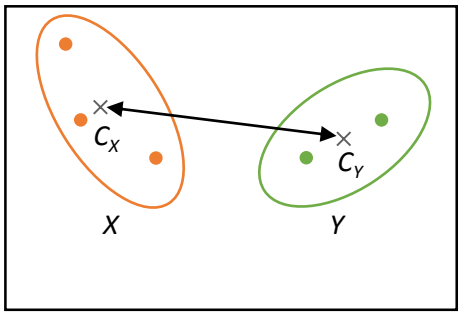
$$\overline{C_Y} = \left(\overline{y_1}, \overline{y_2}, \dots, \overline{y_p}\right), \overline{y_j} = \frac{1}{k} \sum_{i=1}^{k} y_{i,j},$$

pro $j = 1, 2, \dots, p$.

Potom

$$D(X,Y) = \|\overline{C_X} - \overline{C_Y}\|$$

je vzdálenost mezi shluky X a Y.



obr. 7 - Centroidní metoda

Vice podrobností k výpočtu vzdáleností dvou shluků u centroidní metody lze nalézt v [12] a [9]

Mediánová metoda

Mediánová metoda vychází z centroidní metody popsané výše. Metoda se snaží odstranit nedostatek, který má centroidní metoda. Autor této metody J. C. Gower konstatoval, že rozdílné počty objektů ve shlucích způsobí rozdílnou váhu prvních dvou složek rekurzivního přepisu centroidní metody a díky tomu se ztrácejí vlastnosti malých shluků v konečném sjednocení. Metoda se nazývá mediánová, protože spojnice těžiště shluku U s těžištěm shluku $R = P \cup L$ leží na střední příčce trojúhelníku, jehož vrcholy jsou P, L, U. Podrobněji v [12].

Koeficient nepodobnosti je dán vztahem

$$D(O_i, O_j) = \|O_i - O_j\|,$$

$$D(U, \mathrm{R}) = \frac{1}{2}D(U,P) + \frac{1}{2}D(U,L) - \frac{1}{2}D(P,L).$$

Wardova metoda

U Wardovy metody není principem nalézt optimální vzdálenost mezi shluky a ty následně spojit, jak tomu bylo u ostatních metod. U této metody minimalizujeme heterogenitu shluků podle kritéria

minima přírůstku vnitroskupinového součtu čtverců odchylek objektů od těžiště shluků. V každém kroku se pro všechny dvojice odchylek spočítá přírůstek součtu čtverců odchylek, vzniklý jejich sloučením a pak se spojí ty shluky, kterým odpovídá minimální hodnota tohoto přírůstku. Převzato z [29].

3.1.4 Divizní metody

Divizní metody mají inverzní postup shlukování k aglomerativním metodám. Jak bylo popsáno výše, aglomerativní metody postupně spojují shluky, až dostaneme jeden shluk obsahující všechny objekty. Naopak divizní metody vycházejí z jednoho shluku, který je dělen na menší shluky.

Pro metody divizního shlukování tedy nejprve všechny objekty sloučíme do jednoho shluku. Tento jeden shluk představuje počáteční stav. Následně tento shluk rozdělíme na dva shluky podle kritéria pro rozdělení u konkrétní metody. Vzniknou tedy dva shluky. Z těchto dvou nových shluků vybereme jeden a ten opět rozdělíme na dva. Tento postup opakujeme tak dlouho, dokud všechny shluky neobsahují pouze jeden objekt. Díky tomu dostaneme úplné nestratifikované hierarchické shlukování.

Abychom nalezli absolutně optimální rozdělení shluku na dva shluky, je zapotřebí vyzkoušet všechny možné rozklady a nalézt mezi nimi ten nejlepší. Pokud bude původní shluk obsahovat n prvků, je zapotřebí projít všech $2^{n-1} - 1$ rozkladů. Díky takovéto složitosti je jasné, že tento postup se hodí používat jen na velmi malé množiny dat.

MacNaughton–Smithova metoda

MacNaughton–Smithova metoda shlukování přichází se snížením složitosti divizního shlukování až na kvadratickou složitost. Metoda je tedy použitelná i na velká data, ale je zde možnost, že nedostaneme vždy optimální řešení.

Algoritmus začíná ve stavu, kdy jsou všechny objekty přiřazeny do jednoho shluku. V dalším kroku postupně oddělujeme objekty od shluku. Skupina oddělených objektů následně tvoří nový shluk. Před začátkem rozdělování shluku nejprve vypočítáme průměrnou vzdálenost jednotlivých objektů od ostatních objektů ve shluku. Tuto hodnotu označíme jako P_p. Z těchto vypočtených průměrných hodnot vybereme hodnotu maximální. Objekt, který měl maximální průměrnou vzdálenost, představuje základ nově vzniklého shluku. Průměrná vzdálenost P_l tohoto objektu od nově vzniklého shluku je nulová, protože nově vzniklý shluk obsahuje pouze tento jeden objekt. Rozdíl $G = P_p - P_l$ průměrných vzdáleností od objektů původního rozdělovaného shluku a objektů nově vzniklého shluku je kladný. Najdeme další takový objekt mezi zbývajícími objekty v původním shluku, jehož rozdíl průměrných vzdáleností G od objektů v původním shluku a nově vzniklém shluku je maximální. Je-li výsledek

kladný, objekt je přiřazen do nově vzniklého shluku a hledáme další objekt s maximálním rozdílem G. Rozdělování shluku se provádí do doby, dokud je hodnota G kladná. V opačném případě, kdy je hodnota G záporná, objekt zůstane v původním shluku, nepřemísťuje se a rozdělování se tak ukončí. Po ukončení rozdělování máme dva shluky, ze kterých se vybere jeden shluk, který opět bude rozdělen na dva shluky stejným způsobem. Rozdělování trvá do doby, než jsou všechny shluky jednoprvkové.

3.2 Nehierarchické metody

3.2.1 Optimalizační

Metoda K-průměrů (K-means)

Jedná se o iterativní optimalizační metodu shlukování, která se používá v případě, kdy datový soubor obsahuje kvantitativní proměnné. Metoda vychází z počátečního stavu, kde jsou všechny objekty rozděleny do určitého počtu shluků. Počet počátečních shluků bude označován písmenem *k*. Hodnota počtu shluků *k* je stanovena analytikem. Rozdělení objektů se obvykle provádí tak, že se nejprve určí *k* počátečních centroidů, které reprezentují středy shluků. Ke stanovení počátečních pozic centroidů je mnoho přístupů. Mohou být vybrány zcela náhodně ze zadaných objektů. Může se vzít prvních *k* objektů, případně se může použít nějaká složitější metoda, která centroidy umístí na optimálnější pozici. Díky tomu se zkrátí čas hledání řešení. Po stanovení počáteční pozice centroidů se postupně kontrolují vzdálenosti všech objektů od všech centroidů. Pro určení vzdálenosti mezi centroidem a objektem je spočtena euklidovská vzdálenost viz kapitola 2.2. Objekt je následně přiřazen k tomu centroidu, který má euklidovskou vzdálenost nejkratší.

Objekty přiřazené k určitému centroidu tvoří shluk. Pro objekty ve shluku je spočtena nová pozice centroidu, kterým je *m*-rozměrný vektor průměrných hodnot jednotlivých proměnných. Opět se zkoumají vzdálenosti všech objektů od všech centroidů. V případě, že objekt má blíže k centroidu jiného shluku, je objekt přesunut do tohoto bližšího shluku. Celý postup se opakuje do té doby, dokud se mění pozice centroidů. Více v knize [13] a [30].

Sférický K-means s opakovaným půlením (Repeated bisection k-means)

Jedná se o variantu metody K-průměrů. U této metody jsou všechny objekty nejprve přiděleny do jednoho shluku. Tento jeden shluk je rozdělen na dva shluky pomocí klasické metody K-průměrů. Vzniknou tedy dva shluky, ze kterých je vybrán jeden, který se bude opět dělit. Pro výběr vhodného shluku pro dělení je mnoho možných kritérií. Nejčastěji se však vybírá shluk pro dělení podle velikosti. Dělení shluků probíhá do doby, dokud není dosaženo počátečně voleného počtu shluků *k*. Viz literatura [3].

Metoda K-medoinů (Partition around medoid)

Podobně jako u metody K-průměrů i zde metoda počítá s kvantitativními proměnnými a vychází z počátečního rozdělení *k* shluků. Rozdíl proti předchozí metodě k-průměrů je v tom, že pro každý vytvořený shluk je určen medoid. Ten je reprezentován konkrétním objektem ve shluku. Medoid je vybrán tak, aby součet vzdáleností všech objektů ve shluku od tohoto objektu byl minimální.

Následně se zkoumá, zda objekty ve shluku jsou blíže k vlastnímu medoidu nebo medoidu jiného shluku. Pokud je objekt blíže jinému shluku, je k tomuto shluku přidán. Pokud je stále nejbližší původní medoid, objekt zůstává v původním shluku. Po stanovení shluků nastává výpočet nového medoidu. Medoid je stanoven minimalizací funkce, která je součtem vzdáleností jednotlivých objektů od medoidu ve svém shluku. Pokud označíme medoid $m_{g,i}$, kde g reprezentuje shluk, ve kterém se medoid nachází a i reprezentuje objekt ve shluku, potom jsou medoidy určeny tak, aby bylo dosaženo minimum funkce

$$f = \sum_{i=1}^{n} D(x_i, m_{g,i}).$$

Celý postup se opakuje, dokud klesá hodnota funkce *f*. Více k metodě a vzorci viz [13].

Metoda CLARA (Clustering LARge Application)

Jedná se o metodu K-medoidů, která je schopna zpracovávat velké množství dat. Metoda K-medoidů je schopna pracovat efektivně pouze na malém množství dat. Metoda CLARA se tímto pravidlem řídí a aplikuje ho právě na velkém množiny dat. Metoda nejprve náhodně vybere malou množinu objektů a nad touto množinou provede klasickou metodu K-medoidů. Vznikne tedy *k* shluků, do kterých se metoda pokusí přiřadit zbytek dat. Tento postup se opakuje několikrát za sebou. Z výsledného shlukování si vybere to nejvhodnější.

Pokud je množina shlukovaných objektů o hodně menší než množina zpracovávaných dat, nemusí CLARA dojít ke správným výsledkům. Je tedy důležité dobře zvolit počet objektů pro shlukování. Více o metodě viz [21] a [3].

Metoda CLARANS (Clustering Large Application based on RANdomised Search)

Vychází z metody CLARA a PAM. U této metody se opět pracuje s medoidy. Na rozdíl od předchozích metod se však neprochází celá množina objektů, ale pouze objekty do určité vzdálenosti od medoidu. Dále je omezen i počet iterací algoritmu. Maximální vzdálenost a počet iterací jsou určeny na začátku této metody. Zjednodušeně by se dal algoritmus popsat jako prohledávání grafu, kde každý uzel představuje potencionální řešení, což tedy znamená, že bude představovat optimální medoid.

Metoda začne prohledávání v náhodně zvoleném uzlu a z tohoto uzlu navštíví všechny své sousedy do předem zvolené vzdálenosti. Mohou nastat dvě situace. Buďto navštívíme uzel, který lépe vyhovuje, a pokračujeme z něj nebo uzel nevyhovuje a označíme jej za lokální minimum. Konec algoritmu nastane po doběhnutí předem stanoveného počtu iterací.

Metoda K-módů a K-histogramů

Věnuje se shlukování objektů charakterizovaných pomocí nominálních proměnných. Shluky jsou reprezentovány *m*-rozměrnými vektory údajů. V případě K-módů obsahuje vektor modální kategorie jednotlivých proměnných. V případě K-histogramů obsahuje vektor údaje o četnosti kategorií jednotlivých proměnných. Podrobnější popis je možné nalézt v literatuře [13].

Fuzzy shluková analýza

Jedná se o shlukovou analýzu dat s jinou logikou shlukování. V předchozích metodách se objekty přiřazovaly do shluků tím způsobem, že pokud objekt náleží do shluku m_i, tak již nemůže náležet do shluku m_j. Objekty jsou do shluku přiřazeny jednoznačně. Takže objekt do shluku buď patří, nebo nepatří. Přitom je jasné, že objekty blíže středu shluku náleží do shluku více než objekty na krajích shluku, případně objekty na překryvech dvou shluků. Tento problém řeší právě Fuzzy shluková analýza. Ta dovoluje, aby jeden objekt mohl současně patřit do více shluků. Ke každému objektu je spočtena číselná hodnota z intervalu <0,1>, reprezentující stupeň příslušnosti ke každému shluku. Stupeň příslušnosti blížící se hodnotě 0 udává, že daný objekt ke shluku nepatří. Stupeň příslušnosti blížící se 1 naopak určuje, že objekt má velice podobné vlastnosti s ostatními objekty ve shluku a do shluku patří.

Podrobnější informace k Fuzzy shlukové analýze je možné nalézt v [31].

Existuje několik metod, které využívají principy Fuzzy shlukové analýzy. Mezi základní metody patří Fuzzy C-means a Gustafson-Kesselův algoritmus.

Fuzzy C-means

Jedná se o rozšíření klasické metody K-means. Před spuštěním tohoto algoritmu je opět nutno zvolit pevný počet shluků a navíc je potřeba zvolit přesnost. Algoritmus postupně prochází všechny objekty a přiřazuje jim příslušnost ke každému shluku. Po vypočtení příslušností všech objektů se vypočte a porovná přesnost přiřazení objektů do shluků. Pokud je vypočtená přesnost nižší než původně zadaná hodnota algoritmus končí. Pokud nebylo dosaženo dostatečné přesnosti, algoritmus pokračuje další iterací jako je tomu u K-means.

Gustafson-Kesselův algoritmus

Jedná se o vylepšení metody Fuzzy C-means. Základní problém Fuzzy C-means je ten, že počítá s kruhovým tvarem shluků. Ne vždy je však kruhový tvar shluku žádoucí. Gustafson-Kesselův algoritmus využívá takzvané deformační matice ke změně tvaru shluku. Díky této matici může mít shluk elipsoidní tvar a nalézt tak vhodnější rozdělení objektů. Více informací k algoritmu viz [31] a [13].

Shlukování pomocí kostry grafu

Jedná se o jednu z nejlehčích nehierarchických metod shlukování. Metoda je založena na hledání minimální kostry grafu. Graf *G* je v tomto případě reprezentován úplným grafem vytvořeným z objektů určených ke shlukování. Objekty určené ke shlukování představují tedy uzly grafu. Hrany grafu jsou spojnice mezi jednotlivými uzly. Ohodnocení hran představuje vzdálenost dvou uzlů od sebe.

Minimální kostra grafu je takový podgraf grafu G, který je stromem. Je souvislý, obsahuje všechny uzly grafu a suma všech vybraných hran má minimální ohodnocení. To znamená, že jsou vybrány takové hrany, které mají nejkratší vzdálenost mezi dvěma uzly a netvoří cyklus. Počet hran vybraných do kostry grafu je vždy $n\ (n - 1) / 2$, kde n reprezentuje počet uzlů grafu. Pro nalezení minimální kostry grafu existuje mnoho různých algoritmů. Více o hledání kostry je uvedeno v [32].

Z minimální kostry grafu je velice jednoduché nalezení shluků. Pro získání k shluků stačí odebrat $(k - 1)$ nejdelších hran v kostře. Stromový graf se tak rozpadne na k podstromů a tyto podstromy představují jednotlivé shluky. Metoda je též popsána v knize a [13].

3.2.2 Metody s proměnným počtem shluků

MacQueenova metoda se dvěma parametry

Jedná se o modifikaci klasické metody K-means, která je rozšířena o možnost zvýšit nebo naopak snížit počet shluků zadaných na začátku této metody. Je to možné, díky dvěma dodatečným parametrům označovaných písmeny *R0* a *C*. Parametr *R0* představuje rozdělovací konstantu a parametr *C* představuje slučovací konstantu.

Při inicializaci této metody je tedy nutné zvolit počet shluků *k*, parametr *R0* a *C*. Začátkem metody je tedy stejně jako u K-means zvoleno *k* centroidů, které reprezentují těžiště shluků. Po tomto rozdělení se zkontroluje, zda vzdálenost mezi dvěma centroidy není menší než hodnota *C*. Pokud ano, znamená to, že dva centroidy jsou příliš blízko sebe a měly by být sjednoceny. Spočítá se nové těžiště z těchto bodů. Slučování centroidů se opakuje do té doby, dokud nejsou všechny centroidy od sebe vzdáleny o hodnotu větší nebo rovnu parametru *C*. Následně se provede přiřazení objektů k centroidům.

Pokud je vzdálenost objektu od centroidu kratší než *R0,* je objekt přiřazen k tomuto shluku. Následně je přepočítána pozice centroidu, zkontroluje se, zda není porušena podmínka vzdálenosti dvou centroidů. Pokud však je vzdálenost od centroidu větší než *R0*, objekt se sám stává centroidem.

Nad takto vzniklou množinou centroidů použijeme MacQueenovu metodu pro pevný počet shluků. Viz seznam literatury [33] a [3].

Wishartova metoda RELOC

U této metody jsou potřebné čtyři vstupní parametry. Těmito parametry jsou:

- *THRESH* – vzdálenostní práh
- *MINSIZ* – minimální počet objektů ve shluku
- *MINC* – minimální počet shluků
- *MAXIT* – maximální počet iterací

Začátkem metody je potřeba vytvořit počáteční rozdělení objektů do *k* shluků, přičemž *k* > *MINC*. K vytvoření těchto shluků lze použít libovolnou metodu. Těžiště shluků představují centroidy. Po tomto prvotním rozdělení se stále opakují dva kroky (přiřazování objektů a rozpouštění shluků), dokud se nenalezne stabilní rozdělení nebo dokud počet iterací nepřekročí hodnotu *MAXIT*.

Přiřazování objektů do shluků

V této části metody se přiřazují objekty do shluků. Vypočte se vzdálenost objektu od všech centroidů. Pokud je nejkratší vzdálenost od těžiště menší než hodnota parametru *THRESH,* je objekt umístěn do skupiny objektů patřící nejbližšímu centroidu. V opačném případě, tedy pokud vzdálenost od nejbližšího centroidu není menší než hodnota *THRESH,* je objekt umístěn do speciální skupiny objektů, takzvaných zbytků. U shluků, ve kterých došlo ke změně, se přepočítá pozice centroidu. Ve skupině zbytků se střed nepočítá.

Rozpuštění shluků

Postupně procházíme všechny utvořené shluky a kontrolujeme počet objektů ve shluku. Pokud shluk obsahuje menší počet objektů, než je hodnota *MINSIZ,* je rozpuštěn. Všechny jeho objekty jsou tedy přiřazeny do skupiny zbytků.

V každém cyklu mají všechny objekty možnost připojit se k některému shluku, a to i objekty ve skupině zbytků. Tento postup zaručuje slučování dvou blízkých shluků do jednoho a tím snižovat počet shluků. Pokud je dosaženo počtu shluků odpovídajícího hodnotě *MINC*, shlukování je ukončeno, v opačném případě pokračujeme další iterací. Problematice se věnuje též [33] a [3].

Metoda ISODATA (Iterative Self-Organizing Data Analysis Techniques)

Jedná se o poměrně komplexní metodu shlukování. Metoda je schopna shluky jak spojovat, tak je i rozdělovat. Proti dosud popsaným metodám je nejsložitější. Obsahuje velké množství nastavitelných parametrů, které se dělí do dvou typů, na parametry konstantní a na parametry proměnné v průběhu shlukování. Parametry ovlivňují dobu běhu algoritmu, určují podmínky pro slučování a rozkládání shluků a mnohé další.

Konstantní parametry:

- *SPH* – počáteční kulovitost (sphericity) – obvykle hodnota 1,25
- *SPHCH* – maximální počet změn kulovitosti
- *INTERMAX* – počet iterací
- *NPARTS* – počet dělení v jedné iteraci
- *NRWDSD* – očekávaný počet shluků
- *MINACF* – minimální akceptovatelná změna hodnoty funkcionálu kvality rozkladu mezi dvěma za sebou následujícími iteracemi

Proměnné, které lze za běhu upravovat:

- *MINSIZ* – minimální počet objektů ve shluku
- *THETAC* – minimální vzdálenost těžišť shluků
- *NCLST* – maximální počet sloučení shluků

Začátkem algoritmu se nastaví všechny parametry a určí se centroidy. Po této inicializační časti nastává iterační část, která se dá rozdělit do několika částí. Na začátku iterační části se rozdělí objekty Forgyovou metodou. Metoda se ukončí nejpozději dosažením hodnoty *NPARTS*.

Následuje fáze, kde se projdou všechny vzniklé shluky. Zkontroluje se, zda počet objektů ve shluku není menší než hodnota parametru *THETAC*. Pokud je počet objektů menší než *THETAC,* shluk je rozpuštěn.

Další fází je slučování a rozdělování shluků. Ke slučování shluků dochází v případě, že počet shluků je větší než dvojnásobek očekávaného počtu shluků. Slučování shluků se provádí tak, že si nejprve spočítáme vzdálenosti mezi všemi centroidy a porovnáme je. Pokud některé dva centroidy shluků mezi sebou mají menší vzdálenost, než je definováno v parametru *THETAC,* shluky spojíme do jednoho. Tomuto nově vzniklému shluku se vypočítá nové těžiště ze všech objektů obsažených ve shluku. Toto těžiště se prohlásí novým centroidem. Postup slučování se opakuje, dokud není dosaženo hodnoty parametru *NCLST* nebo dokud již nejsou žádné dva centroidy, které by měly mezi sebou menší vzdálenost, než je hodnota parametru *THEPAC* .

