TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Hledání shluků podobných dopravních nehod

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Milan Kováčik**
Vedoucí práce: Ing. Bc. Marián Lamr

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Searching for cluster of similar car accidents

Master thesis

Study programme: N2612 – Electrotechnology and informatics
Study branch: 1802T007 – Information technology

Author: **Bc. Milan Kováčik**
Supervisor: Ing. Bc. Marián Lamr

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Milan Kováčik**
Osobní číslo:	**M14000165**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Hledání shluků podobných dopravních nehod**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Pomocí vhodného nástroje prozkoumejte a připravte data o dopravních nehodách na území ČR pro další analýzu.
2. Seznamte se s metodami shlukové analýzy a vyberte vhodné algoritmy pro vytváření shluků nehod.
3. Vytvořte nástroj pro vyhledávání specifických shluků dopravních nehod.
4. Navrhněte vhodný způsob vizualizace shluků dopravních nehod.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **40–50 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **ŘEZANKOVÁ, Hana, Dušan HÚSEK a Václav SNÁŠEL. Shluková analýza dat. 2., rozš. vyd. Praha: Professional Publishing, 2009, 218 s. ISBN 9788086946818.**

[2] **HAN, Jiawei a Micheline KAMBER. Data mining: concepts and techniques. San Francisco: Morgan Kaufmann Publishers, c2001. Morgan Kaufmann series in data management systems. ISBN 1558604898.**

[3] **BERKA, Petr. Dobývání znalostí z databází. Praha: Academia, 2003. 366s. ISBN 80-200-1062-9**

[4] **SCHILDT, Herbert. Mistrovství - Java. 1. vyd. Brno: Computer Press, 2014. Mistrovství. ISBN 978-80-251-4145-8.**

Vedoucí diplomové práce: **Ing. Marián Lamr**
Ústav mechatroniky a technické informatiky

Konzultant diplomové práce: **RNDr. Klára Císařová, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání diplomové práce: **10. října 2016**

Termín odevzdání diplomové práce: **15. května 2017**

Technická univerzita v Liberci
L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 25. 9. 2017

Podpis: Vavřinec Hilar

Abstrakt

Tato diplomová práce se zabývá hledáním nebezpečných dopravních úseků v České republice. K jejich nalezení je vytvořen analytický nástroj pro hledání shluků dopravních nehod s možností zkoumat jakýkoliv vstupní soubor s validními GPS pozicemi. V tomto nástroji si uživatel může zvolit z několika metod na hledání nebezpečných shluků. Filtrovat data pro vstup do metody lze pomocí jakéhokoliv atributu obsaženého ve vstupním souboru a podle všech hodnot daného atributu je automaticky vybrán typ filtru, jako například časový nebo číselný rozsah.

Jelikož se jedná o analytický nástroj, je možné každý nalezený shluk analyzovat. Při této analýze jsou ve shluku hledány společné příčiny nehody na vybraných atributech. Výstupem jsou pak asociační pravidla, která slouží k detekci předpokladů ke vzniku nehody v daném místě.

Klíčová slova: shluková analýza, DBSCAN, OPTICS, DENCLUE, asociační pravidla, Apriori, doprava, nehody, hustota dopravy

Abstract

This diploma thesis work deals with the search for dangerous stretches of roads with high incidence of traffic accidents in the Czech Republic. An analytical tool has been created in order to search for areas of high traffic accident clusters. Any input file with valid GPS positions can be researched. Several methodS may be applied in order to search for these accident clusters. Data can be filtered for input to methods by any attribute in the file. A filter is selected automatically by values in the attribute. There are a few types of filters, for example, time range or numerical range.

Any found cluster can be analyzed with this tool. This analysis is supposed to find common causes of traffic accidents in a cluster in selected attributes. The output of this analysis are association rules which serve to detect the assumption of the occurence of an accident in a selected cluster.

Key words: clustering analysis, DBSCAN, OPTICS, DENCLUE, association rules, Apriori, traffic, accidents, traffic density

Poděkování

Tímto bych rád poděkoval svému vedoucímu diplomové práce Ing. Bc. Mariánu Lamrovi za konzultace a příkladné vedení při řešení této práce. Dále bych rád poděkoval rodině a přátelům za podporu během celého studia.

Obsah

1 Úvod

Tato diplomová práce navazuje na semestrální projekt, kde bylo řešeno stahování dat o dopravních nehodách v České republice a jejich následné zobrazení v mobilní aplikaci. Data o nehodách poskytuje Ministerstvo dopravy od roku 2007 pomocí vektorové mapy[14]. Součástí dat je přibližně 50 atributů týkajících se vzniku a následků dopravní nehody, včetně GPS souřadnic.