K rozdělení shluků dochází, pokud je počet shluků menší, než polovina očekávaného počtu shluků. Pokud toto nastane, vypočítá se střední odchylka všech znaků objektů v daném shluku. Z těchto vypočítaných středních odchylek se vybere ta největší. Podle tohoto znaku se bude shluk dělit. Střední hodnota vybrané odchylky se bere jako dělící bod. Každý objekt, který má hodnotu znaku větší, než je střední hodnota vybrané odchylky, patří do jednoho shluku, zbylé objekty patří do druhého shluku. Vznikly tedy dva shluky. Pro každý tento shluk vypočítáme jeho centroid jako těžiště objektů ve shluku. Nyní zkontrolujeme, zda vzdálenost nových centroidů není menší než 1.1 definované vzdálenosti *THETAC*. Pokud je vzdálenost mezi centroidy menší, znamená to, že jsou shluky příliš blízko sebe a budeme tedy pracovat s původní skupinou.

Iterativní část algoritmu končí v případě, že bylo dosaženo maximálního počtu iterací *ITERMAX* nebo dvě po sobě jdoucí shlukování skončí se stejným výsledkem. V opačném případě jde algoritmus do dalšího cyklu začínajícího opět Forgyovou metodou.

Metoda řeší většinu problémů, které mají ostatní metody. Nevýhodou této metody je velké množství parametrů, které je potřeba nastavit. Pro nastavení je třeba velmi dobré znalosti o množině dat, kterou shlukujeme a mít představu o výsledku, kterého chceme dosáhnout. Pro metodu je tedy potřeba mít zkušeného analytika, který zadá rozumné hodnoty parametrů. Problematice se věnuje též [3].

Metoda CLASS

Jedná se o modifikaci metody *ISODATA*. Metoda zjednodušuje nastavení parametrů a nároky na analytika. Parametry, které je u této metody nutné nastavit před začátkem běhu, jsou:

- *GAMA* – maximální počet iterací
- *THETAN* – minimální počet prvků ve shluku
- *Sn* – parametr řídící rozklady shluků – parametr se při každém cyklu zvyšuje o 1 a zpřísňuje se tak kritérium rozkladu

Ostatní parametry jsou počítány automaticky. Metoda, stejně jako u předchozí metody, končí při dosažení maximálního počtu iterací GAMA, případně pokud dvě po sobě jdoucí shlukování skončí se stejným výsledkem. Stručný popis algoritmu je možné nalézt též v [3].

4 Metriky vyhodnocení shlukových metod

Jedná se o metody vyhodnocující shlukovaná data. Existují různé typy vyhodnocení výsledku shlukových metod. Některé metriky hodnotí platnost shluku, jiné metriky hodnotí, jak dobře jsou prvky zařazeny do shluků, další metriky zase porovnávají výsledné shlukování dvou různých shlukových metod, jiné pomáhají určit správný počet shluků. Metrik vyhodnocení je velké množství a v této práci tedy nebudou popsány všechny existující typy. Věnovat se budeme převážně metodám, které budou implementovány jako rozšíření pro RapidMineru.

4.1 Rand Index

Rand index nebo též Rand míra se používá ve shlukové analýze pro porovnání výsledků dvou shlukových metod na stejném souboru dat. Vybereme si soubor dat a dvě různé metody shlukování. Necháme soubor dat zpracovat oběma shlukovými metodami a výsledná shlukovaná data obou metod porovnáme. Rand index následně představuje číselnou hodnotu, která udává, jak moc jsou si výsledné shluky podobné nebo naopak nepodobné.

Postup výpočtu Rand indexu:

1. Vezmeme množinu všech n objektů $S = \{o_1, o_2, \dots, o_n\}$ zadaného souboru dat. Dále máme množinu shluků $X = \{X_1, \dots, X_r\}$ první shlukové metody, kde r představuje počet shluků první shlukové metody a množinu shluků $Y = \{Y_1, \dots, Y_s\}$ druhé shlukové metody, kde s představuje počet shluků druhé shlukové metody.
2. Následně vybírám všechny dvojce objektů množiny S a porovnávám jejich zařazení do shluků. Zjistím hodnoty a, b, c a d takto:
    - a je počet párů objektů z množiny S, kde jsou objekty páru ve stejném shluku v množině X a ve stejném shluku v množině Y
    - b je počet párů objektů z množiny S, kde jsou objekty páru v různých shlucích v množině X a v různých shlucích v množině Y
    - c je počet párů objektů z množiny S, kde jsou objekty páru ve stejném shluku v množině X a v různých shlucích v množině Y
    - d je počet párů objektů z množiny S, kde jsou objekty páru v různých shlucích v množině X a ve stejném shluku v množině Y
3. Výsledná hodnota Rand indexu se následně spočítá jako

$$I_R = \frac{a + b}{a + b + c + d}$$

Rand index je hodnota mezi 0 a 1. Pokud se výsledný Rand index blíží spíše k hodnotě 1, jsou data oběma shlukovými metodami rozděleny do stejných shluků. Pokud se výsledný Rand index blíží spíše k hodnotě 0, jsou data oběma shlukovými metodami rozděleny do různých shluků.

Vzorce a informace k výpočtu Rand index lze nalézt též v [35] a [36].

4.2 Dunn Index

Dunn index nebo též nazývaný jako separační index je metrika pro hodnocení shlukových metod. Jde o metriku vnitro-shlukové evaluace shlukových metod. Cílem indexu je identifikovat sady shluků, které jsou kompaktní, s malým rozptylem mezi objekty ve shluku a dobře oddělené shluky mezi sebou. To znamená, že dva různé shluky jsou dostatečně daleko od sebe, tedy objekty jednoho shluku jsou znatelně odděleny od objektů jiného shluku. Naopak objekty ve stejném shluku jsou u sebe blízko. Hodnota Dunn indexu se používá v některých výpočetních systémech jako metoda pro stanovení správné hodnoty počtu shluků.

Postup výpočtu Dunn indexu:

1. Nejprve se zjistí minimální vzdálenost mezi shluky. Tato vzdálenost se zjistí tak, že nalezneme dva objekty z různých shluků takové, že vzdálenost mezi těmito shluky je minimální.
2. Dále se hledá maximální vnitro-shluková vzdálenost. Naleznu tedy dva objekty, které patří do stejného shluku, a zároveň je jejich vzdálenost od sebe maximální.
3. Výsledná hodnota Dunn indexu se následně spočítá jako podíl minimální mezi-shlukové vzdálenosti a maximální vzdálenosti uvnitř shluku.

Matematicky můžeme výpočet Dunn indexu popsat jako:

- vzdálenost mezi shluky

$$D_{h,h'} = \min_{x_i \in C_h,\, x_j \in C_{h'}} D(x_i, x_j)$$

- poloměr shluku

$$diam_h = \max_{x_i, x_j \in C_h} D(x_i, x_j)$$

- výsledný Dunn index je dán vzorcem

$$I_D = \min_{1 \le h \le k} \left\{ \min_{1 \le h' \le k} \frac{D_{h,h'}}{\max\limits_{1 \le l \le k} (diam_l)} \right\},$$

kde k značí počet shluků a C_i značí vybraný i-tý shluk.

Vysoké hodnoty Dunn indexu představují dobře rozdělené a kompaktní shluky. Nízké hodnoty indexu pak reprezentují špatně vytvořené shluky, kde podobnost dvou objektů různých shluků může být velice vysoká. Vzorce převzaty z [13].

4.3 Silueta

Jde o metriku zkoumající vnitro-shlukové ohodnocení. Metrika spočítá pro každý objekt číslo zvané silueta. Silueta představuje hodnotu, jak moc vybraný objekt patří do shluku, do kterého je zařazen shlukovou metodou. Hodnota siluety se spočítá pro každý objekt ve shluku. Ze všech vypočtených hodnot siluety v daném shluku se spočítá průměrná hodnota siluety pro vybraný shluk. Tato průměrná hodnota nese informaci o tom, jak dobře je shluk vytvořený.

Postup výpočtu siluety pro jeden objekt se skládá z několika částí.

1. Vybereme si nejprve objekt x_i ze shluku X. Pro vybraný objekt x_i spočítáme průměrnou vzdálenost h_{x_i} od všech ostatních objektů ve shluku X. Má-li shluk X pouze jeden objekt, tak $h_{x_i} = 0$.
2. Naleznu nejbližší shluk k objektu x_i a to tak, že postupně vybíráme všechny shluky. Pro každý objekt ve vybraném shluku spočítáme jeho vzdálenost od objektu x_i a následně z vypočtených vzdáleností spočítáme průměrnou vzdálenost vybraného shluku od objektu x_i. Z vypočtených průměrných vzdáleností shluků vybereme tu nejmenší. Tak určíme nejbližší shluk k vybranému objektu x_i. Nejbližší shluk označíme písmenem Y. Průměrná vzdálenost objektu x_i a shluku Y se označí h_Y.
3. Silueta pro objekt x_i se vyčíslí následovně:
    - pokud má shluk X pouze jeden objekt, je hodnota siluety $s = 0$
    - pokud je průměrná vnitro-shluková vzdálenost menší než průměrná vzdálenost od nejbližšího shluku, tedy $h_{x_i} < h_Y$, je hodnota siluety $s = 1 - \frac{h_{x_i}}{h_Y}$
    - pokud je průměrná vnitro-shluková vzdálenost větší než průměrná vzdálenost od nejbližšího shluku, tedy $h_{x_i} > h_Y$, je hodnota siluety $s = \frac{h_Y}{h_{x_i}} - 1$
    - pokud se průměrná vnitro-shluková vzdálenost rovná průměrné vzdálenosti od nejbližšího shluku, tedy $h_{x_i} = h_Y$, je hodnota siluety $s = 0$

Hodnota siluety se pohybuje v intervalu od hodnoty -1 do hodnoty +1.

- V případě, že se hodnota siluety s objektu x_i blíží k hodnotě +1, je objekt x_i správně zařazen do shluku X. Vzdálenost objektu x_i od ostatních objektů ve shluku X je nízká v porovnání vzdálenosti od objektů nejbližšího shluku Y.
- V případě, že se hodnota siluety s objektu x_i blíží k hodnotě 0, je objekt x_i umístěn na pomezí shluků X a Y. Vzdálenost od objektů ve shluku X a objektů shluku Y je velice podobná. Objekt x_i byl zařazen do shluku X pouze náhodou.
- V případě, že se hodnota siluety s objektu x_i blíží k hodnotě -1, je objekt x_i nesprávně zařazen do shluku X. Vzdálenost objektu x_i od ostatních objektů ve shluku X je vysoká v porovnání vzdáleností od objektů nejbližšího shluku Y. Otázkou zůstává, proč byl objekt x_i zařazen do shluku X a ne do nějakého bližšího shluku.

Jak již bylo řečeno, hodnota siluety se určí pro každý objekt shlukovaného souboru dat. Pro stanovení průměrné hodnoty siluety ve shluku, se spočítá průměrná hodnota siluet objektů v daném shluku. Další hodnotou, která se používá k určení, jak dobře shlukování dopadlo, je průměrná hodnota siluety přes všechny objekty v zadaném souboru dat. Průměrná hodnota siluety celého souboru dat je dobrým vodítkem pro stanovení správného počtu shluků. Další informace možné nalézt v [29] a [34].

4.4 Typické hodnoty

Typické hodnoty, představují takové hodnoty, které jsou nejčastější neboli typické pro daný soubor dat. Určení typické hodnoty se dělí podle typu vstupních dat. Jiný postup určení typických hodnot je pro nominální data a jiný pro data číselná.

V případě, že máme nominální data, typická hodnota je reprezentovaná nejčetněji hodnotou v daném souboru dat. Stačí tedy projít všechna data a nalézt hodnotu, která se nejčastěji opakuje. Pokud budeme chtít určit typickou hodnotu souboru {pes, medvěd, slon, medvěd, kočka, pes, had, slon, medvěd} bude jí medvěd, protože se v souboru objevil nejvícekrát. Může nastat, že dvě nebo více hodnot je v souboru použito stejně často. V takovém případě máme více typických hodnot.

Pokud budeme mít soubor dat složený z náhodných veličin, je určení typické hodnoty o něco složitější. Typická hodnota souboru dat pak může být reprezentována jako aritmetický průměr, medián nebo modus. Podrobnější informace možné nalézt viz [38],[39] a [37].

5 Rapidminer

RapidMiner je softwarová platforma, která poskytuje prostředí pro strojové učení, dolování z dat, dolování z textu, prediktivní analýzu, business analýzu, shlukování a další. Používá se nejen v průmyslu a obchodu, ale i ve výzkumu, vzdělání, prototypování a vývoji aplikací. Podporuje všechny kroky procesu dolování dat včetně vizualizace, validace a optimalizace výsledků. Rané verze softwaru RapidMiner jsou vydávané pod OSI-certified open source licence. Zdrojové kódy jsou tedy zdarma dostupné ke stažení na serveru Sourceforge.com až po verzi RapidMiner 5.3. Od verze RapidMiner 6 již zdrojové kódy veřejně dostupné nejsou.

Pro více informací o RapidMineru doporučuji navštívit oficiální webové stránky www.RapidMiner.com, případně knihu [40].

6 Implementace vlastních operátorů

Rozšířit RapidMiner jde hned několika způsoby. Pokud chceme přidat jen velmi jednoduchou funkcionalitu do RapidMineru, můžeme použít takzvaný Scripting Operator. Jedná se o operátor implementovaný přímo v RapidMineru. Do operátoru je možné napsat velmi jednoduchý skript, který RapidMiner rovnou provede. Pro skript operátoru se používá skriptovací jazyk Groovy.

Pokud budeme chtít přidat nějaké složitější operátory nebo nové zobrazení výsledků různých výpočtů, tak nám již Scripting Operator nemusí stačit. V tom případě budeme muset rozšířit přímo RapidMiner nebo vytvořit plugin pro RapidMiner.

6.1 Zdrojové kódy RapidMineru

Jednou možností je stažení kompletních zdrojových kódů RapidMineru a ty rozšířit o další funkcionalitu. RapidMiner je distribuovaný pod licencí Affero General Public License, je tedy možné stáhnout zdrojové kódy z internetu až do verze 5.3. V současné době jsou kódy ke stažení hned na několika místech. Oficiálním místem ke stažení je sever SourceForge.net[1].

Přehledný popis, jak stáhnout zdrojové kódy z internetu, včetně importu staženého projektu do vývojového prostředí a spuštění projektu ve vývojovém prostředí, je možné nalézt na komunitních webových stránkách RapidMineru[2].

Dalším místem pro získání kompletních zdrojových kódů RapidMineru je server GitHub.com[3]. Taktéž se jedná o oficiální zdroj. U zdrojových kódu je i krátký popis pro různé operační systémy a jak na nich stažené zdrojové kódy spustit.

Neoficiálních verzí RapidMineru, případně částečně upravených verzí RapidMineru, je na internetu možné dohledat velké množství. Pokud budete chtít sami rozšiřovat RapidMiner o vlastní operátory, doporučují kromě čistých zdrojových kódů stažených z oficiálního zdroje stáhnout i upravenou verzi RapidMineru s podporou pro editaci již existujících operátorů. Jde o verzi vydanou TU Dortmund University. Tato verze RapidMineru má možnost pro jakýkoliv operátor otevřít přímo jeho implementaci ve vývojovém prostředí. To znamená, že pokud si vybereme v RapidMineru nějaký operátor, můžeme přímo ze spuštěné aplikace spustit vývojové prostředí, ve kterém se nám otevře přímo zdrojový kód vybraného operátoru. Vybraný operátor mohu následně editovat, případně se podívat, jak

[1] Přímý odkaz pro stažení zdrojových kódů RapidMineru ze serveru SourceForge.net: http://sourceforge.net/projects/rapidminer/

[2] Přímý odkaz na komunitní stránky RapidMineru s návodem na stažení zdrojových kódů: http://community.rapidminer.com/eclipse/

[3] Přímý odkaz pro stažení zdrojových kódů RapidMineru ze serveru GitHub.com: https://github.com/rapidminer

je napsaný a inspirovat se tak pro tvorbu vlastního operátoru. Toto se jeví jako nápomocné pro pochopení, jakým stylem operátory pracují a s integrací vlastních operátorů. Návod na stažení zdrojových kódů a spuštění této verze RapidMineru je popsán na webových stránkách TU Dortmund University[4].

6.2 Šablona pluginu pro RapidMiner

Šablonu pluginu pro RapidMiner je vhodné použít v případě, že chceme rozšiřovat RapidMiner pouze o nové operátory, případně o nová okna pro zobrazení dat. Pokud budeme chtít změnit již existující operátory, přepsat zobrazení dat nebo změnit vláknové zpracování operátorů, budeme muset stáhnout kompletní zdrojové kódy RapidMineru a implementovat změny přímo v kódu. Kde získat kompletní zdrojové kódy RapidMineru je popsáno v předchozí kapitole.

Šablona pluginu pro RapidMiner je ke stažení na serveru GitHub.com[5]. Rovněž je na GitHub.com popsáno, jak správně šablonu importovat do vývojového prostředí. Základní informace o psaní pluginu pro RapidMiner je možné nalézt v dokumentu „How to extend RapidMiner 5“ viz [16]. Dokument je možné stáhnout na oficiálních webových stránkách RapidMineru.

6.3 Implementované operátory

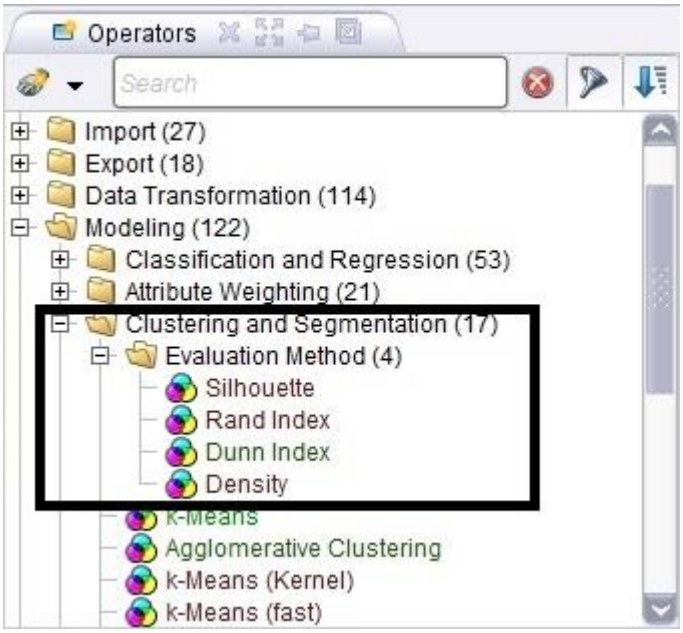
Tato diplomová práce se týká vývoje operátorů pro vyhodnocení shlukové analýzy, vnitro-shlukovou evaluaci, porovnávání shlukových metod a zjišťování typických hodnot. Z tohoto důvodu se přesně hodí vyvíjet plugin pro RapidMiner.

Má diplomová práce rozšiřuje RapidMiner o čtyři operátory. Po přidání pluginu do RapidMineru je možné najít všechny implementované operátory v liště zvané „Operators“. Operátory jsou umístěny v adresářové struktuře **Modeling > Clustering and Segmentation > Evaluation Method**.

Přidané operátory jsou:

- Rand Index
- Dunn Index
- Silhouette
- Density

[4] Přímý odkaz na návod pro stažení a spuštění rozšířené verze RapidMineru o editaci existujících operátorů: http://www-ai.cs.uni-dortmund.de/SOFTWARE/RMD/index.html
[5] Přímý odkaz pro stažení RapidMineru ze serveru GitHub.com: https://github.com/rapidminer

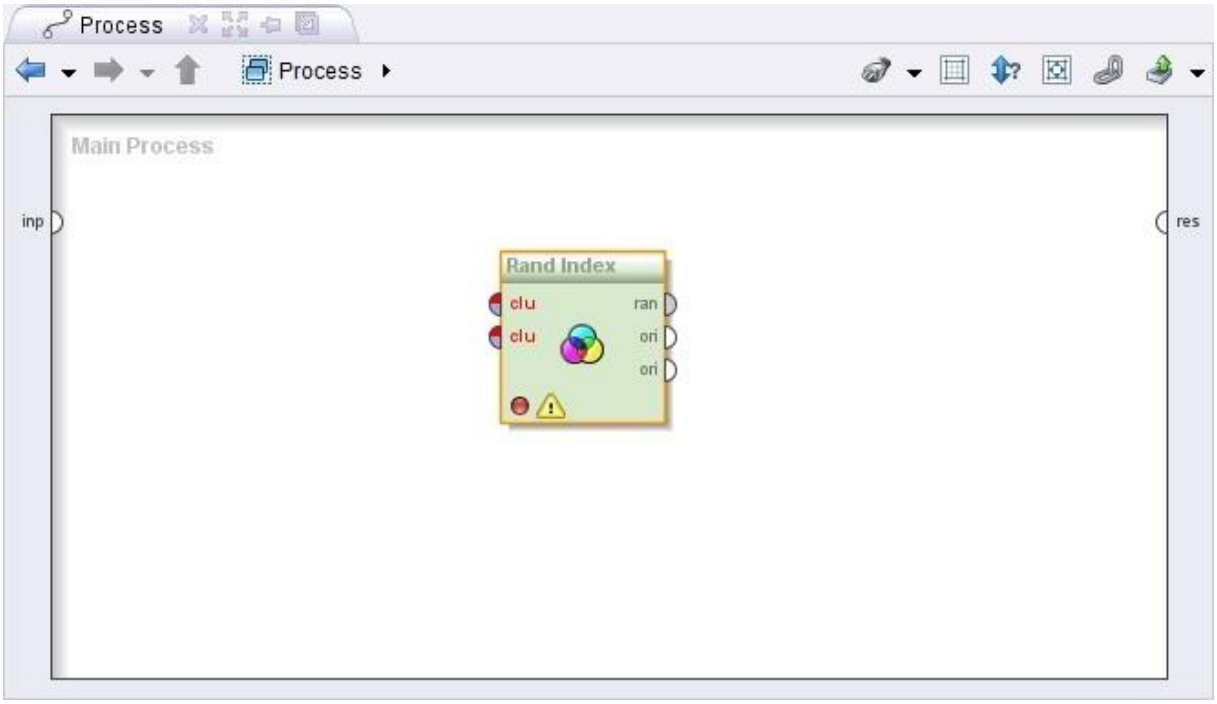


obr. 8 - Okno Operators RapidMiner

V dalších podkapitolách budou popsány jednotlivé operátory.

6.4 Rand Index operátor

Rand index slouží k porovnání výstupů dvou shlukových analýz. Podrobnější popis výpočtu Rand indexu je v předchozí kapitole 4.1.



obr. 9 - Rand Index operátor v RapidMineru

Na obr. 9 je zobrazeno, jak vypadá implementovaný Rand Index operátor v RapidMineru. Z obrázku je vidět, že operátor má dva vstupy a tři výstupy.

6.4.1 Vstupní porty Rand Index operátoru

Rand index operátor má dva vstupní porty. Vstupní porty jsou pojmenované:

- cluster set A
- cluster set B

Cluster set A i **cluster set B** očekávají soubory dat, která jsou výstupem z různých shlukových analýz. Vstupní porty mají naprogramovanou kontrolu vstupních dat. To tedy znamená, že vstupní datová sada musí být objektem zvaným *ExampleSet*[6] a musí obsahovat atribut *cluster*. V případě, že vstupní data nejsou objektem *ExampleSet*, případně data neobsahují atribut *cluster*, RapidMiner uživatele informuje, že výpočet Rand indexu nebude možné provést. O problémech je uživatel informován v okně **Problems**, viz obr. 10.

[6] ExampleSet je objekt, který v RapidMineru obsahuje kompletní data uložené do tabulky a navíc obsahuje meta informace ke každému sloupci. Jedná se o jeden z hlavních objektů pro data s tabulkovou strukturou. S tímto objektem pracuje většina operátorů v RapidMineru, které zpracovávají tabulková data.

obr. 10 - Ukázka chybových hlášek

Všechny shlukové metody implementované v RapidMineru objekt *ExampleSet* s atributem *cluster* přímo vracejí. Kontrola vstupu se tedy týká převážně importovaných již shlukovaných dat. V takovém případě, by si měl uživatel RapidMineru dát pozor na správné označení sloupce, reprezentující umístění objektu do shluku.

6.4.2 Výstupní porty Rand Index operátoru

Rand index operátor má tři výstupní porty. Výstupní porty jsou pojmenované:

- rand index
- original cluster set A
- original cluster set B

Rand index výstupní port vrací hodnotu vypočteného Rand indexu. Tento port je vhodné připojit přímo na port pro výsledky. Prozatím není implementovaný žádný operátor, který by s vrácenou hodnotou uměl pracovat. Port vrací mnou implementovaný objekt *RandIndexIOObject*.

Original cluster set A a **original cluster set B** vracejí nezměněné hodnoty vstupních dat. Tyto výstupní porty jsou zde pro případ, že by uživatel chtěl kromě výpočtu hodnoty Rand indexu s daty dále pracovat nebo zobrazit výsledné hodnoty shlukování.

6.4.3 Parametry Rand Index operátoru

Rand Index operátor nemá žádné další parametry.

6.4.4 Zobrazení výstupních hodnot

Jelikož má Rand index tři výstupní porty, je možné připojit všechny tři porty na porty RapidMineru pro zobrazení výsledků. **Original cluster set A** a **original cluster set B** vracejí objekt

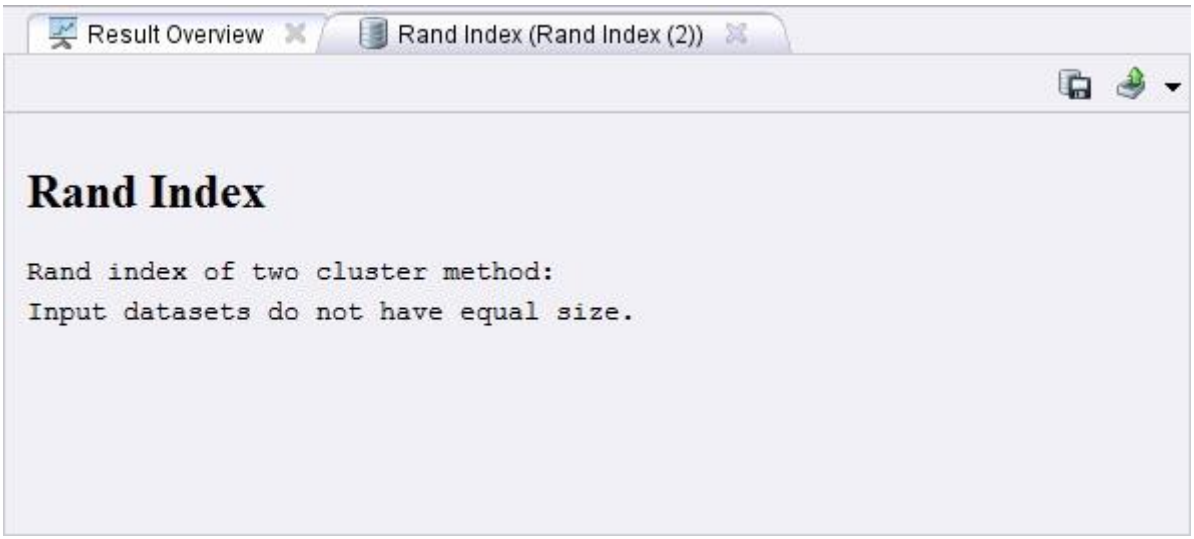
ExampleSet, pro který má RapidMiner již implementované zobrazení. Pro výstupní port **rand index** jsem implementoval vlastní textový renderer, který po doběhnutí výpočtu hodnoty Rand indexu zobrazí nové okno obsahující hodnotu vypočteného Rand indexu. Pro ukázku použiji vstupní soubor dat Iris a nechám data shlukovat pomocí shlukovací metody k-means, kde $k = 3$. Ještě jednou použiji stejnou datovou sadu a tentokrát k-means nastavím hodnotu $k = 10$. Výstupy z k-means operátorů nastavím jako vstup pro Rand Index operátor. Zobrazený výsledek si můžeme prohlédnout na obr. 11.

obr. 11 - Zobrazení hodnoty Rand index

Během výpočtu může dojít k několika výjimkám, při kterých Rand Index operátor není schopen vypočítat výslednou hodnotu. V takových případech nevypíše hodnotu Rand indexu, ale místo toho vypíše důvod, proč k výpočtu nedošlo.

Jedním z důvodů, proč k výpočtu hodnoty Rand indexu nedojde, je, že vstupní soubory dat nemají stejný počet řádků. V takovém případě se místo vypočtené hodnoty Rand indexu zobrazí zpráva „Input datasets do not have equal size.“.

Dalším důvodem, kdy k výpočtu nemusí dojít, je, že soubory dat sice mají stejný počet řádků, ale rozdílné sloupce. V takovém případě se místo vypočtené hodnoty Rand Indexu zobrazí zpráva „Input datasets do not have equal elements.“.



obr. 12 - Zobrazení Rand indexu v případě rozdílných vstupních souborů dat

6.4.5 Implementace Rand Index operátoru

Nejdůležitější třídy pro můj Rand Index operátor implementované v pluginu jsou:

- com.rapidminer.operator.*RandIndex.java*
- com.rapidminer.operator.*RandIndexData.java*
- com.rapidminer.operator.gui.*RandIndexIOObject.java*

Třída RandIndex

RandIndex představuje třídu, která dědí abstraktní třídu RapidMineru com.rapidminer.operator.*Operator*. Díky dědění třídy *Operator* je třída *RandIndex* braná jako operátor pro RapidMiner.

Aby mohl RapidMiner správně pracovat s operátorem Rand Index, je nutné ve třídě implementovat několik povinných částí. Nejprve je nutné vytvořit vstupní a výstupní porty. Pro vytvoření portů se používají právě metody z předka, a to konkrétně *getInputPorts()* a *getOutputPorts()*. Metody vracejí objekt reprezentující porty nově vytvořeného operátoru. Do tohoto objektu se následně vytvoří nové porty pro operátor Rand Index. Pro Rand Index operátor je tedy potřeba vytvořit dva vstupní porty reprezentující **cluster set A** a **cluster set B**. Dále je potřeba vytvořit tři výstupní porty. Jeden port pro vypočtenou hodnotu Rand indexu. Dva porty pro vrácení originálních nezměněných souborů dat ze vstupu.

Další povinnou částí je konstruktor, který přijímá parametr OperatorDescription. Tento konstruktor je volaný RapidMinerem v případě, že je v RapidMineru použit operátor. V konstruktoru se

kromě volání konstruktoru předka stanovují i pravidla pro vstupní a výstupní porty. Pro Rand Index operátor jsou stanovena následující pravidla:

- vstupní port **cluster set A** přijímá ExampleSet, a to pouze v případě, že obsahuje atribut cluster
- vstupní port **cluster set B** přijímá ExampleSet, a to pouze v případě, že obsahuje atribut cluster
- výstupní port **rand index** nemá zvláštní pravidla
- výstupní port **original cluster set A** lze použít pouze v případě, že je připojený vstupní port **cluster set A**
- výstupní port **original cluster set B** lze použít pouze v případě, že je připojený vstupní port **cluster set B**

Poslední částí třídy *RandIndex* je velmi důležitá metoda *doWork()*. Metoda je volána RapidMinerem v případě, že je spuštěn výpočet. Jedná se tedy o metodu, která se stará o načtení dat ze vstupních portů, o zpracování dat, o výpočet výsledků a následné odeslání spočtených dat na výstupní porty.

Třída RandIndexData

Třída *RandIndexData* představuje třídu, která provádí samotný výpočet hodnoty Rand indexu. O výpočet se stará metoda *calculateRandIndex(ExampleSet clusterSetA, ExampleSet clusterSetB)*. Metoda nejprve zkontroluje velikost obou vstupních datových souborů. V případě, že se velikosti liší, rovnou výpočet ukončí, protože je jasné, že dodané soubory dat jsou různé. V opačném případě se pokračuje ve výpočtu hodnoty Rand indexu. Postup výpočtu hodnoty Rand indexu je popsán v kapitole 4.1.

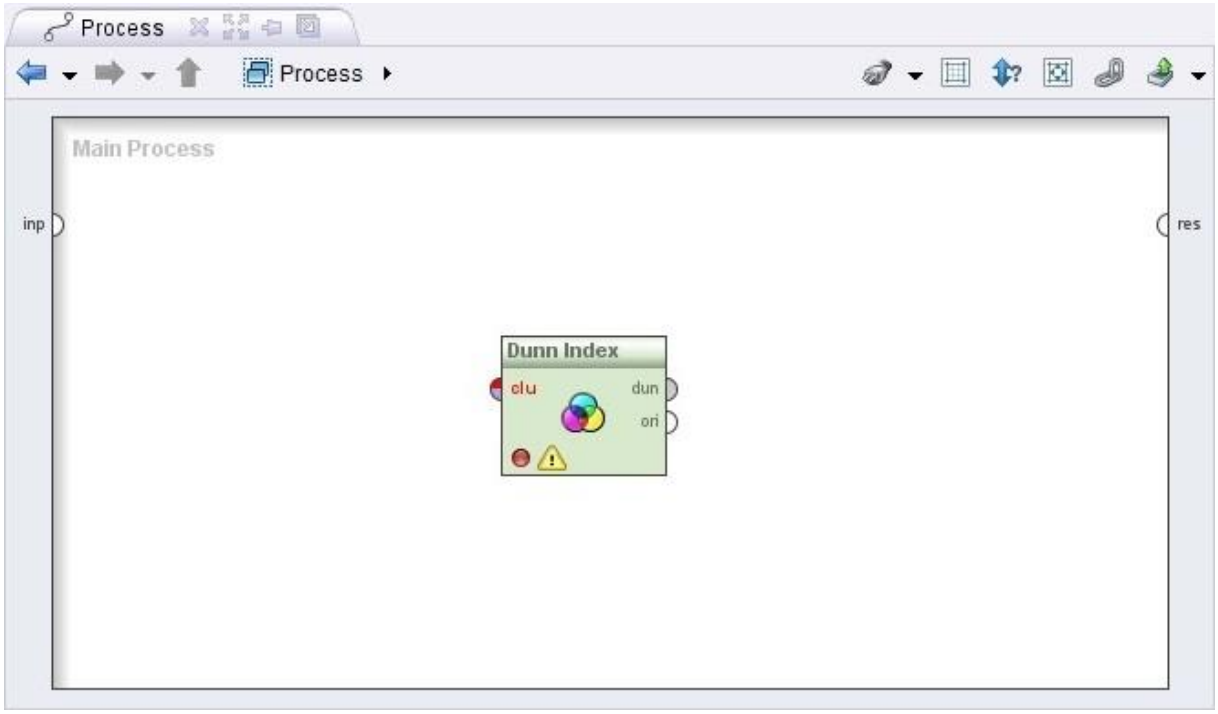
RandIndexData rovněž představuje datový objekt pro třídu *RandIndexIOObject*, který se stará o vykreslení výsledné hodnoty.

Třída RandIndexIOObject

Třída *RandIndexIOObject* se stará o načtení, zpracování a formátování vypočtených výsledků, aby mohly být RapidMineren vykresleny. Aby RapidMiner mohl po vypočítání výsledku vykreslit data pomocí třídy *RandIndexIOObject*, je potřeba, aby třída *RandIndexIOObject* dědila *ResultObjectAdapter*. *ResultObjectAdapter* představuje základní abstraktní nadtřídu pro vykreslování textových výsledků. Je potřeba překrýt metodu *toResultString()*, která vrací již naformátovaný výsledek. Výsledné naformátování je vidět na obr. 11 a obr. 12.

6.5 Dunn Index operátor

Hodnota Dunn indexu reprezentuje, jak moc jsou shluky kompaktní a dobře oddělené. Výpočet Dunn indexu je podrobněji popsán v předchozí kapitole 4.2. Tato kapitola se týká Dunn Index operátoru implementovaného pro RapidMiner.



obr. 13 - Dunn Index operátor v RapidMineru

Obr. 13 ukazuje, jak vypadá implementovaný Dunn Index operátor v RapidMineru. Dunn Index operátor má jeden vstup a dva výstupy. Vstup a výstupy budou podrobněji popsány níže.

6.5.1 Vstupní porty Dunn Index operátoru

Dunn Index operátor má jeden vstupní port. Vstupní port je pojmenovaný **cluster set**. Vstupní port **cluster set** očekává na vstupu objekt typu *ExampleSet*. Jedná se o standardní objekt v RapidMineru, který reprezentuje data v tabulkové podobě. Dále je potřeba, aby data vložená v objektu *ExampleSet* obsahovala sloupec *cluster*. V opačném případě RapidMiner vypíše uživateli hlášku s problémem, který nastal. Dunn Index operátor má implementovanou stejnou kontrolu vstupu, jak bylo popsáno v podkapitole popisující Rand Index operátor.

6.5.2 Výstupní porty Dunn Index operátoru

Dunn Index operátor má dva výstupní porty pojmenované:

- dunn index
- original cluster set

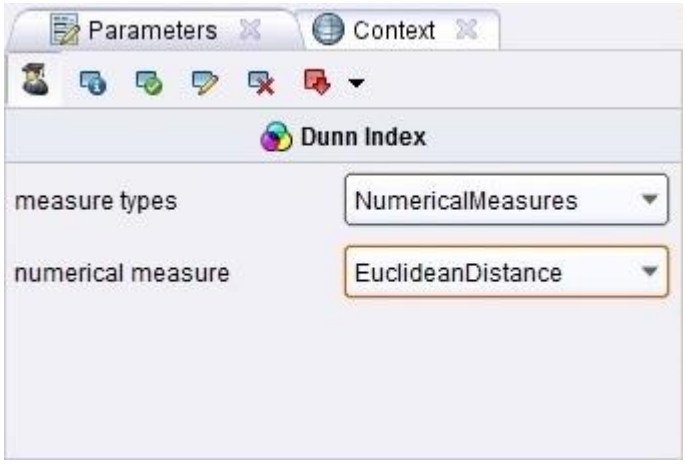
Dunn index výstupní port vrací hodnotu vypočítaného Dunn indexu. Tento port je vhodné připojit přímo na port pro výsledky. Prozatím není implementovaný žádný operátor, který by s vrácenou hodnotou uměl dále pracovat. Port vrací objekt *DunnIndexIOObject*.

Original cluster set vrací nezměněné hodnoty vstupních dat. Tyto výstupní porty jsou zde z důvodu, pokud by uživatel chtěl kromě výpočtu hodnoty Dunn indexu, s daty dále pracovat nebo zobrazit výsledné hodnoty shlukování.

6.5.3 Parametry Dunn Index operátoru

Po kliknutí na Dunn Index operátor přidaného do okna „Process“ v RapidMineru můžeme nastavit celkem dva parametry. Viz obr 14.

- measure type
- <type> measure



obr. 14 - Parametry Dunn Index operátoru

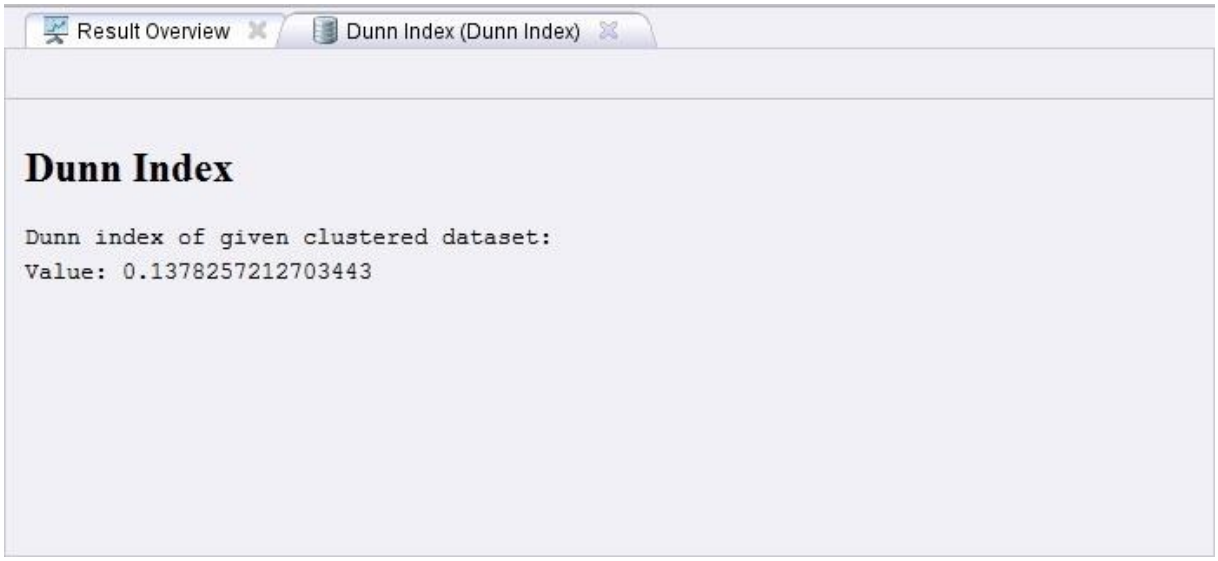
Measure type představuje typy měr, které chceme pro výpočet podobnosti objektů použít. Typ míry by měl odpovídat typu dat, která očekáváme na vstupu. Na výběr jsou celkem čtyři míry.