Hlavním úkolem této diplomové práce je vytvořit analytický nástroj pro hledání shluků dopravních nehod v České republice. Velký důraz je kladen na univerzálnost řešení, tak aby nástroj dokázal zpracovávat jakákoliv data obsahující GPS souřadnice. Primárně by měl sloužit ke hledání shluků dopravních nehod. Proto je nutné vytvořit prostředí, ve kterém bude jednoduché data na vstupu filtrovat. Toto filtrování pak musí platit všeobecně pro jakákoliv vstupní data. Jelikož nástroj bude zaměřený na hledání shluků musí být připraven především na data obsahující číselné hodnoty, datumy, časy, ale i prostý text.

V první části budou implementovány vybrané algoritmy na hledání shluků dopravních nehod. Více různých algoritmů slouží především pro kontrolu výsledků, ale také k vyřešení problémů, které postihují některé algoritmy. Většina jednodušších algoritmů má veliké problémy s nalezením smysluplných shluků v oblastech s různou hustotou bodů.

Důležitou součástí je také analýza nalezených shluků. V této části je potřeba hledat jejich společné vlastnosti. K tomuto účelu byl vybrán algoritmus Aprior, který hledá asociační pravidla. Jako vstup pro tento algoritmus jsou zbylé atributy o vzniku a následcích nehod v jednom shluku. Hledání asociačních pravidel musí být dostatečně robustní, aby si uživatel mohl zvolit v jakých atributech chce hledat pravidla a také jaké atributy pro něho mají prioritu.

Všechny dosažené výsledky hledání a analýzy budou zobrazeny přímo v nástroji, ale také zde musí být možnosti je exportovat pro další použití. Zde je potřeba mít možnost zobrazit nebo exportovat shluky včetně šumu - tedy nehod, které nepatří

do žádného shluku. Podle šumu lze pouhým zobrazením v mapě pozorovat, jestli shluky dávají smysl, nebo jestli je potřeba upravovat vstupní parametry.

Jelikož je tento nástroj určen primárně ke hledání shluků nehod v České republice, nabízí se možnost, jak využít data o sčítání hustoty dopravy. V době psaní této práce bylo sčítání naposledy provedeno v roce 2010[16]. Je tedy nezbytné rozšířit již dostupná data o dopravních nehodách o hustotu dopravy v místě nehod a optimalizovat některé algoritmy tak aby byly na hustotu dopravy citlivé.

2 Získávání znalostí z databází

Je proces hledání informací nebo vzorů na velkých datech či ve velkých databázích. Tyto informace jsou na první pohled skryty a k jejich nalezení je potřeba vykonat několik kroků. Jako první je třeba zjistit strukturu dat, jakých hodnoty a datových typů jednotlivé položky nabývají. Dále je potřeba si uvědomit, co v datech chceme hledat. Podle tohoto zjištění lze aplikovat některou již známou metodu a případně ji optimalizovat pro specifické požadavky. Nejčastěji používané metody se skládají ze statistických výpočtů, strojového učení, nebo umělé inteligence. Nejedná se tedy o jednoduchý výběr části dat, ze kterého by byly ihned získány informace. Jedná se o proces nad celými daty a až po zpracování všech dat lze dojít k nějakému závěru.

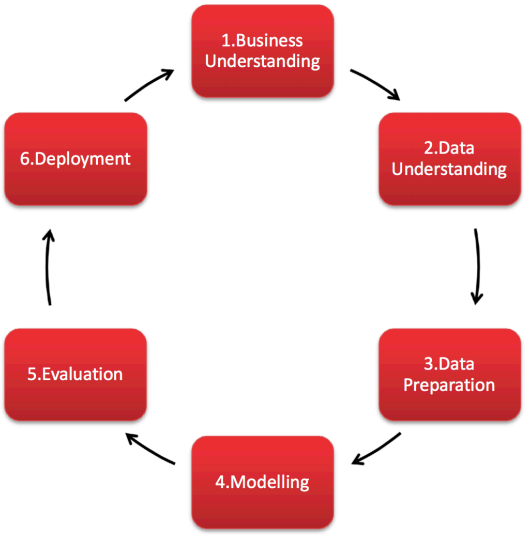
První takovéto metody probíhaly manuálně, jako například Bayesovska věta nebo regresní analýza. Až s rozvojem počítačové techniky a nástupem relačních databází se rapidně zvýšil objem uložených dat a vznikala potřeba z dat dolovat některé informace. Zpočátku šlo spíše o analytické nástroje. Až v 90-tých letech a nástupem datových skladů, které shromažďovaly data z různých zdrojů, se začaly objevovat automatizované metody pro hledání vzorů. S přibývajícím množstvím metod s různým zaměřením, bylo stále těžší celý postup popsat.[1][5]