Hodnoty i název druhého parametru se odvíjí od zvolené hodnoty parametru **Measure type**. Je možné zvolit tyto kombinace měr:

- NumericalMeasure
  - EuclideanDistance
  - CamberraDistance
  - ChebychevDistance
  - ColerationSimilarity
  - CosineSimilarity
  - DiceSimilarity
  - DynamicTimeWarpingDistance
  - InnerProductSimilarity
  - JaccardSimilarity
  - KernelEuclideanDistance
  - ManhattanDistance
  - MaxProductSimilarity
  - OverlapSimilarity
- MixedMeasure
  - MixedEuclideanDistance
- NominalMeasure
  - NominalDistance
  - DiceSimilarity
  - JaccardSimilarity
  - KulczynskiSimilarity
  - RogersTanimotoSimilarity
  - RussellRaoSimilarity
  - SimpleMachingSimilarity
- BregmanDivergences
  - GeneralizedDivergence
  - ItakuraSaitoDistance
  - KLDivergence
  - LogarithmicLoss
  - LogisticLoss
  - MahalanobisDistance
  - SquaredEuclideanDistance
  - SquaredLoss

6.5.4 Zobrazení výstupních hodnot

Dunn Index operátor má dva výstupní porty a oba je možné připojit na porty RapidMineru pro zobrazení výsledků. **Original cluster set** vrací objekt *ExampleSet*, pro který má RapidMiner již implementované zobrazení. Pro výstupní port **dunn index** jsem implementoval vlastní textový rederer, který po doběhnutí výpočtu hodnoty Dunn indexu zobrazí nové okno obsahující hodnotu vypočteného Dunn indexu. Pro ukázku výsledného zobrazení použiji vstupní soubor dat Iris a nechám data shlukovat pomocí aglomerativní shlukové metody průměrné vazby a pro výpočet vzdálenosti mezi objekty zvolím Euklidovskou metriku. Počet shluků, do kterých se mají data shlukovat, zvolím $k = 3$. Výstup z operátoru pro shlukování dat připojím jako vstup pro Dunn Index operátor. U Dunn Index operátoru je možné zvolit metriku pro počítání podobnosti objektů. Pro ukázku zvolím, stejně jako u aglomerativní shlukové metody, euklidovskou metriku. Zobrazený výsledek si můžeme prohlédnout na obr. 15.



obr. 15 - Zobrazení hodnoty Dunn index

6.5.5 Implementace Dunn Index operátoru

Nejdůležitější třídy pro Dunn Index operátor implementované v pluginu jsou:

- com.rapidminer.operator.*DunnIndex.java*
- com.rapidminer.operator.*DunnIndexData.java*
- com.rapidminer.operator.gui.*DunnIndexIOObject.java*

Třída DunnIndex

Třída *DunnIndex* představuje základní třídu reprezentující operátor v RapidMineru. Třída dědí stejného předka a implementuje stejné metody, jak už bylo popsáno u Rand Index operátoru v popisu třídy *RandIndex*. Rozdíly oproti třídě *RandIndex* jsou dva. První je počet vstupních portů a počet výstupních portů. Na rozdíl od Rand Index operátoru má Dunn Index operátor jen jeden vstupní port pojmenovaný **cluster set** a dva výstupní porty pojmenované **dunn index** a **original cluster set.** Pro vstupní port a výstupní porty jsou stanovena stejná pravidla jako pro porty Rand Index operátoru:

- vstupní port **cluster set** přijímá *ExampleSet,* a to pouze v případě, že obsahuje atribut cluster
- výstupní port **dunn index** nemá zvláštní pravidla
- výstupní port **original cluster set** lze použít pouze v případě, že je připojený vstupní port **cluster set**

Dalším rozdílem Dunn Index operátoru oproti Rand Index operátoru je nutnost překrytí metody *getParameterTypes().* Dunn Index operátor má totiž možnost výběru dvou parametrů ovlivňující výpočet výsledné hodnoty. Metoda *getParameterTypes()* se stará o správné načtení vybraných parametrů. Podrobnější informace k parametrům Dunn Index operátoru jsou popsány v podkapitole Parametry Dunn Index operátoru 6.5.3.

Třída DunnIndexData

Třída *DunnIndexData,* obdobně jako třída *RandIndexData*, se stará o samotný výpočet hodnoty Dunn indexu. Pro výpočet se používá metoda *calculateDunnIndex(Map<String, List <Example>> separatedMap, DistanceMeasure measure).* Vstupní parametry metody jsou dva, a to *separatedMap* a *measure*. *SeparatedMap* je klasická hašovací mapa, kde klíčem jsou názvy shluků a hodnotou je seznam objektů patřících do daného shluku. *Measure* představuje metriku výpočtu vzdálenosti dvou objektů, kterou si zvolil uživatel jako parametr pro Dunn Index operátor. Postup samotného výpočtu hodnoty Dunn indexu je popsán v kapitole 4.2.

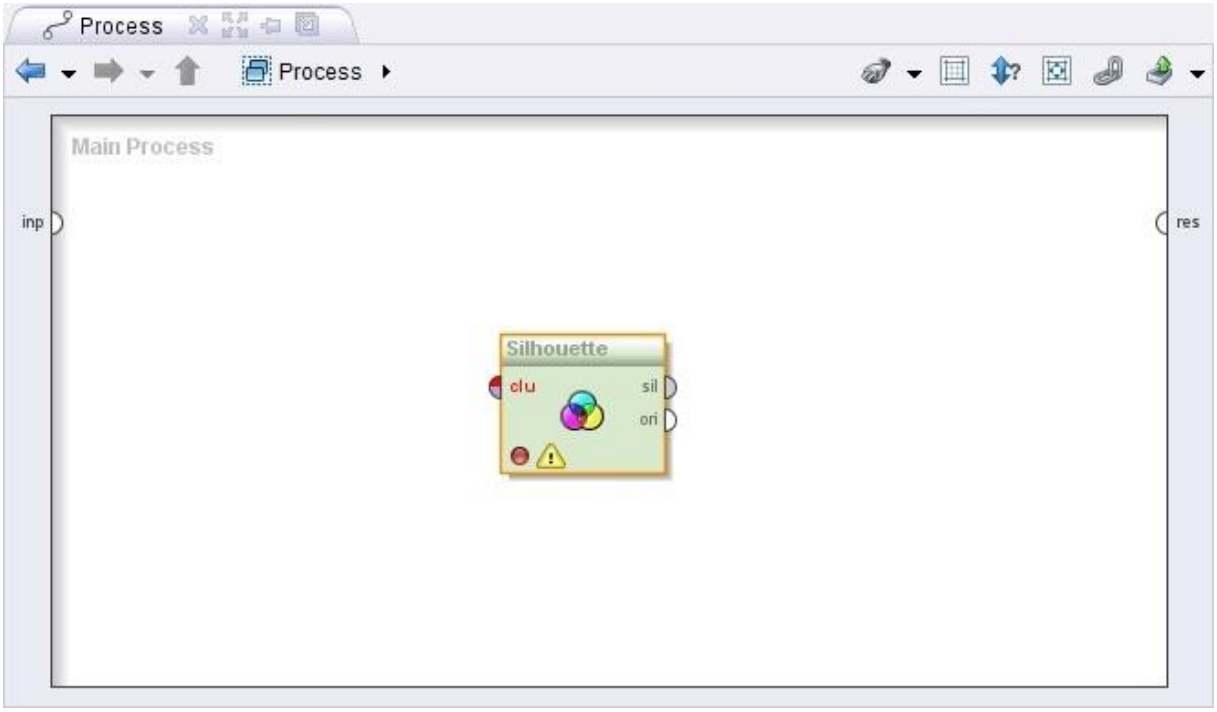
DunnIndexData rovněž představuje datový objekt pro třídu *DunnIndexIOObject*, který se stará o vykreslení výsledné hodnoty.

Třída DunnIndexIOObject

Třída pro formátování výsledků zajišťující formátování výstupu. Výpis výsledku operátoru Dunn Index je textový, postup je tedy obdobný, jak již bylo popsáno v třídě RandIndexIOObject. Výsledné naformátování je vidět na obr. 15.

6.6 Silhouette operátor

Silhouette operátor představuje operátor, který počítá a graficky zobrazuje hodnoty siluety. Operátor spočítá hodnotu siluety pro každý objekt v souboru dat, spočítá průměrnou siluetu ve shlucích. Navíc výsledné hodnoty siluety zobrazí do přehledného grafu, ze kterého lze přehledně určit, jak shlukování dopadlo. Tato podkapitola se bude věnovat implementaci a popisu operátorů přímo v RapidMineru. Bližší informace k určení hodnoty siluety jsou popsány v kapitole 4.3.



obr. 16 - Silhouette operátor v RapidMineru

Obr. 16 ukazuje, jak vypadá implementovaný Silhouette operátor v RapidMineru. Silhouette operátor má jeden vstup a dva výstupy. Vstup a výstupy budou podrobněji popsány níže.

6.6.1 Vstupní porty Silhouette operátoru

Silhouette operátor má pouze jeden vstupní port pojmenovaný **cluster set**. Jak název napovídá, Silhouette operátor očekává na vstupu objekt představující tabulková data, která jsou zařazena do shluků. Operátor tedy přijímá objekt typu *ExampleSet* obsahující atribut se zařazením jednotlivých objektů s daty do shluků. Sloupec musí být pojmenovaný jako *cluster*. Je zde, jako u předchozích dvou operátorů, implementována kontrola vstupních hodnot. Podrobněji je kontrola popsána již v podkapitole popisující vstupní porty pro Rand Index operátor.

6.6.2 Výstupní porty Silhouette operátoru

Silhouette operátor má dva výstupní porty pojmenované:

- silhouette set
- original cluster set

Silhouette set výstupní port obsahuje hned několik vypočítaných hodnot. Port vrací objekt SilhouetteIOObject. Jeho součástí jsou vypočtené průměrné hodnoty pro clustery a navíc obsahuje i celý soubor dat s přidaným speciálním atributem pojmenovaným *silhouette*. Tento port je vhodné připojit přímo na port pro výsledky. Prozatím není v RapidMineru implementovaný žádný operátor, který by s vráceným objektem uměl dále pracovat.

Original cluster set vrací hodnoty vstupních dat s přidaným speciálním atributem pojmenovaným *silhouette*. Jedná se tedy o rozšířená vstupní data o jeden sloupec. Výstupní objekt portu **original cluster set** je standardní objekt RapidMineru *ExampleSet*. S tímto objektem umí pracovat velká část operátorů v RapidMineru. Je tedy možné výsledná data použít pro další výpočty.

6.6.3 Parametry Silhouette operátoru

Pokud přidáme operátor Silhouette do okna „Process“ a označíme jej, můžeme v okně „Parameters“ nastavit dva parametry určující metriku výpočtu vzdálenosti objektů. Viz obr 17.

- measure type
- <type> measure

obr. 17 - Parametry Silhouette operátoru

Measure type představuje typy měr, které chceme pro výpočet podobnosti objektů použít. Typ míry by měl odpovídat typu dat, která očekáváme na vstupu. Na výběr jsou celkem čtyři míry. Hodnoty i název druhého parametru se odvíjí od zvolené hodnoty parametru **Measure type.**

Hodnoty parametrů, které lze zvolit, jsou obdobné jako u Dunn Index operátoru. Výpis všech možností, které lze v parametrech nastavit, je vypsaný v kapitole 6.5.3.

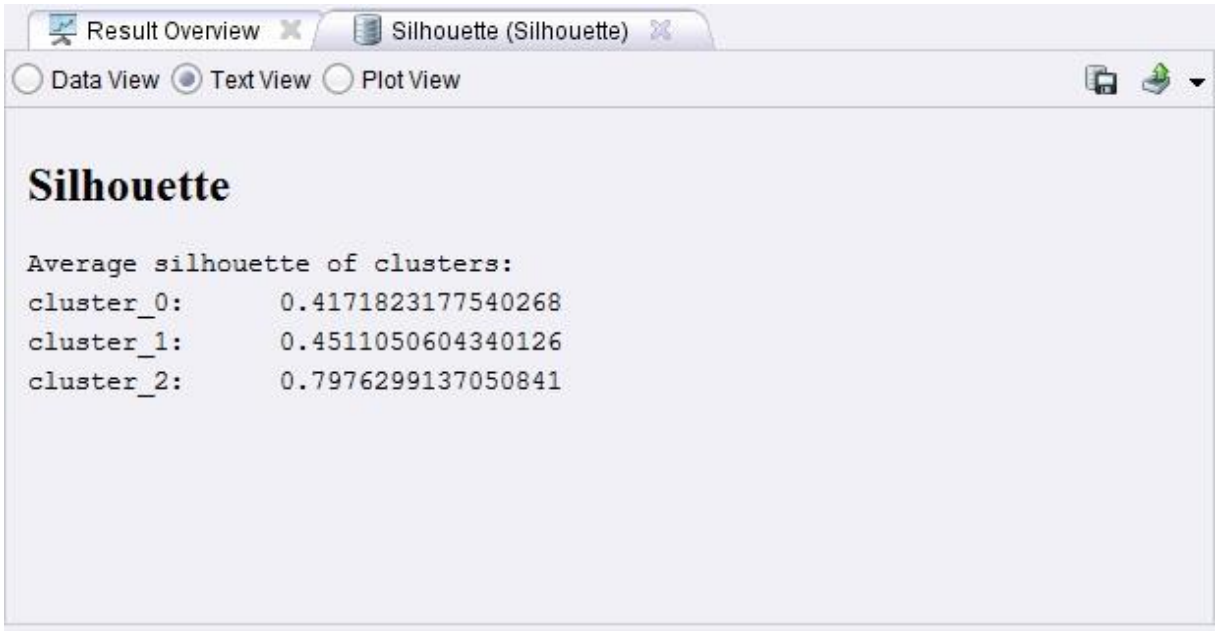
6.6.4 Zobrazení výstupních hodnot

Silhouette operátor má dva výstupní porty, které je možné připojit na porty RapidMineru pro zobrazení výsledků. **Original cluster set** vrací objekt *ExampleSet*, pro který má RapidMiner již implementované zobrazení. Pro výstupní port **silhouette set**, který vrací objekt *SilhouetteIOObject*, jsem implementoval vlastní zobrazení výsledků. Je možné vybrat ze tří zobrazení výsledků, a to textové (Text View), tabulkové (Data View) a grafické (Plot View). Mezi pohledy lze přepínat pomocí radio přepínače viz obr. 18.

obr. 18 - Přepínač pro změnu zobrazení výsledků u Silhouette operátoru

Ukázku všech tří zobrazení výsledků předvedu na souboru dat Iris, jako shlukovou metodu vyberu k-means, který bude mít nastaven počet shluků $k = 3$ a metriku podobnosti používat Euklidovskou vzdálenost. Operátor Silhouette bude rovněž používat Euklidovskou vzdálenost.

Pro textové zobrazení (Text View) jsem implementoval, stejně jako u předchozích operátorů, metodu, která v RapidMineru řeší formátování a výpis formátovaného textu. Textové zobrazení tiskne hodnoty průměrných siluet k jednotlivým shlukům. Hodnoty průměrné siluety se tisknou s přesností na 16 desetinných míst, viz obr. 19.



obr. 19 - Textové zobrazení hodnot průměrné siluety operátorem Silhouette

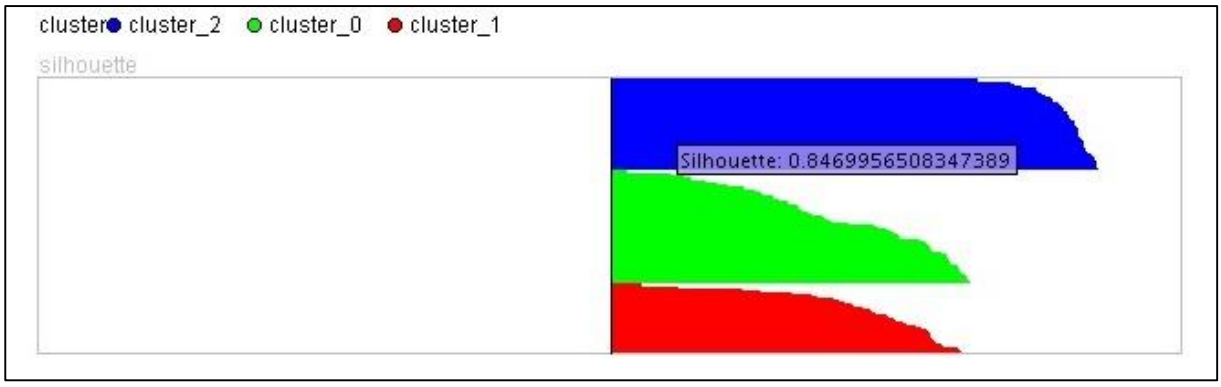
Zobrazení dat do tabulky (Data View) slouží k přehlednějšímu zobrazení výsledných hodnot siluety pro různé shluky. Tabulka má dva sloupce pojmenované Cluster a Silhouette. Sloupec Cluster obsahuje názvy jednotlivých shluků, sloupec Silhouette pak obsahuje hodnoty siluety k daným shlukům. Výsledné hodnoty siluety jsou zobrazeny se zaokrouhlením na 3 desetinná místa.

Tabulkové vykreslení dat má proti textovému vykreslení dat jednu hlavní přednost, a to je možnost řazení dat ve sloupcích. Uživatel může seřadit data podle velikosti hodnoty siluety pro jednotlivé shluky od nejvyšší po nejnižší a obráceně. Pro řazení sloupce stačí kliknout na název sloupce v hlavičce tabulky. U názvu sloupce v hlavičce tabulky se následně zobrazí malý trojúhelník reprezentující směr řazení. Řazení se mění při každém dalším kliknutí na název sloupce v hlavičce tabulky. Jde tedy snadno zjistit, které shluky jsou dobře vytvořené a které shluky naopak špatně vytvořené.

Pro vykreslení dat do tabulky se používá třída *SilhouetteDataRender*, která vytváří tabulkový model dat pro RapidMiner. Třída zároveň nastavuje i možnost řazení hodnot v tabulce. Obr. 20 ukazuje, jak vypadá tabulkové zobrazení dat v RapidMineru.

obr. 20 - Tabulkové zobrazení hodnot průměrné siluety operátorem Silhouette

Grafické zobrazení (Plot View) výstupu Silhouette operátoru vykresluje graf reprezentující hodnoty siluet pro jednotlivé objekty souboru dat. Grafické zobrazení si můžeme prohlédnout na obr. 21. Z obrázku je vidět, že graf je rozdělen ve středu černou čárou. Středová čára rozděluje graf na dvě souměrné části a reprezentuje hodnotu 0. Do levé části grafu se vykreslují hodnoty záporných siluet, tedy hodnoty v intervalu $\langle -1, 0 \rangle$, a v pravé části grafu se vykreslují hodnoty kladných siluet, tedy hodnoty v intervalu $\langle 0,1 \rangle$. V grafu se vykresluje hodnota siluety pro každý objekt v zadaném souboru dat. Pro každý objekt zadaného souboru dat je v grafu vyhrazen jeden řádek o velikosti 1px. Výška grafu se tedy odvíjí podle počtu objektů v zadaném souboru dat. Díky tomu jsou vykresleny vždy všechny hodnoty siluet jednotlivých objektů a nedochází tak ke ztrátě informace. Výška vykresleného řádku v grafu se odvíjí podle velikosti vypočítané siluety konkrétního prvku.



obr. 21 - Grafické zobrazení hodnot siluety pro jednotlivé prvky souboru dat operátorem Silhouette

Základní zobrazení grafu vykreslí hodnoty siluet seřazené podle jednotlivých shluků. Uvnitř shluků je další řazení podle velikosti hodnot siluety. Každému shluku je nastavena jedna barva, jde tedy přehledně zjistit, jak na tom jednotlivé shluky jsou. Která barva patří, ke kterému shluku se určí podle legendy v horní části nad grafem.

Z grafu lze zjistit i konkrétní hodnotu siluety. V případě, že uživatel klikne do grafu myší, vypíše se mu vybraná hodnota siluety jako tooltip. Tooltip je rovněž zobrazen na obr. 21.

Tím však možnosti grafu nekončí. Na levé straně vedle grafu jsou k dispozici čtyři drop-down menu, která slouží ke změně řazení a vykreslení grafu. Drop-down menu jsou pojmenovaná:

- Sort first dimension
- Sort secont dimension
- Sort third dimension
- Color

První tři drop-down menu slouží ke změně řazení hodnot siluet. **Sort first dimension** seřadí všechny hodnoty siluety podle vybraného atributu. **Sort second dimension** seřadí hodnoty podle vybraného atributu a navíc respektuje seřazení podle atributu zvolených v **sort first dimension**. **Sort third dimension** seřadí hodnoty siluety podle vybraného atributu a navíc respektuje seřazení podle atributů zvolených v **sort first dimension** a v **sort second dimension**. V drop-down menu **color** je možné vybrat atribut, podle kterého bude graf obarven. Podle vybrané hodnoty v **color** se překreslí i legenda ke grafu.

Pro ukázku mohu například změnit obarvení grafu z původní hodnoty, kdy jsou jednotlivé hodnoty siluety obarveny podle shluků, na obarvení podle atributu label. Všechny hodnoty siluet seřadím pouze podle hodnot siluet, viz obr. 22.

obr. 22 - Grafické zobrazení hodnot siluety s jiným než základním řazením operátorem Silhouette

Díky možnosti řazení grafu přes více dimenzí může uživatel RapidMineru zjišťovat a vykreslovat další informace o shlukování, které nemusejí být z dat přímo zřejmé. Vícerozměrné řazení je vhodné používat právě na nominální data s omezeným počtem hodnot. Pro reálná nebo číselná data je použitelná většinou jen první dimenze řazení.

6.6.5 Implementace Silhouette operátoru

Nejdůležitější třídy pro Silhouette operátor implementované v pluginu jsou:

- com.rapidminer.operator.*Silhouette.java*
- com.rapidminer.operator.*SilhouetteData.java*
- com.rapidminer.operator.gui.*SilhouetteIOObject.java*
- com.rapidminer.operator.gui.*SilhouetteDataRender.java*
- com.rapidminer.operator.gui.*SilhouetteDataPlotRender.java*
- com.rapidminer.operator.gui.*SilhouettePlotter.java*

Třída Silhouette

Třída reprezentuje operátor pro RapidMiner. Třída řeší načítání dat ze vstupního portu a posílá vypočtená data na výstupní porty. Implementace třídy je obdobná jako pro třídu *DunnIndex*.

Třída SilhouetteData

Jde o hlavní třídu starající se o výpočet hodnot siluety. Pro výpočet hodnot siluety se používá metoda *calculateSilhouette(Map<String, List<Example>> separatedElements, DistanceMeasure*

measure). Výpočet se provádí tak, že se nejprve spočítá průměrná vzdálenost objektu s ostatními objekty ve shluku a to pro každý objekt v souboru dat. Následně se spočítají průměrné vzdálenosti nejbližších shluků pro každý objekt. Nakonec se s těchto dvou vytvořených množin průměrných vzdáleností dopočítají výsledné hodnoty siluet pro jednotlivé objekty. Podrobněji je výpočet siluety popsaný v kapitole 4.3.

SilhouetteData rovněž představuje datový objekt pro třídu *SilhouetteIOObject*, který se stará o vykreslení výsledné hodnoty.

Třída SilhouetteIOObject

Jedná se o třídu starající se o vykreslování a formátování výsledných hodnot. Konstruktor této třídy očekává jako parametr *SilhouetteData*, který obsahuje vypočtené hodnoty siluet. Pro textový výpis výsledku je postup formátování obdobný jako pro *RandIndexIOObject*.

Třída SilhouetteDataRender

Jde o třídu starající se o vykreslení výsledných hodnot siluety do tabulky. Ve třídě se též povoluje řazení sloupců, automatická změna velikosti sloupce nebo roztažení sloupců.

Třída SilhouetteDataPlotRender

SilhouetteDataPlotRender představuje třídu, která se stará o vykreslení grafu siluety. Přijímá objekt *SilhouetteIOObject*, ze kterého si načte data siluet pro jednotlivé instance datového souboru a předpřipraví je pro tvorbu grafu. Graf siluety je následně vytvořen třídou *SilhouettePlotter*.

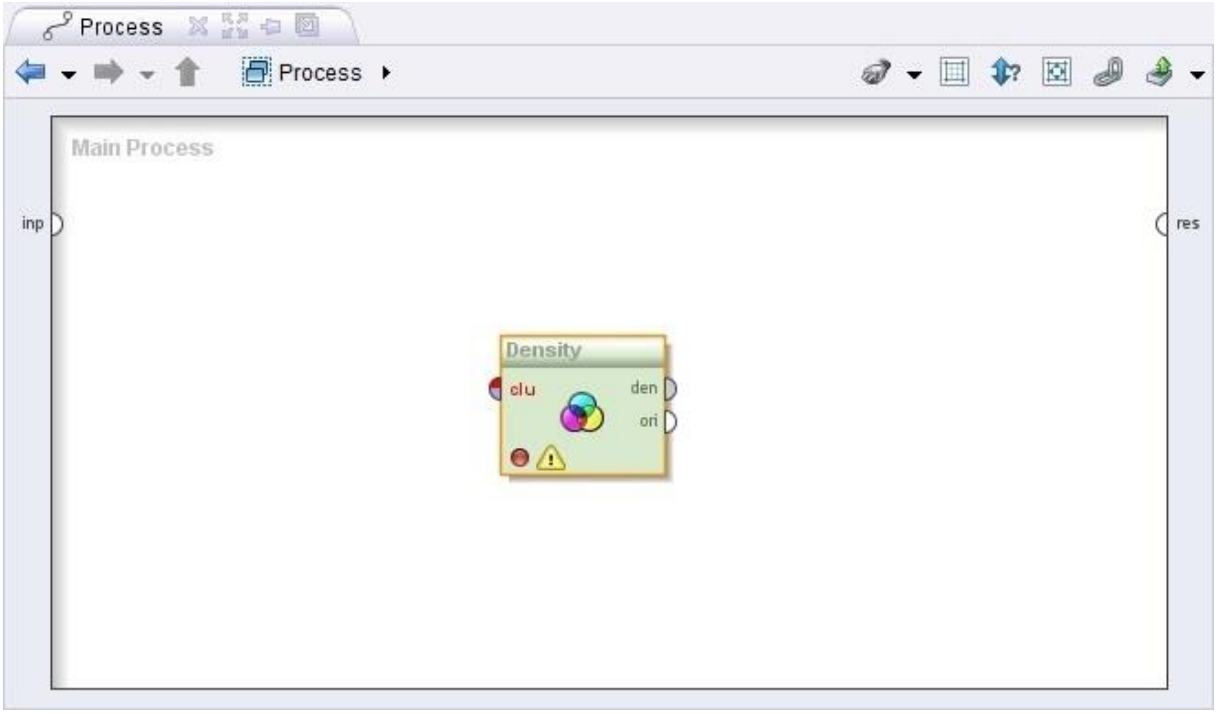
Třída SilhouettePlotter

Třída pro tvorbu grafu siluety. Pro vykreslování grafu siluety se používá java.awt.*Graphics2D*. Mimo vykreslení samotného grafu se ve třídě řeší i šířka grafu podle aktuální velikosti okna, řazení a obarvování grafu podle vybraných atributů i výpis aktuální hodnoty siluety do tooltipu.

6.7 Density operátor

Jde o operátor, který určuje typické hodnoty pro zadaný soubor dat, typické hodnoty pro každý shluk a vykresluje sadu grafů, zobrazující hustotu jednotlivých atributů ve shlucích. Jedná je o operátor,

který ze shlukovaných dat zjišťuje další informace. Tato kapitola se věnuje popisu Density operátoru implementovaného pro RapidMiner. Typickým hodnotám je věnována kapitola 4.4.



obr. 23 - Density operátor v RapidMineru

Obr. 23 ukazuje, jak vypadá implementovaný Density operátor v RapidMineru. Density operátor má jeden vstup a dva výstupy. Vstup a výstupy budou podrobněji popsány níže.

6.7.1 Vstupní porty Density operátoru

Density operátor má pouze jeden vstupní port pojmenovaný **cluster set**. Vstup, stejně jako u předchozích metod, přijímá pouze shlukovaná data. Pravidla pro vstupní port odpovídají pravidlům popsaných v podkapitole 6.6.1 popisující vstupní porty pro Silhouette operátor.

6.7.2 Výstupní porty Density operátoru

Density operátor má dva výstupní porty pojmenované:

- density output
- original cluster set

Density output výstupní port vrací objekt *DensityIOObject*. Jedná se o objekt, který s sebou nese velké množství informací. Obsahuje kompletní vstupní datový soubor, určené typické hodnoty pro celý datový soubor, určené typické hodnoty pro jednotlivé clustery. Tento port je vhodné připojit přímo na port pro výsledky. Prozatím není v RapidMineru implementovaný žádný operátor, který by s vráceným objektem uměl dále pracovat.

Original cluster set vrací nezměněné hodnoty vstupních dat. Tyto výstupní porty jsou zde pro případ, že by uživatel chtěl kromě zjištění typických hodnot pomocí Density indexu s daty dále pracovat nebo zobrazit výsledné hodnoty shlukování.

6.7.3 Parametry Density operátoru

Density operátor nemá žádné další parametry.

6.7.4 Zobrazení výstupních hodnot

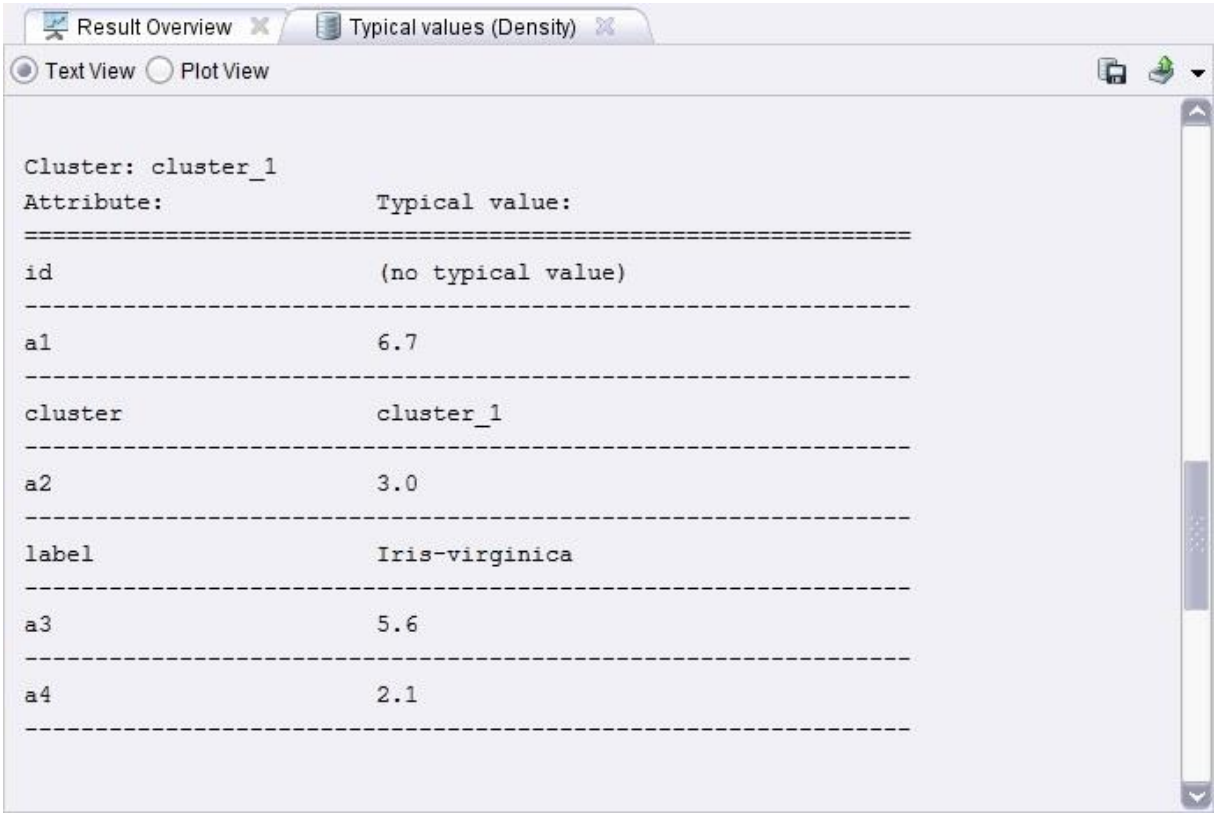
Density operátor má dva výstupní porty, které je možné připojit na porty RapidMineru pro zobrazení výsledků. **Original cluster set** vrací objekt *ExampleSet*, pro který má RapidMiner již implementované zobrazení. Pro výstupní port **density output**, který vrací objekt *DensityIOObject*, jsem implementoval vlastní zobrazení výsledků. Je možné vybrat ze dvou zobrazení výsledků, a to textové (Text View) a grafické (Plot View). Mezi pohledy lze přepínat pomocí radio přepínače viz obr. 24.

obr. 24 - Přepínač pro změnu zobrazení výsledků u Density operátoru

Ukázku obou zobrazení výsledků předvedu na souboru dat Iris, jako shlukovou metodu vyberu k-means, která bude mít nastaven počet shluků $k = 3$ a metriku podobnosti použije euklidovskou vzdálenost.

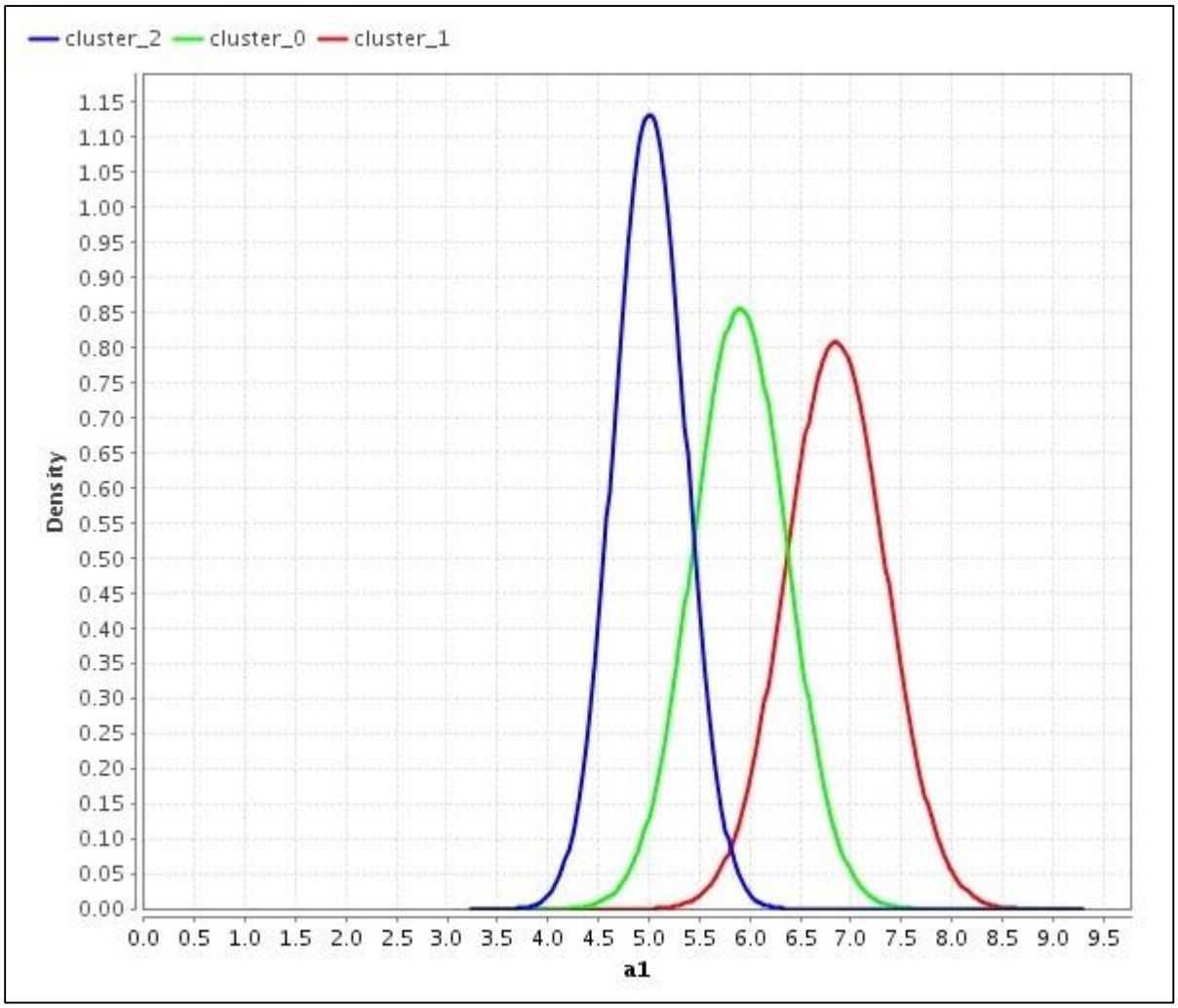
Pro textové zobrazení (Text View) jsem implementoval, stejně jako u předchozích operátorů metodu, která v RapidMineru řeší formátování a výpis formátovaného textu. Tisknou se zde tabulky obsahující typické hodnoty nejprve pro celý soubor dat a následně pro jednotlivé shluky. Tabulky typických hodnot se tisknou pod sebe. Tabulka s daty obsahuje dva sloupce. V prvním sloupci je název atributu a v druhém sloupci je typická hodnota daného atributu. V případě, že má atribut více typických

hodnot, vypisují se všechny oddělené čárkou. Může nastat situace, že sloupec dat obsahuje nominální data, však každá hodnota je použita právě jednou. S tímto samozřejmě Density operátor počítá. Místo vypsání všech hodnot, kterých může být i několik tisíc, se vypíše na řádek, že atribut nemá typickou hodnotu. Výsledné vypsání typických hodnot pro cluster_1 si můžeme prohlédnout na obr. 25. Na obrázku si můžeme všimnout i vytisknuté informace, že atribut id nemá typickou hodnotu.



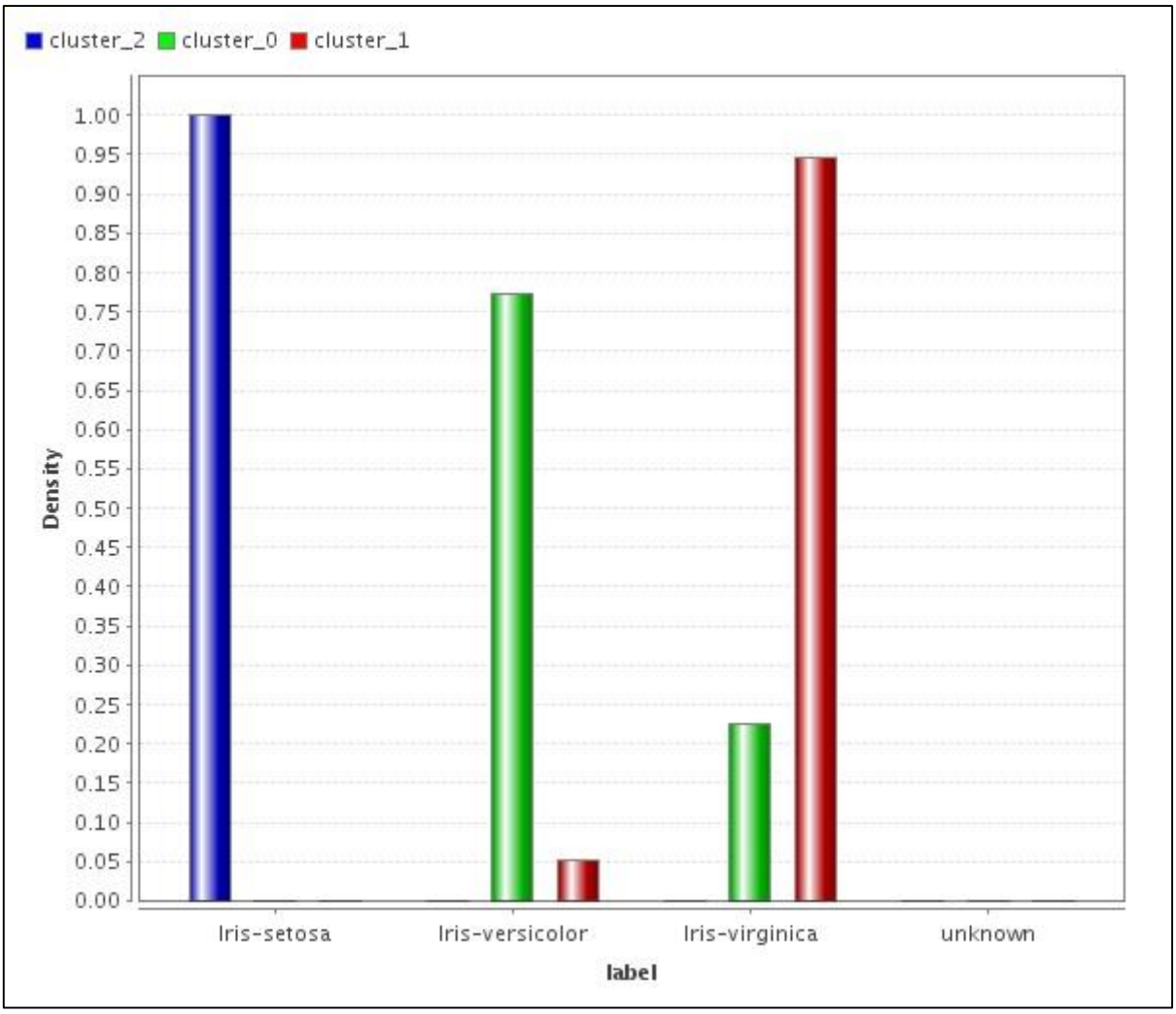
obr. 25 - Textové zobrazení typických hodnot pro cluster_1 operátorem Density

Pokud přepneme zobrazení výsledků Density operátoru z textového na grafické, zobrazí se řada grafů. Počet grafů je určen podle počtu atributů vstupního souboru dat. Pro každý sloupec dat je vytvořen jeden graf, který graficky vykresluje zastoupení jednotlivých hodnot v daném sloupci. Vykreslují se dva různé typy grafů, pro numerická data a pro nominální data. V základním nastavení se pro numerická data vykreslí spojitá funkce pro každý shluk. Jednotlivé spojité funkce představují hustotu zastoupení číselných hodnot v určitém shluku. Z grafu lze přehledně určit, jaké hodnoty jsou pro daný shluk typické a porovnat je s ostatními shluky. Pro ukázku vyberu první atribut ze shlukovaného souboru dat Iris. Na obrázku obr. 26 je vidět, že shluk pojmenovaný cluster_2 má pro atribut a1 největší zastoupení hodnot mezi 4,0 až 6,0, shluk cluster_1 mezi hodnotami 5,5 až 8,2 a shluk cluster_0 mezi hodnotami 4,8 až 7,0.



obr. 26 - Grafické zobrazení hustoty dat pro první atribut a1 ze souboru dat Iris operátorem Density

Pro nominální data se vykresluje sloupcový graf, kde pro každou hodnotu atributu lze zjistit jeho procentuální zastoupení ve shluku. Pro ukázku použiji opět shlukovaný soubor dat Iris a atribut label, který obsahuje právě 3 nominální hodnoty Iris-setosa, Iris-versicolor, Iris-virginica. Na obr. 27 je vidět, že 100% objektů s názvem Iris-setosa je zařazeno do shluku cluster_2. Shluk cluster_1 obsahuje 6% objektů s názvem Iris-versicolor a 94% objektů s názvem Iris-virginica. Poslední shluk cluster_0 obsahuje 77% objektů s názvem Iris-versicolor a 23% objektů s názvem Iris-virginica.



obr. 27 - Grafické zobrazení hustoty dat pro atribut label ze souboru dat Iris operátorem Density

Základní nastavení, které vykresluje hustotu každého atributu podle zařazení do shluků, je možné měnit. Místo určení hustoty podle shluku lze určovat hustotu atributů podle jakéhokoli jiného nominálního atributu. Jde tedy zjišťovat a vykreslovat další vztahy mezi jednotlivými atributy. V případě předchozí ukázky je možné zjistit, jak vypadá hustota dat v jednotlivých sloupcích například podle atributu label.