2.1 Metodologie

Slouží především k popisu celého procesu, který je rozdělen do několika kroků. Tyto kroky jsou navzájem propojeny a je možné vracet se tam a zpět k dosažení optimálních výsledků v jakémkoliv kroku. Tento postup má sloužit především jako návod, jak co nejefektivněji a nejrychleji získat informace z dat.

2.1.1 Metodologie CRIPS-DM

CRISP-DM (CRoss – Industry Standard Process for Data Mining) vznikla jako projekt Evropské komise pro standardizaci dolování dat. Oproti SEMMA se navíc



Obrázek 2.1: Průběh metodologie CRISP-DM (převzato z [3]).

zabývá i samotným využitím dat v reálném životě. Můžeme říct, že se jedná o obchodní model, který zahrnuje jak prvotní požadavky a cíle, tak závěrečné hodnocení a samotné nasazení do praxe.[3]

- **Business understanding** - Hlavní je porozumět problematice a stanovit si své cíle i finanční možnosti. Snažíme se nalézt správnou otázku co chceme hned na začátku tak, abychom minimalizovali riziko případného neúspěchu špatně zadanými kritérii.
- **Data understanding** - Zisk dat nebo přístup k datům. V této fázi se provádí základní analýza dat. Zjišťuje se typ formátu dat, jakých nabývají hodnot, množství dat, atd. Dále zjišťujeme základní vztahy mezi atributy.
- **Data preparation** - Obvykle jedna z nejnáročnějších fází. Data bývají často v různých datových zdrojích s různými formáty je potřeba je transformovat do jedné vstupní množiny. Odfiltrovat data s chybějícími hodnotami a vybrat pouze ty atributy, které potřebujeme. Někdy je potřeba data standardizovat pokud máme atributy v jiných formátech.
- **Modeling** - Na začátku modelování vybereme algoritmus (nebo algoritmy). Následně je nutné sestavit testovací mechanismus, který bude hodnotit výsledky. V průběhu modelování jsou často měněny vstupní parametry, tyto změny

musí být zaznamenány s odůvodněním. Po otestování jsou modely oceněny podle parametrů zadání z prvního kroku.

- **Evaluation** - Hodnocení výsledků jak model obstál v dosažení obchodních cílů. Pokud jsou výsledky dostatečné pro obchodní záměry, provádí se kompletní revize, jestli nebyly přehlédnuty některé důležité faktory. Na základě těchto výsledků rozhodneme co dál, jestli se vrátit k některému z předešlých kroků, nebo pokračovat k nasazení.

- **Deployment** - Volíme strategii pro nasazení během této fáze nesmíme zapomenout na vytvoření postupů pro kontrolu a údržbu modelu. Pokud se model používá denně, je tento krok nezbytný. Na závěr by měla být sepsána závěrečná zpráva o výsledcích celého procesu.

2.2 Vybrané data miningové úlohy

Metody dolování dat můžeme rozdělit do několika kategorií podle typu použitých operací nad daty. Samotné rozdělení a počet typů úloh není nikde definováno a nelze říci jaké je to správné rozdělení.

2.2.1 Klasifikace

Je typ úlohy kde jsou vstupní data zařazena do tříd nebo kategorií. Následně je sledován nějaký target (cíl) v datech, který ukazuje jaký byl závěr. Nejlépe si to můžeme představit na datech o bankovních půjčkách, kde jeden řádek je jeden klient. Klient má několik parametrů, jako je věk, plat, zdravotní omezení, atd. Jako poslední je sloupec s daty, zda půjčku zaplatil či nikoli. Z těchto dat je sestaven model pro nové zájemce a půjčku. Podle toho do jakých kategorií nový žadatel zapadne lze snadno zjistit do jakého targetu bude patřit.

2.2.2 Regresní analýza

Je dost podobná klasifikaci, ale snaží se predikovat číslo ne skupinu. Stejně jako u klasifikace je nutné znát i výsledky jednotlivých položek. Její použití je velice široké, lze predikovat šanci na úspěšnost operace pacienta, nebo odhadnout cenu domu. U stanovení ceny domu sledujeme několik důležitých faktorů jako je velikost

domu, stáří, lokalita, stav. Pokud máme data o prodeji podobných domů, lze celkem přesně stanovit cenu takového domu.