6.7.5 Implementace Density operátoru

Nejdůležitější třídy pro Density operátor implementované v pluginu jsou:

- com.rapidminer.operator.*Density.java*
- com.rapidminer.operator.*DensityData.java*
- com.rapidminer.operator.gui.*DensityIOObject.java*

- com.rapidminer.operator.gui.*DensityPlotRender.java*
- com.rapidminer.operator.gui.*DensityPlotter.java*
- com.rapidminer.operator.gui.*DensitySinglePlotter.java*

Třída Density

Třída reprezentuje operátor pro RapidMiner. Třída řeší načítání dat ze vstupního portu a posílá vypočtená data na výstupní porty. Implementace třídy je obdobná jako pro třídu *DunnIndex*.

Třída DensityData

V případě Density operátoru se neprovádí žádný výpočet, ale hledá v datech typické hodnoty. Třída *DensityData* nejprve vyhledá typické hodnoty pro celý dodaný soubor dat. Následně rozdělí soubor dat podle shluků a postupně hledá typické hodnoty pro různé shluky.

DensityData rovněž představuje datový objekt pro třídu *DensityIOObject*, který se stará o vykreslení výsledné hodnoty.

Třída DensityIOObject

Jedná se o třídu starající se o vykreslování a formátování výsledných hodnot. Konstruktor této třídy očekává jako parametr *DensityData*, který obsahuje nalezené typické hodnoty. Pro textový výpis výsledku je postup formátování obdobný jako pro *RandIndexIOObject*.

Třída DensityPlotRender

DensityPlotRender představuje třídu, která se stará o vykreslení grafů hustoty pro jednotlivé atributy. Třída pracuje s objektem *DensityIOObject*, ze kterého si načítá potřebná data pro vykreslování příslušných grafů. Pro tvorbu samotných grafů se používá třída *DensityPlotter*.

Třída DensityPlotter

Třída *DensityPlotter* reprezentuje kontejner pro jednotlivé grafy. Jedná se o obalovou třídu pro *DensitySinglePlotter*. Třída řeší změny parametrů pro vykreslení jednotlivých grafů, vykreslení legendy apod.

Třída DensitySinglePlotter

DensitySinglePlotter je třídou představující graf hustoty pro jeden konkrétní atribut. Ve třídě se též řeší, o jaká data se jedná a jaký typ grafu má být vykreslen. V případě nominálních dat je vykreslován sloupcový graf a pro numerická data je vykreslován graf spojité funkce.

7 Testování na skutečných datech

V této kapitole předvedu správnou funkčnost všech čtyř mnou implementovaných operátorů pro RapidMiner. Pro ukázku využiji několik souborů dat, na kterých bude předvedena správná funkce. Součástí kapitoly boudou i výkonnostní testy.

7.1 Testovací soubory dat

7.1.1 Iris

Základním testovacím souborem dat, na kterém chci předvést správnou funkci všech implementovaných operátorů, bude soubor dat **Iris**. Jedná se o jeden z nejznámějších datových souborů pro klasifikaci a shlukování. Tento soubor dat je ve spoustě shlukovacích a klasifikačních programů přímo vložen a určen k testování. RapidMiner není výjimkou. **Iris** je možné v RapidMineru nalézt v okně Repositories pod cestou Samples > data > Iris.

Iris obsahuje 3 třídy, z nichž každá obsahuje 50 instancí. Jednotlivé třídy představují určitý typ kosatce. Jedná se tedy o soubor dat se 150 instancemi. Soubor dat **Iris** má celkem 5 atributů:

- a1 - délka kališního lístku
- a2 - šířka kališního lístku
- a3 - délka okvětního lístku
- a4 - šířka okvětního lístku
- label - typ kosatce (Iris Setosa, Iris Versicolour, Iris Virginica)

Data i popis lze nalézt na [41].

7.1.2 Allstate Purchase Prediction Challenge

Pro testování běhu implementovaných operátorů s většími vstupními daty jsem si zvolil soubor dat **Allstate Purchase Prediction Challenge**. Soubor dat obsahuje historie nákupů zákazníků v obchodech. Soubor dat jsem vybral na serveru kaggle.com[7]. Ke stažení je zde na výběr trénovací soubor dat, který obsahuje 665 250 instancí a dále testovací soubor dat, který obsahuje 198 857 instancí. Atributů v těchto datech je celkem 25 a většina z nich reprezentovaná nominálními hodnotami.

[7] Přímý odkaz pro stažení souboru dat Allstate Purchase Prediction Challenge ze serveru kaggle.com: https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data

Popis jednotlivých atributů:

Název	Typ	Popis
customer_ID	nominální	unikátní identifikátor zákazníka
shopping_pt	nominální	unikátní identifikátor nákupního místa
record_type	nominální	typ nákupního místa
day	nominální	den v týdnu (0 - 6, 0 = Pondělí)
time	numerické	čas (HH:MM)
state	nominální	stát, kde probíhalo nakupování
location	nominální	místo, kde došlo k nákupu
homeowner	dichotomická	na kolik lidí se vztahuje daná operace (1, 2, 3 nebo 4)
car_age	numerická	zda zákazník vlastní své bydlení nebo ne (0 = ne, 1 = ano)
car_value	nominální	stáří zákazníkova auta
rist_factor	nominální	hodnota zákazníkova auta, když bylo nové
age_oldest	numerická	věk nejstaršího člena nakupující skupiny
age_youngest	numerická	věk nejmladšího člena nakupující skupiny
married_couple	dichotomická	zda obsahuje nakupující skupina manželský pár (0 = ne, 1 = ano)
C_previous	nominální	jaký výrobek má nebo v minulosti měl jako C (0 = nic, 1, 2, 3, 4)
duration_previous	numerická	jak dlouho (v letech) byl zákazník kryt svým předchozím emitentem
A, B, C, D, E, F, G	nominální	vlastnosti produktu
cost	numerická	cena vybraných vlastností produktu

Data i popis lze nalézt na [42].

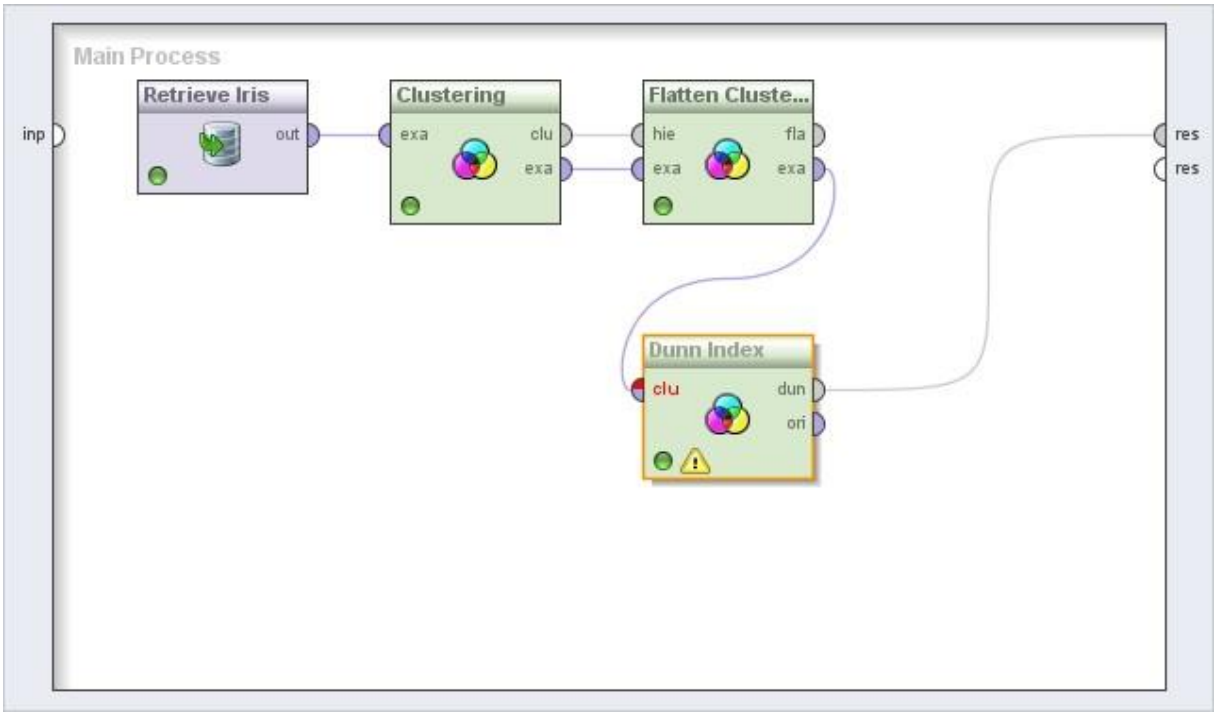
7.2 Ověření správnosti výpočtu operátorů

7.2.1 Ověření Dunn Index operátoru

Implementované jsou celkem 4 operátory. S ukázkou funkčnosti operátorů začnu nejprve s operátorem Dunn Index. Dunn Index operátor spočítá hodnotu reprezentující, jak dobře jsou shluky vytvořené. Čím vyšší hodnota Dunn indexu, tím shlukování dopadlo lépe. Může se tedy jednat o dobrou metodu pro stanovení optimálního počtu shluků. V knize [13] je vyobrazen graf, který reprezentuje

závislost hodnot Dunn indexu na počtu shluků. Hodnoty pro graf jsou počítány systémem SYSTAT[8]. Pro ukázku funkčnosti mého operátoru porovnám hodnoty Dunn indexu vypočítané mým operátorem pro RapidMiner s hodnotami spočítanými softwarem SYSTAT, které jsou uvedeny v knize [13].

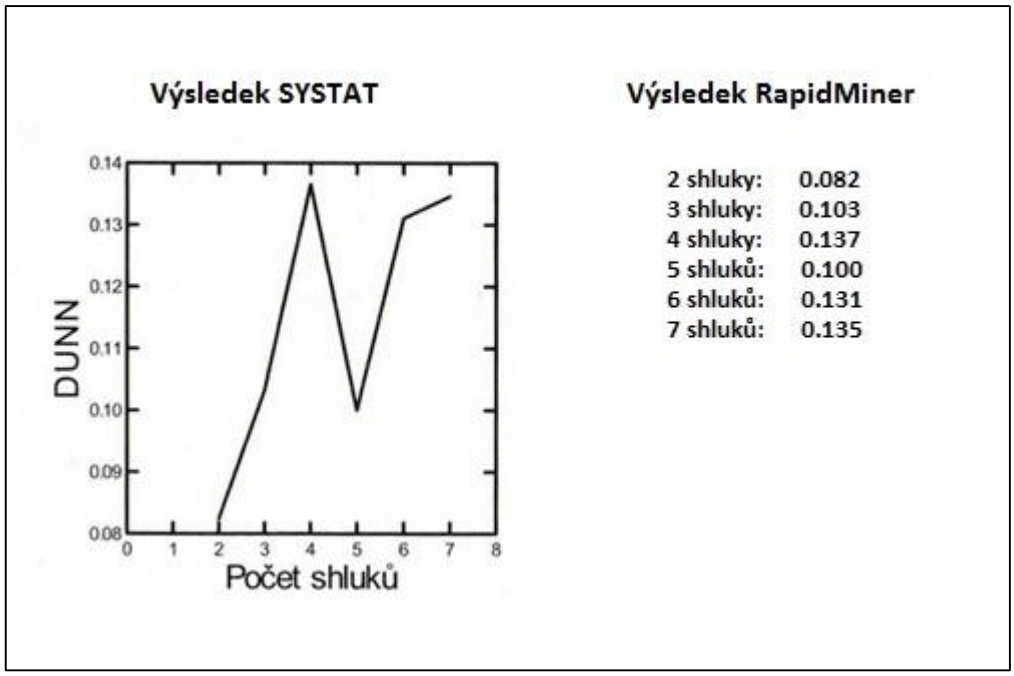
Soubor dat, který je shlukovaný, je datový soubor Iris popsaný v kapitole 6.1.1. SYSTAT určuje hodnoty Dunn indexu při použití hierarchického shlukování, využívajícího euklidovskou vzdálenost a metodu úplného spojení. Graf zobrazuje hodnoty Dunn indexu spočítané systémem SYSTAT pro počty shluků 2, 3, 4, 5, 6 a 7. Pro stejné nastavení shlukování spočítám hodnoty Dunn Indexu mým operátorem a porovnám hodnoty s hodnotami ze systému SYSTAT.



obr. 28 - Ukázka zapojení Dunn Index operátoru v RapidMineru

Na obr. 28 jsou vidět čtyři operátory. První je **Retrieve Iris**, který se stará o načtení datového souboru. Následuje operátor **Clustering**, který reprezentuje hierarchické shlukování využívající euklidovskou vzdálenost a metodu úplného spojení. Operátor **Clustering** vrací dendrogram, který neobsahuje data rozdělená do shluků. V RapidMineru je tedy nutné použít další operátor zvaný **Flatten Clustering**, který rozdělí dendrogram do shluků. Když jsou data rozdělená do shluků, můžeme spočítat hodnotu Dunn indexu.

[8] SYSTAT je statistický a analytický software. Pro více informací http://www.systat.com/ .



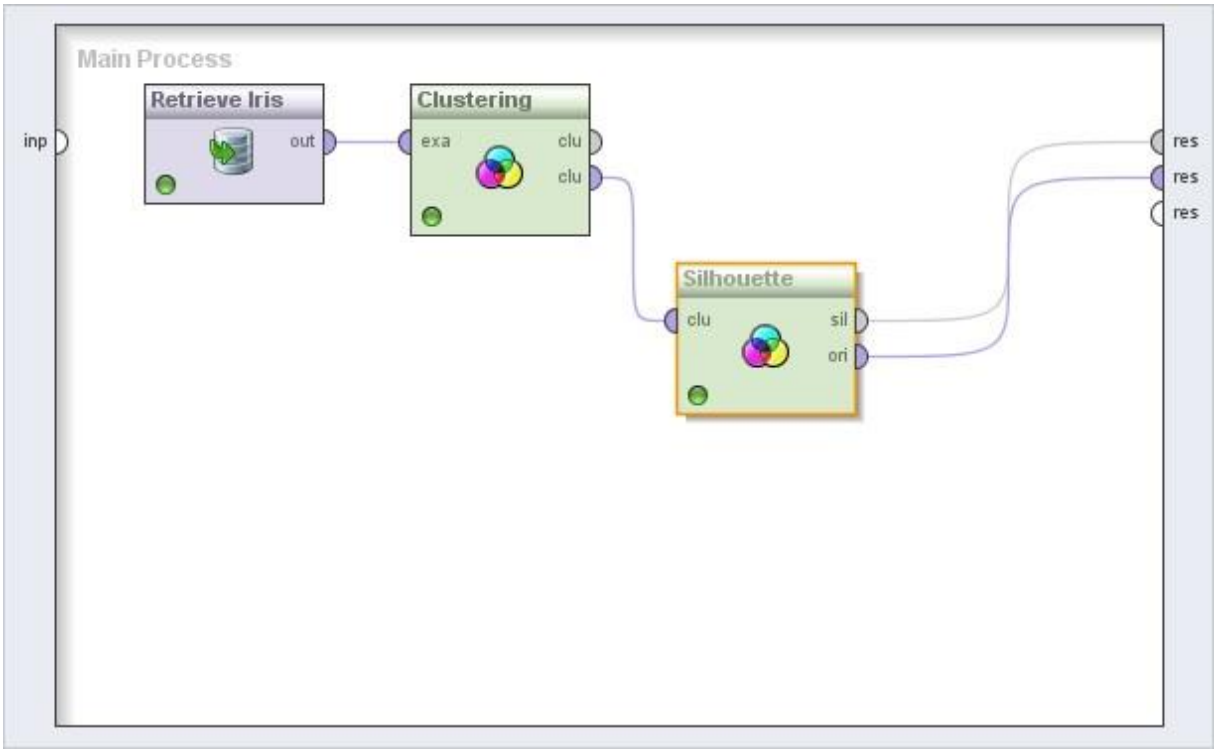
obr. 29 - Porovnání vypočtených Dunn indexů v systému SYSTAT a v RapidMineru

Při porovnání výsledků ze systému SYSTAT a výsledků vypočtených mým operátorem Dunn Index lze vidět, že se naprosto shodují. Tímto bych chtěl demonstrovat správnost výpočtu mého operátoru. Graf výsledků SYSTAT vyobrazen na obr. 29 byl převzat z knihy [13].

7.2.2 Ověření Silhouette operátoru

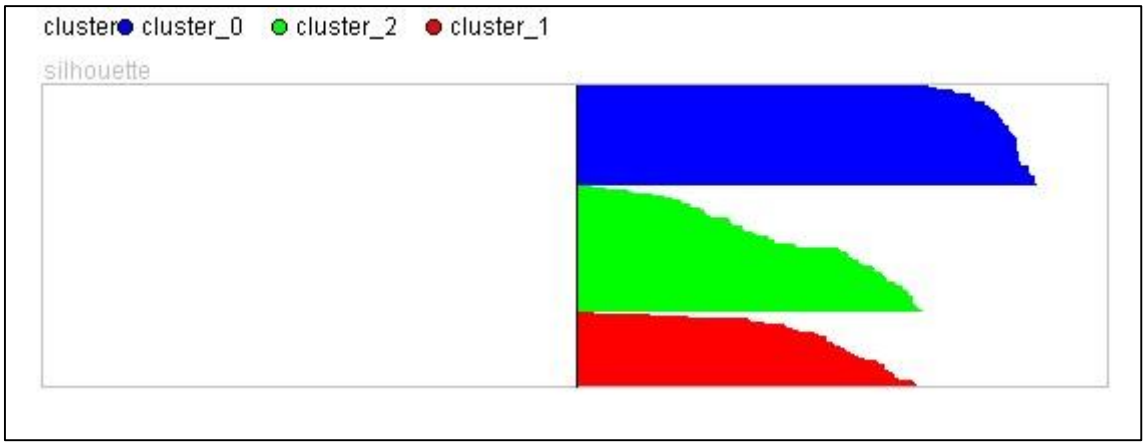
Silhouette operátor slouží k výpočtu průměrné hodnoty siluety ve shlucích a dále pro výpočet siluet všech shlukovaných objektů. Pro porovnání správnosti výpočtu mého operátoru jsem zvolil interaktivní programové prostředí Matlab[9]. V Matlabu lze též provést shlukování dat, a dokonce i vykreslit graf výsledné siluety. Pro porovnání použiji graf siluety vypočítaný operátorem Silhouette v RapidMineru a graf siluety vytvořeny Matlabem. Pro shlukování vyberu datový soubor Iris, který je popsán v kapitole 7.1.1. Jako shlukovací metodu zvolím k-means s nastaveným počtem shluků $k = 3$. Metriku pro výpočet vzdálenosti zvolím Manhattanskou.

[9] Matlab je interaktivní programové prostředí vyvíjené společností MathWorks. Pro více informací o tomto softwaru je možné nalézt na http://www.mathworks.com/products/matlab/ .

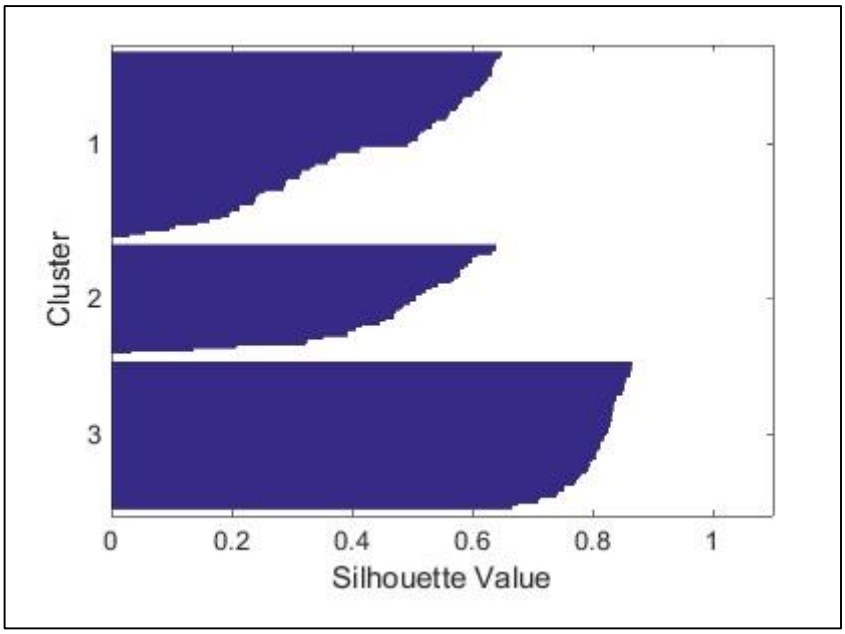


obr. 30 - Ukázka zapojení Silhouette operátoru v RapidMineru

Z obr. 30 je vidět zapojení operátoru pro shlukování popsaného výše a operátoru pro výpočet siluety. K-means shlukovací operátor vrací přímo shlukovaná data, není tedy potřeba připojovat operátor **Flatten Clustering**, jak tomu bylo u hierarchického shlukování.



obr. 31 - Graf siluety v RapidMiner



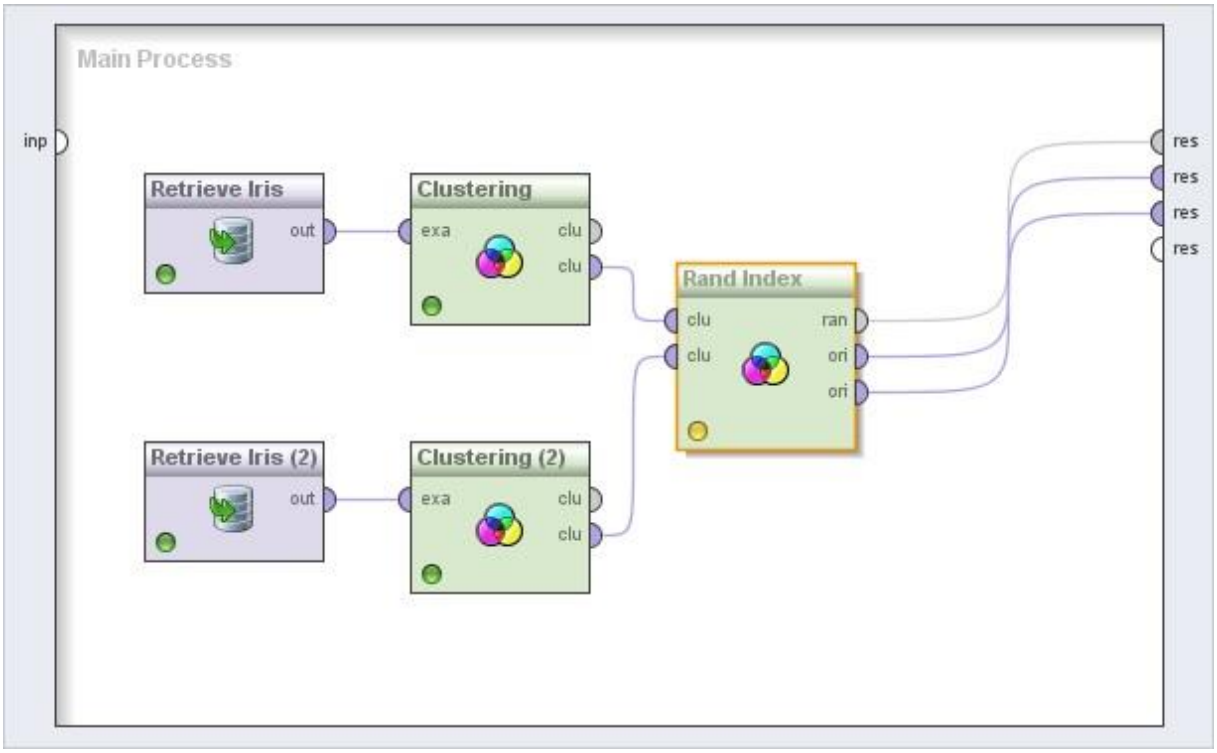
obr. 32 - Graf siluety Matlab

Při porovnání grafů vyobrazených na obr. 31 a obr. 32 lze vidět, že grafy jsou totožné. Hodnoty siluet vypočítané mým operátorem Silhouette pro RapidMiner odpovídají hodnotám vypočtených Matlabem. Tímto bych chtěl demonstrovat správnost výpočtu mého operátoru.