2.2.3 Shluková analýza

Do této skupiny patří metody, které hledají podobné položky a sdružují je do skupin, kde jsou si nejpodobnější položky. Shlukování probíhá bez znalosti předchozích výsledků, takže v datech nemáme cílovou proměnnou. Shlukovat můžeme různé typy záznamů, například zákazníky v internetovém obchodě. Zákazníci pak budou rozděleni do skupin podle vzájemné podobnosti, která nemusí být na první pohled patrná. Tímto typem úlohy se budeme zabývat v kapitole 3.

2.2.4 Asociační pravidla

Nejznámější typ úlohy, co se dolování dat týče, slouží především k marketingovým průzkumům. Nejčastější úlohou je analýza nákupního košíku, kde jsou hledány spojitosti mezi dvěma nebo více položkami. Díky těmto pravidlům můžeme zákazníkovi cíleně nabízet další zboží. Jedná se tedy o prediktivní metody, kde se v datech hledají pravidla. Ta jsou následně otestována na jiné sadě dat. Nakonec jsou použita k nabízení souvisejících produktů, nebo k tvorbě newsletterů. Tento trend je už běžnou praxí velikých internetových obchodů, které nutí každého zákazníka k registraci, tak aby získali další potřebné informaci pro hledání nových pravidel.[1][5]

Na počátku 90. let tento termín zpopularizoval Agrawal na úloze analýza nákupního košíku. V ní se zabýval souvislostmi mezi produkty, co si zákazníci v supermarketech koupili. Pokud košík obsahoval například párek a chleba, byla veliká pravděpodobnost, že bude obsahovat i hořčici. Asociační pravidla jsou tedy ve tvaru předpoklad (co již máme v košíku), který implikuje závěr (co si nejspíš koupíme). Můžeme si je představit jako klasickou konstrukci IF THEN, jestliže mám párek, pak si nejspíš koupím chleba.

$$parek, chleba \Rightarrow horcice \tag{2.1}$$

2.3 Charakteristiky pravidel

U asociačních pravidel nás nejčastěji zajímá jak často byl splněn předpoklad a závěr. Ale zajímají nás i případy, kdy byl splněn předpoklad, ale závěr nikoli.

Z tabulky [2.1] získáme dvě nejdůležitější charakteristiky asociačních pravidel, podporu (support) a spolehlivost (confidence). Podpora je definována jako počet položek, které splňují předpoklad i závěr, ku všem položkám v košíku.

$$support(a) = \frac{a}{a+b+c+d} * 100(\%) \tag{2.2}$$

Confidence nám udává jak je pravidlo spolehlivé. V podstatě nám říká v kolika procentech případů, kdy byl splněn předpoklad, byl splněn také závěr.

$$confidence(a) = \frac{a}{a+b} * 100(\%) \tag{2.3}$$

2.4 Apriori

Je algoritmus na hledání asociačních pravidel, na vstupu přijímá dva vstupní parametry, minimální support a minimální confidenci. V prvním kroku projde celý soubor vstupních dat a vytvoří množinu kandidátů, tedy všech položek ve všech košících. Následně pro každého kandidáta spočítá support a odstraní ty kandidáty, které nesplňují zadaný support. [1][6]

Na tabulkách 2.2 je vidět průběh prvního kroku. V první tabulce máme vstupní data se čtyřmi nákupními košíky, z těchto košíků je vytvořena množina kandidátů C1 s vypočtenou četností ve všech košících. Při zadaném minimálním supportu 2 nám vznikne frekventovaná množina L1, ve které jsou pouze kandidáti s minimálním supportem.

	Závěr	¬Závěr	$\sum$
Předpoklad	a	b	r
¬Předpoklad	c	d	s
$\sum$	k	l	n

Tabulka 2.1: Kontingenční tabulka

Nákup	Položky
1	I1, I3
2	I1, I2, I4
3	I1, I4
4	I1, I3, I4

$\Rightarrow C1$

Položka	Počet
I1	4
I2	1
I3	2
I4	3

$\Rightarrow L1$

Položka	Počet
I1	4
I3	2
I4	3

Tabulka 2.2: Průběh prvního kroku Apriori

V dalším kroku se vytvoří množiny kandidátu C2, které obsahují dvě položky. Zde využijeme vlastnost Apriori a vezmeme pouze položky z předchozí frekventované množiny. Následuje stejný proces jako v prvním kroku.

$\Rightarrow C2$

Položky	Počet
I1, I3	2
I1, I4	3
I3, I4	1

$\Rightarrow L2$

Položky	Počet
I1, I3	2
I1, I4	3

Tabulka 2.3: Průběh prvního kroku Apriori

Stejný proces by následoval dokud lze generovat kandidáty, poté hledání frekventovaných množin končí. Následuje hledání pravidel z nalezených frekventovaných množin L. Například z množiny L2 můžeme vzít množinu I1, I4:

- **I1 → I4**

$$confidence\{I1 \rightarrow I4\} = \frac{support\{I1, I4\}}{support\{I1\}} * 100 = 3/4 * 100 = 75\% \quad (2.4)$$

- **I4 → I1**

$$confidence\{I4 \rightarrow I1\} = \frac{support\{I1, I4\}}{support\{I4\}} * 100 = 3/3 * 100 = 100\% \quad (2.5)$$

Z prvního pravidla (I1 → I4) je zřejmé, že pokud se v košíku objeví položka I1, pak s pravděpodobností 75 % bude ve stejném košíku i položka I4. Při výpočtu těchto pravidel se vyberou pouze pravidla splňující minimální confidenci zadanou na začátku algoritmu.

3 Shluková analýza

Algoritmy shlukové analýzy třídí objekty do shluků (clusterů) podle podobnosti objektů. Vytvořené shluky jsou tak tvořeny objekty, jejichž atributy jsou si velmi podobné. Zatímco s objekty mimo shluk mají nízkou podobnost atributů. Podobnost objektů je určena hodnotami jednotlivých atributů, pro určení podobnosti se často využívají vzdálenostní funkce.

Nevědomě shlukovou analýzu používá každý z nás. Rozdělujeme různé objekty do kategorií podle společných vlastností, například lidskou populaci dělíme na dvě skupiny, muž a žena. Tyto skupiny jsou jednoznačné a vznikají tak pouze dva shluky bez šumu. Pokud chceme jednotlivce dělit do skupin, musíme se rozhodovat na základě jakých atributů je budeme dělit: věk, výška, vzdělání, náboženství, atd. Čím více atributů máme tím, je složitější definovat shluk, proto vzniká spousta malých skupin kde, jsou si jednotlivé objekty maximálně podobné.

Shlukování probíhá pouze na základě podobnosti některých atributů. Nejedná se tedy o samoučící metody, a není tedy zapotřebí trénovací množina dat, nebo předdefinované skupiny, do kterých mají být objekty rozděleny. Naopak některé objekty nemusí být v žádné skupině a tvoří šum. Šum se zdá na první pohled zbytečný a nepodstatný, ale některé metody se zabývají právě hledáním odlehlých hodnot.

Následující kapitoly popisují jaké by dobrá shluková analýza měla mít vlastnosti, s jakými typy proměnných může pracovat a jak vstupní data předzpracovat.[9]

3.1 Vlastnosti shlukovacích metod

Každá shlukovací metoda by měla splňovat různé vlastnosti na základě jejího typu a použití. Typické vlastnosti shlukovacích metod jsou:[8]

- **Škálovatelnost:** Většina metod dobře pracuje s malým objemem dat. Ve skutečnosti je zapotřebí zpracovávat rozsáhlé databáze, proto je zapotřebí aby algoritmy byly vysoce škálovatelné.

- **Pracovat s šumem:** Data uložená v databázích často obsahují nekonzistentní data. Taková data musí metoda umět rozpoznat a nezahrnovat je do výsledných shluků, ani nesmí nijak ovlivnit vzniklé shluky. Takové záznamy jsou nejčastěji filtrovány před vstupem do metody, nebo jsou označeny jako šum.

- **Shluky různého tvaru:** Shlukování nejčastěji probíhá na základě vzdálenostních funkcí (Eukleidovská vzdálenot, Manhattonská vzdálenost, ...). To může vést k vytváření podobně velikých, hustých a kulovitých tvarů. Znemožňuje tak vytváření libovolných tvarů, které by lépe popisovaly objekty ve výsledných shlucích.

- **Různé typy vstupních atributů:** Ve většině shlukovacích metod se využívají numerická data. Někdy je ale zapotřebí data shlukovat na základě jiných datových typů, například binárních, ordinálních v některých případech i různého datového typu.