7.2.3 Ověření Rand Index operátoru

Rand Index operátor slouží k porovnání výsledků shlukování dvou různých shlukových metod. Velikost hodnoty Rand indexu ukazuje, jak rozdílné shluky každá metoda vytvořila. Podrobněji je popsán Rand Index v kapitole 4.1.

Bohužel jsem nenalezl software, který by byl vhodný pro porovnání výpočtů mnou implementovaného Rand Index operátoru. Ukázku funkčnosti operátoru předvedu pouze v RapidMineru. Soubor dat ke shlukování zvolím opět Iris. Vybraný datový soubor je popsán v kapitole 7.1.1. Metodu pro shlukování použiji k-means s nastavenou euklidovskou metrikou.

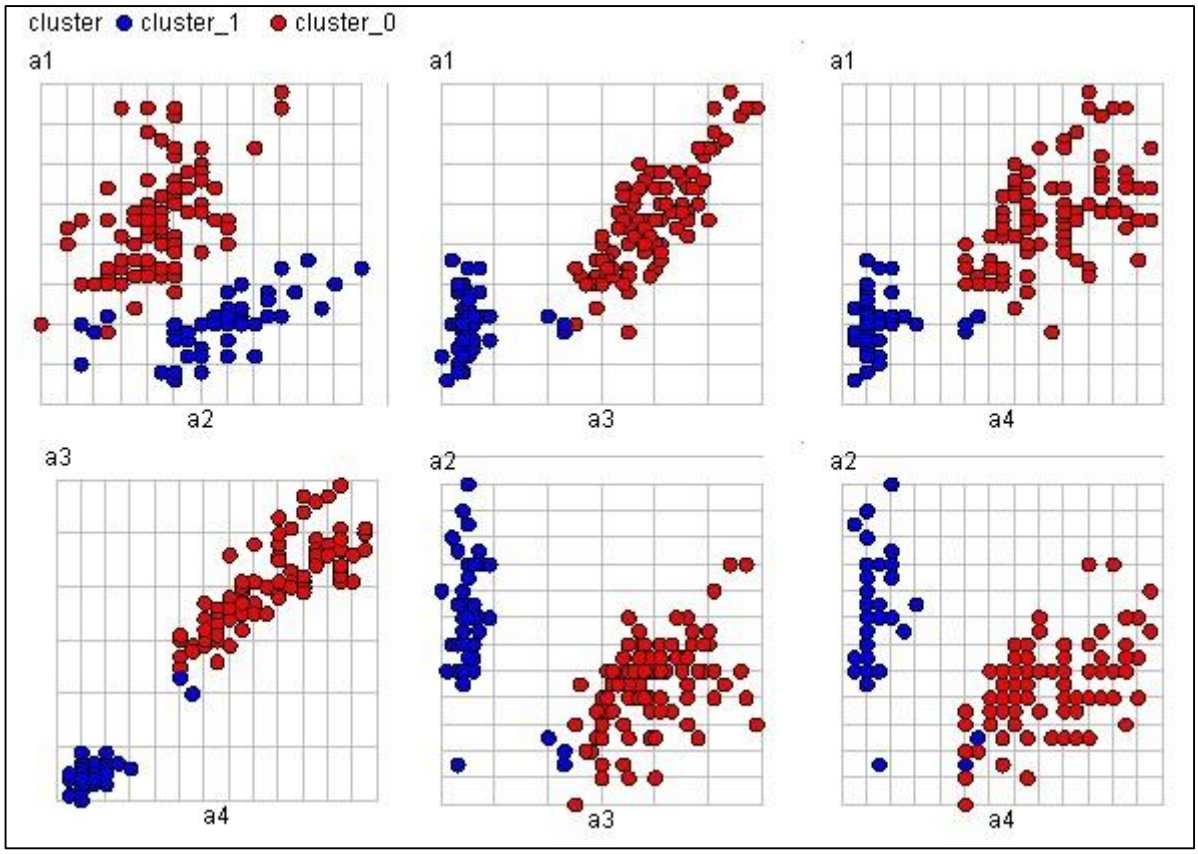


obr. 33 - Ukázka zapojení Rand Index operátoru v RapidMineru

Na obr. 33 vidíme správné zapojení Rand Index operátoru. Jsou potřeba načíst dva vstupní soubory dat, a to oba dva stejné. V případě rozdílných vstupních souborů dat nemá Rand Index operátor jak shlukování porovnat. Dále je nutné použít dva různé operátory pro shlukování. V mé ukázce jsou oba operátory shlukování k-means. Jeden s nastavenou hodnotou $k = 3$ a druhý s nastavenou hodnotou $k = 2$. Pro takovéto nastavení je výsledný Rand Index roven hodnotě 0.7701118568232662. Pokud se zamyslíme nad strukturou datového souboru a porovnáme výsledné shlukování pro $k = 3$ a pro $k = 2$ zjistíme, že rozdíl ve shlucích není tak velký.

Pro $k = 2$ jsou vytvořeny dva shluky.

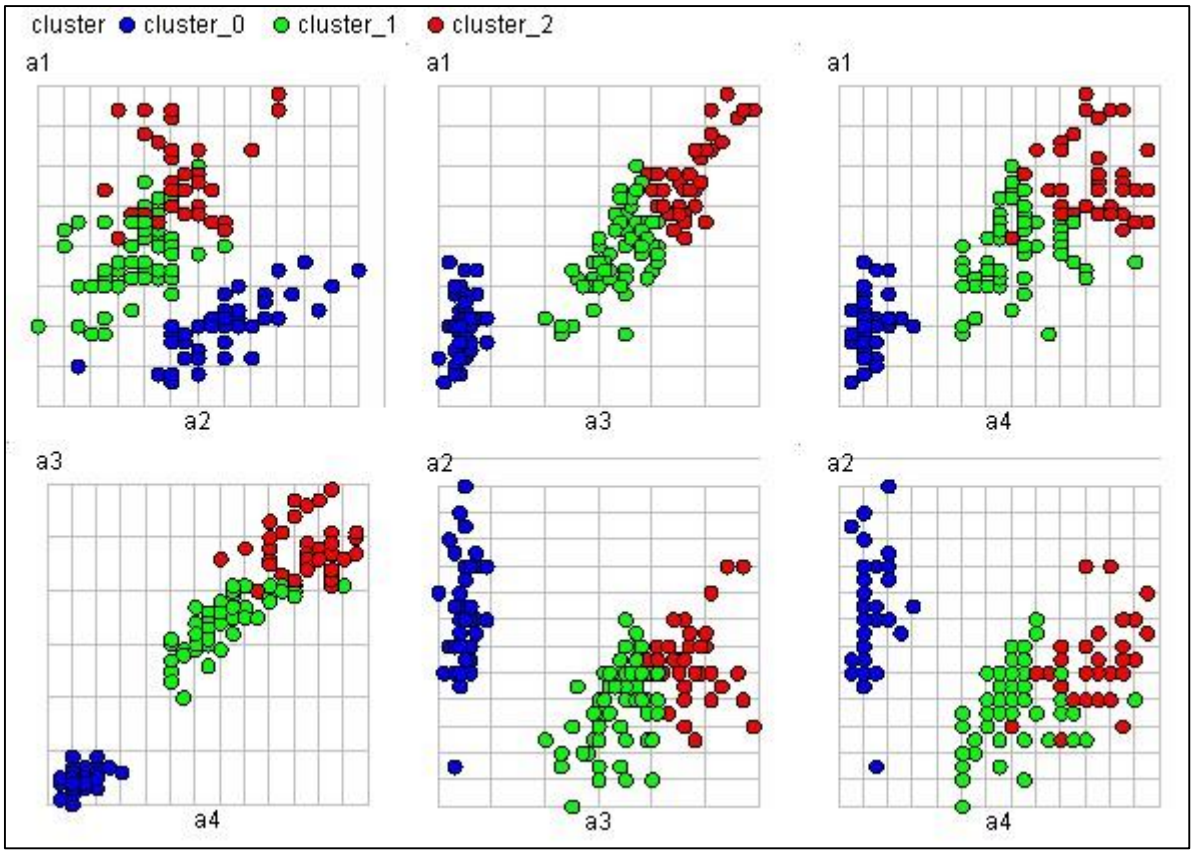
- **K2cluster_0** obsahuje 97 objektů
- **K2cluster_1** obsahuje 53 objektů



obr. 34 - Graf atributů datového souboru Iris pro k-means s hodnotou k = 2

Pro $k = 3$ jsou vytvořeny tři shluky.

- **K3cluster_0** obsahuje 50 objektů
- **K3cluster_1** obsahuje 62 objektů
- **K3cluster_2** obsahuje 38 objektů

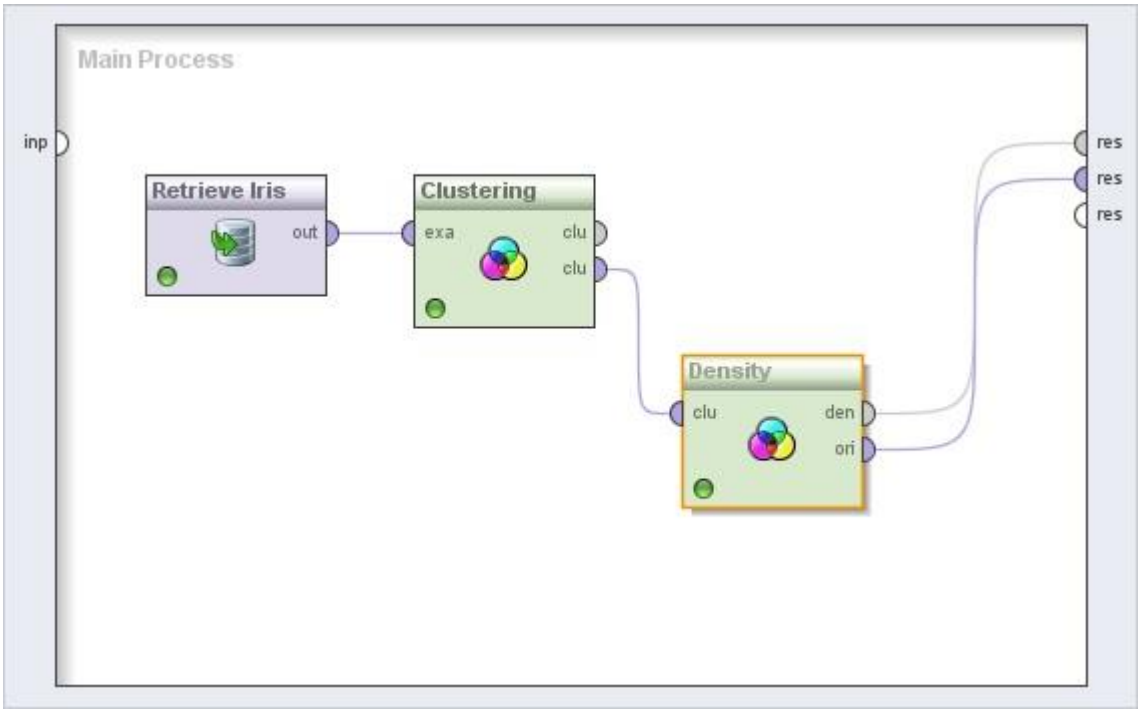


obr. 35 - Graf atributů datového souboru Iris pro k-means s hodnotou k = 3

Pokud tedy porovnáme obě shlukovaná data, zjistíme, že shluk **K2cluster_1** obsahuje většinu dat shodných s **K3cluster_0**. To tedy znamená, že **K2cluster_0** obsahuje většinu dat shodných se shluky **K3cluster_1** a **K3cluster_2**. Pokud tedy porovnám tyto dvě metody shlukování, zjistím, že se v převážné většině dat stalo pouze to, že se jeden velký shluk **K2cluster_0** rozdělil na dva menší shluky **K3cluster_1** a **K3cluster_2**. Změna těchto dvou datových souborů je právě přidaný shluk **K3cluster_2** obsahující 38 objektů. Při výpočtu Rand indexu se tedy zjistí rozdíl mezi shlukovanými soubory dat přibližně 23%. Jedná se samozřejmě jen o hrubý odhad, ale i z něj lze odvodit, že vypočtená hodnota Rand indexu je správná.

7.2.4 Ověření Density operátoru

Density operátor nepočítá žádné vlastní hodnoty, ale hledá ve shlukovaných datech další informace. Správnost výsledků lze pro malý soubor dat, jakým je právě Iris, zjistit pouhým pohledem. Pro operátor nechám tedy datový soubor Iris shlukovat metodou k-means s nastaveným $k = 3$. Jako metriku výpočtu vzdáleností zvolím euklidovskou.



obr. 36 - Použití Density operátoru

Po doběhnutí operátorů v RapidMineru se vypíší typické hodnoty pro celý datový soubor, následované typickými hodnotami pro jednotlivé shluky. Porovnat správně nalezené typické hodnoty pro celý soubor dat Iris je velice jednoduché. Pro číselné atributy a1, a2, a3, a4 stačí seřadit celý datový soubor podle daného atributu a podívat se na hodnotu, která je na řádku 75. Pokud tyto hodnoty porovnám s hodnotami nalezenými Density operátorem, zjistím, že jsou totožné.

- typická hodnota a1 = 5,8
- typická hodnota a2 = 3,0
- typická hodnota a3 = 4,3
- typická hodnota a4 = 1,3

Datový soubor Iris obsahuje 50 instancí Iris Setosa, 50 instancí Iris Versicolour a 50 instancí Iris Virginica. To tedy znamená, že typickou hodnotou atributu label jsou všechny 3 názvy kosatců, protože se v datovém souboru objevují stejně často. Ke stejnému závěru došel i operátor Density.

Pro zjištění typických hodnot v jednotlivých shlucích lze použít stejný postup a ověřit tak správnost operátoru Density. Při prozkoumání shlukovaných dat a výsledků Density operátoru pro jednotlivé shluky jsem zjistil, že typické hodnoty nalezené operátorem jsou správné.

7.2.5 Použití všech operátorů

Jednotlivé operátory jsou na sobě absolutně nezávislé. Lze tedy zapojit všechny operátory pro vyhodnocení shlukování za sebe a RapidMiner s tím nebude mít žádný problém.

7.3 Testování na větším souboru dat

Pro testování operátorů na větším množství dat jsem si zvolil soubor dat Allstate Purchase Prediction Challenge. Podrobněji je tento soubor dat popsán v kapitole 7.1.2. Nebudu však testovat se všemi 25 atributy. Některé atributy jsou nic neříkající v rámci shlukování. Nebudu počítat s atributy A, B, C, D, E, F, G, C_previous, duration_previous, customer_ID, risk_factor, time. Délka datového souboru je rovněž příliš velká. RapidMiner pro takto obrovské soubory dat počítá velice dlouho. Je to způsobeno tím, že RapidMiner nevyužívá plný výpočetní výkon počítače.

7.3.1 Testování Dunn Index operátoru na větších datech

Na výše uvedeném souboru dat jsem testoval rychlost Dunn Index operátoru. Jako shlukovací metodu jsem použil k-means a jako míru podobnosti jsem zvolil MixedEuclideDistance. Operátorem jsem se snažil nalézt nejlepší hodnotu Dunn indexu. Zkoušel jsem různé hodnoty počtu shluků:

- k = 2 – hodnota Dunn indexu: 0.00250
- k = 3 – hodnota Dunn indexu: 0.00521
- k = 4 – hodnota Dunn indexu: 0.00504
- k = 5 – hodnota Dunn indexu: 0.00705
- k = 6 – hodnota Dunn indexu: 0.00378
- k = 7 – hodnota Dunn indexu: 0.00731
- k = 8 – hodnota Dunn indexu: 0.00790
- k = 9 – hodnota Dunn indexu: 0.00730
- k = 10 – hodnota Dunn indexu: 0.01389
- k = 11 – hodnota Dunn indexu: 0.00750
- k = 12 – hodnota Dunn indexu: 0.00750
- k = 13 – hodnota Dunn indexu: 0.01191
- k = 14 – hodnota Dunn indexu: 0.00982

Výpočet hodnot pro soubor dat s 20 000 instancemi a 13 atributy trvá průměrně 9 minut. Průměrná hodnota běhu operátoru byla spočítána jako průměr z dob běhů operátoru pro $k = 2$ až $k =$

14. Informace o délce běhu operátoru byla získána z RapidMineru. Z vypočítaných hodnot jsem zjistil, že nejvhodnější je použít k = 10.

7.3.2 Testování Silhouette operátoru na větších datech

Operátor Dunn Index pomohl vybrat počet shluků. Pro stejné nastavení shlukového algoritmu jako u Dunn Index operátoru zjistím, jak dobře jsou shluky komplexní právě pro hodnotu k = 10. Využiji k tomu operátor Silhouette. Silhouette operátor zjistil průměrné hodnoty siluet v jednotlivých shlucích, viz obr. 37.

Cluster	Silhouette
cluster_0	0.435
cluster_1	0.570
cluster_2	0.478
cluster_3	0.497
cluster_4	0.466
cluster_5	0.490
cluster_6	0.484
cluster_7	0.490
cluster_8	0.567
cluster_9	0.475

obr. 37- Tabulka vypočtených průměrných siluet

Výpočet hodnot pro soubor dat s 20 000 instancemi a 13 atributy trvá průměrně 25 minut. Průměrná hodnota běhu operátoru byla spočítána jako průměr z dob běhů operátoru pro několik po sobě jdoucích spuštění s různým nastavením shlukovacího algoritmu. Informace o délce běhu operátoru byla získána z RapidMineru.