- **Minimální požadavky na znalost problému:** Především u metod, u kterých je nutné definovat vstupní parametry. Tyto parametry mohou značně ovlivnit kvalitu nalezených shluků. Často je také problém s nalezením optimálních vstupních parametrů.

- **Necitlivost na seřazení vstupních dat:** Pro stejná vstupní data mohou některé algoritmy najít zcela rozdílné shluk. Ve shlukové analýze jsou zapotřebí algoritmy, kterým nezáleží na pořadí dat na vstupu.

- **Vytvářet shluky na vysokodimenzionálních datech:** Zpracovat nízkodimenzionální data (do 3 atributů) zvládne většina algoritmů. Nejzajímavější jsou však ty, které dokáží zpracovat data s více atributy (řádově desítky atributů).

- **Shlukování na základě omezení:** Často je nutné hledat takové shluky, které odpovídají určitému omezení.

- **Srozumitelné výsledky:** Výstupem algoritmu by měly být srozumitelné a snadno interpretovatelné výsledky. Nemusí se jednat jen o shluky, ale i šum by měl být správně prezentován. Důležité proto je, aby nástroj uměl prezentovat výsledky algoritmu.

3.2 Podobnost a typy proměnných

Podobnost dvou objektů je ve shlukové analýze reprezentována vzdáleností těchto objektů. Výpočet vzdálenosti se liší podle datového typu atributů. V této kapitole jsou prezentovány nejčastěji používané typy proměnných.[1]

3.2.1 Intervalové proměnné

Jedná se o spojité hodnoty, které jsou rozděleny téměř lineárně. Typicky se jedná o proměnné jako je váha, výška, věk, atp. Podobnost dvou objektů se nejčastěji počítá pomocí vzdálenostních funkcí.

Problém intervalových je jednotka proměnné. Pokud se změní jednotka proměnné, může to mít vliv na výsledek, proto se často využívá převod na bezjednotkové proměnné pomocí standardizace. [Kapitola 4.3.1]

- **Eukleidovská vzdálenost:**

$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + ... + |x_{in} - x_{jn}|^2} \tag{3.1}$$

- **Minkowského vzdálenost:**

$$d(i,j) = (|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + ... + |x_{in} - x_{jn}|^p)^{\frac{1}{p}} \tag{3.2}$$

- **Manhattonská vzdálenost:**

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + ... + |x_{in} - x_{jn}| \tag{3.3}$$

Pro všechny vzdálenostní funkce musí platit následující pravidla:

1. $\mathbf{d(\ i\ ,\ j\ ) \geq 0}$ **:** Vzdálenost je vždy kladné číslo.
2. $\mathbf{d(\ i\ ,\ i\ ) = 0}$ **:** Vzdálenost objektu k sobě samému je 0.
3. $\mathbf{d(\ i\ ,\ j\ ) = d(\ j\ ,\ i\ )}$ **:** Vzdálenostní funkce je symetrická.
4. $\mathbf{d(\ i\ ,\ j\ ) \leq d(\ i\ ,\ h\ ) + d(\ j\ ,\ h\ )}$ **:** Vzdálenost objektů i a j nesmí být vyšší než součet vzdáleností i a j k objektu h. (podmínka triangularity).

	Objekt i		
Objekt j	1	0	$\sum$
1	q	r	q + r
0	s	t	s + t
$\sum$	q + s	r + t	p

Tabulka 3.1: Kontingenční tabulka pro porovnání binárních proměnných i a j.

3.2.2 Binární proměnné

Binární proměnné mohou nabývat pouze dvou stavů, nejčastěji 0 a 1, nebo true (pravda) a false (nepravda). Jednoduše si to lze představit na pohlaví člověka, existují pouze dvě možnosti: muž, nebo žena. Nezle nabývat jiného stavu. Podle váhy obou hodnot jsou binární proměnné rozděleny na symetrické a asymetrické.

- **Symetrické binární proměnné:** Obě hodnoty těchto proměnných mají stejnou pravděpodobnost výskytu a tudíž stejnou váhu. Jedná se o proměnné typu, již zmíněného stavu pohlaví člověka. Podobnost vypočtená na základě těchto proměnných je invariantní, nezáleží tedy jestli pohlaví muže označíme jako 0 nebo 1, vždy dosáhneme stejného výsledku. Pro výpočet podobnosti se nejčastěji používá jednoduchý koeficient shody:

$$d(i, j) = \frac{r + s}{q + r + s + t} \tag{3.4}$$

  Kde jednotlivé parametry funkce jsou popsány v následující kontingenční tabulce[3.1].

- **Asymetrické binární proměnné:** Jedná se o binární proměnné, kde každá hodnota má jinou váhu. Typickým příkladem je test určení onemocnění. Pozitivní výsledek u dvou osob má vyšší hodnotu než negativní test. Pokud provedeme více testů a zahrneme do vyhodnocení i symptomy onemocnění, můžeme posoudit s jakou pravděpodobností mají dvě osoby stejné onemocnění. Pro výpočet se nejčastěji používá Jaccardův koeficient, kde výsledek s vyšší váhou má hodnotu 0 a s nižší hodnotu 1. Počet negativních shod (parametr t) je pro výpočet nepodstatný, a tak není výpočtu zahrnut.

$$d(i,j) = \frac{r+s}{q+r+s} \tag{3.5}$$

3.2.3 Nominální proměnné

Jedná se o proměnné, které vychází z binárních proměnných. Na rozdíl od nich však nabývají více stavů, které jsou předem definované. Například ovoce lze dělit podle druhu: jablko, hruška, atd., přičemž tato množina je konečná a předem definovaná. Vzdálenost dvou objektů lze vypočítat podobně jako u binárních proměnných pomocí jednoduchého koeficientu shody:

$$d(i,j) = \frac{p-m}{p} \tag{3.6}$$

Kde p počet proměnných (sloupců v databázi) a m je počet shod u objektů i a j v těchto proměnných.

3.2.4 Ordinální proměnné

Ordinální proměnné mohou, podobně jako nominální, nabývat více hodnot. Ovšem hodnoty, kterých mohou nabývat jsou uspořádány, podle toho jak veliký význam mají. Například hodnocení náročnosti předmětu: velmi lehký, lehký, průměrně náročný, těžký, velmi těžký.

Jelikož ordinální proměnné nabývají M různých seřazených stavů, lze místo jejich stavů pracovat s jejich pořadím z intervalu $< 1; M_f >$. Pokud porovnáváme více atributů je dobré hodnoty transformovat do intervalu $< 0; 1 >$, tak aby měli všechny hodnoty stejnou váhu. Transformaci lze provést pomocí následujícího vzorce:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1} \tag{3.7}$$

Kde r_{if} je číselná hodnota pořadí pro stav proměnné objektu i. Pomocí této transformace lze s vypočtenými hodnotami z počítat stejně jako s intervalovými proměnnými.

3.2.5 Poměrové proměnné

jedná se o proměnné s nelineární stupnicí, nejčastěji s exponenciální, nebo logaritmickou. Vzdálenost mezi těmito proměnnými lze vypočítat třemi způsoby:

- Počítat s poměrovými proměnnými stejně jako s intervalovými. Tento způsob může zkreslovat výsledné shluky.
- Pomocí logaritmické transformace převést hodnoty následovně:

$$y_{if} = log(x_{if}) \tag{3.8}$$

  Kde x_{if} je hodnota proměnné f. Dále s vypočtenou hodnotou y_{if} lze pracovat jako s intervalovou proměnnou.

- Chovat se k poměrovým proměnným stejně jako ke spojitým ordinálním proměnným. A po převodu na interval $< 0; 1 >$ s nimi počítat stejně jako s intervalovými.

3.2.6 Proměnné různého typu

U shlukové analýzy obsahují data obsažená v databázi většinou více datových typů pro různé atributy. Jednou z možností jak tato data zpracovat, je provést více shlukových analýz. Tato metoda vede k výraznému zkreslení a proto je lepší zpracovávat všechny atributy současně.

Pro všechna data p atributů, lze určit vzdálenost mezi dvěma objekt i a j pomocí následujícího vzorce:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}} \tag{3.9}$$

Kde $\delta_{ij}^{(f)}$ je roven nule pokud nastane jedna ze dvou podmínek:

1. x_{if} nebo x_{jf} chybí
2. proměnná f je asymetrická binární proměnná a zároveň x_{if} i $x_{jf} = 0$

Jinak je $\delta_{ij}^{(f)}$ rovno 1.

Příspěvek $d_{ij}^{(f)}$ proměnné f se vypočítá podle typu proměnné:

- **Intervalové proměnné:**

$$d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}} \tag{3.10}$$

  Kde h jde přes všechny hodnoty proměnné f.