7.3.3 Testování Rand Index operátoru na větších datech

Rand Index operátor porovnává výsledná shlukovaná data dvou shlukových metod. Jelikož je soubor dat velký a jediný operátor, který je v RapidMineru schopen shlukovat takové množství dat v čase kratším než několik hodin, je pouze metoda k-means. V testu porovnám dva výsledky s k-means pro různé hodnoty k. Oba operátory k-means budou mít nastavenou míru podobnosti MixedEuclideDistance. Počet shluků pro první operátor nastavím na $k = 10$. Druhému operátoru

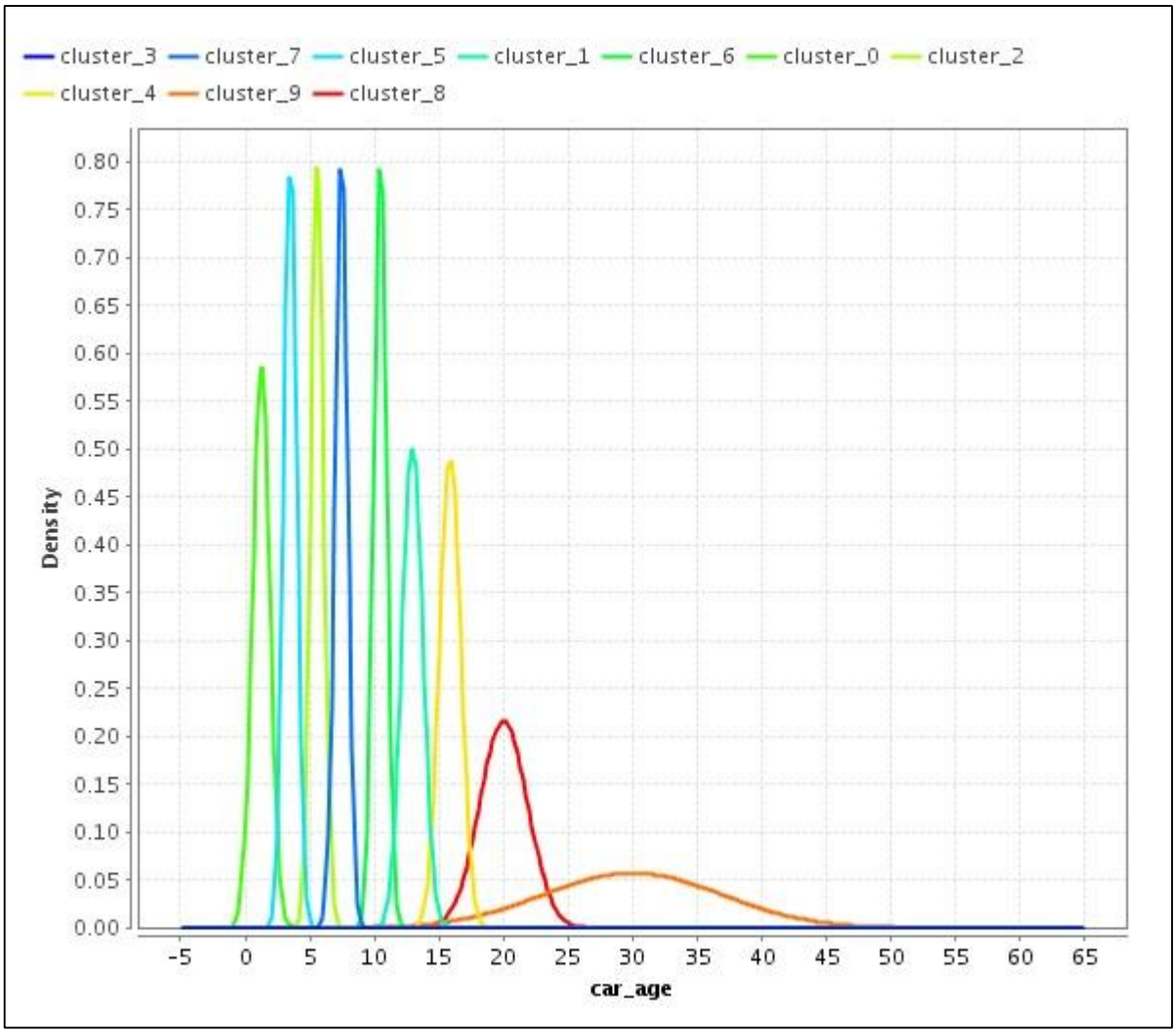
nastavím počet shluků $k = 13$, protože při testech operátoru Dunn Index tento počet shluků vyšel jako druhý nejlepší.

Pro výše popsané nastavení vyšel výsledný Rand index 0.9236. Z hodnoty lze vyčíst, že shlukování pro $k = 10$ a $k = 13$ se liší o 7,64%.

Výpočet hodnot pro soubor dat s 20 000 instancemi a 13 atributy trvá průměrně 5 minut. Průměrná hodnota běhu operátoru byla spočítána jako průměr z dob běhů operátoru pro několik po sobě jdoucích spuštění s různým nastavením shlukovacího algoritmu. Informace o délce běhu operátoru byla získána z RapidMineru.

7.3.4 Testování Density operátoru na větších datech

Operátorem Density následně mohu zjistit, jaké typické hodnoty jsou v jednotlivých shlucích. Shlukovací metoda a nastavení shlukování bude odpovídat nastavení popsané v kapitole 7.3.1. Zde jsme zjistili, že nejvhodnější je použít k = 10, pro které určíme typické hodnoty. Z výsledků Density operátoru si pak můžeme prohlédnout, jak shlukovací algoritmus rozdělil data do jednotlivých shluků a prohlédnout si typické hodnoty ve shlucích. Jako ukázku uvedu graf zobrazující atribut stáří auta (car_age), kterým zákazník přijel. Atribut je ze souboru dat Allstate Purchase Prediction Challenge popsaného v podkapitole 7.1.2. Z grafu lze vidět, jak shlukovací algoritmus rozdělil stáří aut do jednotlivých shluků.



obr. 38 – Graf hustoty stáří auta

Density operátor je rychlý i na větších datech. Pro 20 000 instancí operátoru nezabere hledání typických hodnot ani 1 vteřinu.

Závěr

Tato diplomová práce je zaměřena na shlukovou analýzu dat a vyhodnocení shlukování. Seznámil jsem se s teorií výpočtu podobnosti a nepodobnosti pro nominální data využívaná shlukovacími metodami. Následovalo prozkoumání různých shlukovacích metod počítajících právě s nominálními daty, ať už to byly hierarchické metody shlukování či nehierarchické metody shlukování. Díky těmto znalostem mohly přijít na řadu výpočty se shlukovanými daty, tedy vnitro-shluková evaluace dat. Poznal jsem mnoho různých metod pro evaluaci shlukovaných dat a po konzultaci s vedoucím práce jsme se dohodli, které metody hodnocení budou implementovány. Vybral jsem čtyři metody vyhodnocení shlukování, a to konkrétně výpočet hodnot Rand indexu, Dunn indexu, siluety a určení typických hodnot uvnitř shluků, včetně grafů hustoty zastoupení jednotlivých proměnných ve shlucích. Po získání těchto potřebných teoretických znalostí přišlo na řadu seznámení se softwarem RapidMiner a implementační část diplomové práce.

Nejprve bylo nutné naučit se pracovat se samotným RapidMinerem. To netrvalo dlouho, protože pro RapidMiner jsou vytvořené srozumitelné tutoriály, se kterými jde učení rychle. Složitější část přišla na řadu při shánění zdrojových kódu RapidMineru. Zdrojové kódy několikrát měnily své uložiště, bylo tedy nutné zjistit, které uložiště je oficiální s nejaktuálnější verzí. Po získání zdrojových kódů jsem začal zkoumat, jakým stylem bude nejlepší RapidMiner rozšířit. Nakonec jsem se rozhodl pro možnost implementace pluginu, který se do RapidMineru přidá jako dodatečná knihovna, než přímo zasahovat do zdrojových kódu RapidMineru. Výhodou mé volby je snadné přidání nových operátorů do RapidMineru, naopak nevýhodou byla nepřístupnost všech dodatečných knihoven, které RapidMiner využívá a možnost testování pouze ve spuštěném RapidMineru.

Pro vybrané metody vnitro-shlukové evaluace jsem naimplementoval čtyři operátory. Konkrétně operátor Rand Index, operátor Dunn Index, operátor Silhouette a operátor Density. Mým cílem bylo, aby operátory využívaly co nejvíce již implementované třídy RapidMineru, abych zbytečně nepřidával duplicitní funkcionalitu. Z tohoto důvodu značnou část práce zabral průzkum zdrojového kódu RapidMineru. Svou pozornost jsem věnoval především algoritmům pro shlukování implementovaných v RapidMineru a dále základním mechanizmům práce RapidMineru s operátory, abych věděl, jak bude nejvhodnější mé operátory pro RapidMiner implementovat. Vyznat se ve zdrojovém kódu bylo složité převážně z toho důvodu, že jednak RapidMiner není nijak zdokumentovaný a jelikož se jedná o open source projekt, tudíž jej muže rozšiřovat spousta lidí, kód potom nemá jednotnou strukturu. Při prohledávání zdrojových kódů jsem mimo jiné zjistil, že všechny shlukovací metody v RapidMineru používají jeden velký balík obsahující sadu podobnostních metrik. Tato sada metrik obsahovala spoustu různých metod výpočtu podobností jak pro nominální data, tak pro numerická a smíšená data. Po tomto zjištění jsem se domluvil s vedoucím práce, že mnou implementované operátory nebudou zaměřeny pouze na nominální data, ale zpracují i numerická a

smíšená data. Dostudoval jsem tedy podobnostní metriky pro numerická a smíšená data včetně shlukovacích metod zpracovávajících tyto data. Díky tomuto jsou mé operátory mnohem komplexnější a dají se bez problémů použít na všechny typy dat.

Při implementaci operátorů Rand Index a Dunn Index jsem si vyzkoušel základní principy přidání nového operátoru do RapidMineru, práci se vstupními a výstupními porty, omezení pro porty, kontrolu dat na vstupních portech, práci s nastavitelnými parametry, výpis vypočtených hodnot, nastavení vzhledu operátorů, zatřízení do výběrového menu mezi ostatní operátory atd. V případě operátorů Silhouette a Density jsem se dále naučil vykreslovat data v několika různých zobrazeních. Pro tyto operátory je možné kromě textového výstupu zobrazovat též grafy. K tomu využívám grafické knihovny již vložené do RapidMineru. Mimo samotné vykreslení grafů lze u obou operátorů s grafy dále pracovat, a to konkrétně měnit řazení dat v grafech, přebarvovat graf podle různých atributů apod.

V poslední fázi diplomové práce jsem se věnoval ověření správnosti hodnot vypočtených mými operátory a výkonnostnímu testování. Kontrolu správnosti výsledků jsem prováděl porovnáním proti jiným výpočetním programům. Vypočtené hodnoty mými operátory jsem kontroloval s výsledky vypočtenými programy Matlab a SYSTAT a výsledné hodnoty se shodovaly. Při výkonnostních testech jsem využíval soubor dat Allstate Purchase Prediction Challenge ze serveru Kaggle.com. Jedná se o soubor dat obsahující desetitisíce instancí představující nákupy zákazníků v obchodech. Soubor dat je z velké části reprezentován nominálními hodnotami. Z výkonnostních testů jsem zjistil, že operátory zvládají bez problému zpracovat desetitisícové soubory dat v rámci několika minut.

Závěrem bych ještě rád podotkl, že tato problematika mě velmi zaujala a shlukování a shlukové algoritmy považuji za pozoruhodné téma s velkým potenciálem pro další výzkum.

Literatura a použité zdroje

[1] TSUNENORI, Ishioka. 2005. An expansion of X-means for automatically Determining the optimal number of clusters: : Progressive interations of K-means and merging of the clusters [online]. : 6 [cit. 2015-05-06]. Dostupné z: http://www.rd.dnc.ac.jp/~tunenori/doc/487-053.pdf

[2] KELBEL, Jan a David ŠILHÁN. 2005. Shluková analýza. Shluková analýza [online]. : 11 [cit. 2015-05-06]. Dostupné z: http://gerstner.felk.cvut.cz/biolab/X33BMI/slides/KMeans.pdf

[3] KUČERA, Jiří. 2008. Shluková analýza. KUČERA, Jiří. Shluková analýza [online]. [cit. 2015-05-06]. Dostupné z: http://is.muni.cz/th/172767/fi_b/5739129/web/web/main.html

[4] K-means clustering. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-06]. Dostupné z: http://en.wikipedia.org/wiki/K-means_clustering

[5] Cluster analysis. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-06]. Dostupné z: http://en.wikipedia.org/wiki/Cluster_analysis

[6] VELMURUGAN, T. a T. SANTHANAM. 2010. Computational Complexity between K-Means and K-Medoids Clustering Algorithms for Normal and Uniform Distributions of Data Points. Journal of Computer Science [online]. 6(3): 363-368 [cit. 2015-05-06]. DOI: 10.3844/jcssp.2010.363.368. ISSN 1549-3636.

[7] SINGH, Shalini S a N C CHAUHAN. 2011. K-means v/s K-medoids: A Comparative Study [online]. [cit. 2015-05-06]. Dostupné z: http://www.bvmengineering.ac.in/docs/published%20papers/cpit/cpit/201405.pdf

[8] BEN-HUR, Asa, David HORN, Hava T. SIEGELMANN a Vladimir VAPNIK. 2001. Support Vector Clustering [online]. [cit. 2015-05-06]. Dostupné z: http://www.jmlr.org/papers/volume2/horn01a/horn01a.pdf

[9] Hierarchical clustering. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-06]. Dostupné z: http://en.wikipedia.org/wiki/Hierarchical_clustering

[10] ŘEZANKOVÁ, Hana, Luboš MAREK, Michal VRABEC, Lukáš KALENSKÝ a Pavel ŘEZANKA. 2000. Typy proměnných. IASTAT: Interaktivní učebnice statictiky [online]. [cit. 2015-05-06]. Dostupné z: http://iastat.vse.cz/typy_promennych.html

[11] Shluková analýza. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-06]. Dostupné z: http://cs.wikipedia.org/wiki/Shluková_analýza

[12] LUKASOVÁ, Alena a Jana ŠARMANOVÁ. 1985. Metody shlukové analýzy. 1. vyd. Praha: Státní nakladatelství technické literatury, 210 s.

[13] ŘEZANKOVÁ, Hana, Dušan HÚSEK a Václav SNÁŠEL. 2009. Shluková analýza dat. 2., rozš. vyd. Praha: Professional Publishing, 218 s. ISBN 978-80-86946-81-8.

[14] RapidMiner. 2014. RapidMiner [online]. [cit. 2015-05-06]. Dostupné z: https://rapidminer.com

[15] RapidMiner. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-06]. Dostupné z: http://en.wikipedia.org/wiki/RapidMiner

[16] How to Extend RapidMiner. 2014. RapidMiner [online]. [cit. 2015-05-06]. Dostupné z: https://rapidminer.com/wp-content/uploads/2014/10/RapidMiner-extensions.pdf

[17] Taxonomie. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-06]. Dostupné z: http://cs.wikipedia.org/wiki/Taxonomie

[18] Taxonomy (general). 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-06]. Dostupné z: http://en.wikipedia.org/wiki/Taxonomy_(general)

[19] Taxonomy (biology). 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-06]. Dostupné z: http://en.wikipedia.org/wiki/Taxonomy_(biology)

[20] HORÁK, Jiří. 2002. Shluková analýza. HORÁK, Jiří. Institut Geoinformatiky [online]. [cit. 2015-05-06]. Dostupné z: http://gis.vsb.cz/pad/Kap_6/kap__6_5_1.htm

[21] KASPŘÍKOVÁ, Nikola. Shluková analýza vícerozměrných dat v programu R: - příklad použití metod PAM, CLARA a fuzzy shlukové analýzy [online]. [cit. 2015-05-06]. Dostupné z: http://data.tulipany.cz/ClustR.pdf

[22] DUDA, Richard O. 2001. Pattern classification. 2nd ed. New York: J. Wiley, xx, 654 s. ISBN 04-710-5669-3. Dostupné také z: https://books.google.cz/books?id=Yte2cXVES9EC

[23] WITTEN, I, Eibe FRANK a Mark A HALL. 2011. Data mining: practical machine learning tools and techniques. 3rd ed. Amsterdam: Morgan Kaufmann, xxxiii, 629 s. Morgan Kaufman series in data management systems. ISBN 978-0-12-374856-0. Dostupné také z: https://books.google.cz/books?id=bDtLM8CODsQC&pg=PT11&dq=978-0-12-374856-0&hl=cs&sa=X&ei=cWpLVZqoIbGu7Absg4GQCQ&ved=0CE4Q6AEwBg#v=onepage&q=clustering&f=false

[24] GOWER, J. C. 1971. A General Coefficient of Similarity and Some of Its Properties. Biometrics. **27**(4). DOI: 10.2307/2528823.

[25] CHAVENT, Marie, Yves LECHEVALLIER a Olivier BRIANT. 2007. DIVCLUS-T: A monothetic divisive hierarchical clustering method. Computational Statistics [online]. **52**(2): 687-701 [cit. 2015-05-07]. DOI: 10.1016/j.csda.2007.03.013. Dostupné z: https://hal.archives-ouvertes.fr/hal-00260963/document

[26] GOWER, J. C., G. J. S. ROSS a Alexander MEHLER. 1969. Minimum Spanning Trees and Single Linkage Cluster Analysis: Introducing Context-Sensitivity into the Generation of Spanning Trees. Applied Statistics [online]. **18**(1): 217-232 [cit. 2015-05-07]. DOI: 10.1007/978-3-540-77978-0_11. Dostupné z: http://www.jstor.org/discover/10.2307/2346439?uid=3737856&uid=2&uid=4&sid=21106740632373

[27] Single-linkage clustering. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-07]. Dostupné z: http://en.wikipedia.org/wiki/Single-linkage_clustering

[28] Complete-linkage clustering. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-07]. Dostupné z: http://en.wikipedia.org/wiki/Complete-linkage_clustering

[29] MELOUN, Milan a Jiří MILITKÝ. Přednosti analýzy shluků ve vícerozměrné statistické analýze [online]. [cit. 2015-05-07]. Dostupné z: http://meloun.upce.cz/docs/publication/152.pdf

[30] K-means clustering. OnMyPhD [online]. [cit. 2015-05-07]. Dostupné z: http://www.onmyphd.com/?p=k-means.clustering

[31] HÖPPNER, Frank. 1999. Fuzzy cluster analysis: methods for classification, data analysis, and image recognition. Chichester ; New York: J. Wiley, x, 289 p. ISBN 04-719-8864-2.

[32] Spanning tree. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-07]. Dostupné z: http://en.wikipedia.org/wiki/Spanning_tree

[33] HYNAR, Martin. 2003. Metody shlukovaní [online]. [cit. 2015-05-07]. Dostupné z: http://www.fit.vutbr.cz/study/courses/ZZD/public/seminar0304/Shlukovani1-text.pdf

[34] Silhouette (clustering). 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-07]. Dostupné z: http://en.wikipedia.org/wiki/Silhouette_%28clustering%29

[35] Evaluation of clustering. 2009. The Stanford Natural Language Processing Group [online]. [cit. 2015-05-07]. Dostupné z: http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

[36] Rand Index. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-07]. Dostupné z: http://en.wikipedia.org/wiki/Rand_index

[37] Mode (statistics). 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-07]. Dostupné z: http://en.wikipedia.org/wiki/Mode_(statistics)

[38] Arithmetic mean. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-07]. Dostupné z: http://en.wikipedia.org/wiki/Arithmetic_mean

[39] Median. 2001-. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-07]. Dostupné z: http://en.wikipedia.org/wiki/Median

[40] RapidMiner: data mining use cases and business analytics applications. 2013. Boca Raton: Taylor, liv, 465 s. Data mining and knowledge discovery series. ISBN 978-1-4822-0549-7.

[41] Iris Data Set. UCI: Machine Learning Repository [online]. [cit. 2015-05-07]. Dostupné z: https://archive.ics.uci.edu/ml/datasets/Iris

[42] Allstate Purchase Prediction Challenge. Kaggle [online]. [cit. 2015-05-07]. Dostupné z: https://www.kaggle.com/c/allstate-purchase-prediction-challenge/data

[43] AJAY, K.P. Soman; Shyam Diwakar; V. 2000. Insight into Data Mining: Theory and Practice. New Delhi: Prentice Hall of India. ISBN 978-812-0328-976.

[44] MORIK, Katharina a Jan CZOGALLA. Rapid Development of RapidMiner Extensions [online]. [cit. 2015-05-07]. Dostupné z: http://www-ai.cs.uni-dortmund.de/SOFTWARE/RMD/Rapid%20Development%20of%20RapidMiner%20Extensions.pdf

[45] CHISHOLM, Andrew. 2013. Exploring data with RapidMiner: explore, understand, and prepare real data using RapidMiner's practical tips and tricks. 1st ed. Birmingham: Packt Publishing, 148 s. ISBN 978-178-2169-338.

Obsah CD

- Shlukovací algoritmy pro nečíselná data.pdf
- Shlukovací algoritmy pro nečíselná data.docx
- Desky.pdf
- Desky.docx
- Zdrojové kódy
  - rapidminer-extension-cluster-evaluation
    - …
- Plugin pro RapidMiner
  - rapidminer-extension-cluster-evaluation-1.0.0-all.jar

Příloha A

Zdrojové kódy rozšíření pro RapidMiner

Kompletní zdrojové kódy jsem umístil na server GitHub.com. Kódy jsou veřejné a kdokoli si může zdrojová kódy stáhnout. Kromě zdrojových kódů je zde možné získat vytvořený plugin, který se jednoduše přidá do RapidMineru jako knihovna. URL pro stažení zdrojových kódů: https://github.com/LurtzZz/RapidMiner-cluster-evaluation .

Úprava zdrojových kódů

Pro úpravu zdrojových kódu je nutné mít nainstalovaný Gradle verze 2.1 nebo novější verzi. V případě, že používáte vývojové prostředí Eclipse import projektu lze provést následovně:

- Vybereme **File > Import**
- Zobrazí se okno **Select**, kde vybereme **Gradle > Gradle Project** a klikneme na tlačítko **Next**
- Zobrazí se okno **Import Gradle Project**, kde do pole **Root Folder** vložíme cestu ke zdrojovým kódům
- Klikneme na tlačítko **Build Model** tím se sestaví model
- Klikneme na tlačítko **Finish**

Vložení pluginu do RapidMineru

Vytvořený plugin lze nalézt v projektu na dvou místech:

- rapidminer-extension-cluster-evaluation\rapidminer-studio\lib\plugins
- rapidminer-extension-cluster-evaluation\rapidminer-extension-cluster-evaluation\build\libs

Generovaný název pluginu je **rapidminer-extension-cluster-evaluation-1.0.0-all.jar**. Aby RapidMiner mohl načíst plugin je nutné nalézt umístění instalovaného RapidMineru v počítači (většinou *C:\Program Files (x86)\RapidMiner*\) a nakopírovat **rapidminer-extension-cluster-evaluation-1.0.0-all.jar** do složky *RapidMiner5\lib\plugins*\. Při následujícím spuštěním RapidMineru se automaticky načtou i operátory z přidaného pluginu.