- **Binární nebo nominální proměnné:**

$$d_{ij}^{(f)} = \begin{cases} 0 & x_{if} = x_{jf} \\ 1 & x_{if} \neq x_{jf} \end{cases} \tag{3.11}$$

- **Ordinální nebo poměrová proměnná:** Vypočteme z_{if} a její hodnotu zpracujeme jako intervalovou proměnnou.

3.3 Předzpracování dat

Ještě před samotným dolováním dat je potřeba data pro algoritmy připravit. Tento proces, na základě požadavků odfiltruje nežádoucí hodnoty (null, odlehlé hodnoty, atd.). Následně se data transformují do optimálních hodnot pro další zpracování. K tomu jsou používány následující metody předzpracování dat.

3.3.1 Normalizace dat

Transformuje číselné atributy do předem specifikovaných intervalů, nejčastěji < -1.0, 1.0> nebo < 0.0, 1.0>. Lze použít metodu min-max, která transformuje data do podoby, kde minX je rovno 0.0 a maxX 1.0 pro zvolený interval < 0.0, 1.0>. Hodnoty mezi minX a maxX jsou přepočteny pomocí vztahu [3.9].

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.12}$$

3.3.2 Standardizace

Tuto metodu použijeme na jednotkové typy proměnných. Data se standardizují z důvodu změny jednotky, například pokud změníme jednotku vzdálenosti z metrů na milimetry, to může ovlivnit celý algoritmus. Některé algoritmy jsou na tyto změny náchylné, a tak je lepší tyto atributy převést na bezjednotkové proměnné.

Standardizaci můžeme provést výpočtem z-score. Nejprve ale musíme vypočítat střední odchylku s_f [4.2]

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|) \tag{3.13}$$

Kde x_{1f}, ..., x_{nf} je n hodnot proměnné f, m_f je střední hodnota f.
Pomocí s_f můžeme vypočítat z-score[4.3].

$$z_{if} = \frac{x_{if} - m_f}{s_f} \tag{3.14}$$

3.3.3 Čištění a redukce dat

Obě metody zahrnují redukci dat, čištění odstraňuje nekonzistentní záznamy a někdy také odlehlé hodnoty. Zatímco při redukci dat odstraňujeme nepotřebné atributy pro zvolený algoritmus.

3.4 Rozdělení shlukovacích metod

Výběr metody závisí na povaze dat a co v nich chceme hledat. Často vybereme více metod, porovnáváme výsledky a vybíráme takovou metodu, která nejvíce vyhovuje našim požadavkům. Metody, které lze vybrat jsou rozděleny do několika kategorií:

3.4.1 Metody založené na rozdělování

Jsou metody, kde rozdělujeme všechny objekty n do k. Počet tříd je stanovený na začátku a musí splňovat podmínku $k \geq n$. Další podmínkou je, že každý objekt n náleží právě jedné třídě k a každá třída k musí obsahovat alespoň jeden objekt n.

Obecně tyto metody začínají výběrem k objektů, kde každý objekt zapadne do jedné třídy, následuje roztřídění ostatních objektů do tříd na základě podobnosti. Až jsou všechny objekty v třídách, hledají se takové objekty, které nejlépe reprezentují své třídy. Poté se objekty přesouvají mezi třídami, tak aby se v jedné třídě nacházeli co nejpodobnější objekty.

Největší nevýhodou těchto metod je definice přesného počtu tříd, do kterých objekty zapadnou.

3.4.2 Hierarchické metody

Hierarchické metody vytváří stromovou strukturu shluků podle průběhu metody. Tyto metody můžeme rozdělit do dvou typů:

- **Shlukující** (zdola-nahoru) jsou založeny na principu spojování tříd, na začátku je každý objekt ve vlastní třídě. Následně se jednotlivé třídy spojují dohromady podle vzájemné podobnosti. Konec lze definovat počtem tříd, které mají být nalezeny, pokud takto podmínka není zadána spojí se všechny do jedné třídy.
- **Rozdělující** (shora-dolů) probíhají přesně naopak než shlukující. Na začátku jsou všechny objekty v jedné třídě a následně jsou rozdělovány. Ukončující podmínka je stejná a pokud není zadána, rozpadne se n objektů do k tříd, kde $k = n$.

Tyto metody mají několik nevýhod, hlavní z nich je výběr jak objekty shlukovat/rozdělovat. Pokud zvolíme špatnou metodu není, možné daný krok vzít zpět a objekty přesunout do jiné třídy. Podobně jako metody založené na rozdělování trpí stejnou vlastností, musí být zadán počet tříd.

3.4.3 Metody založené na mřížce

Objekty jsou rozděleny do buněk, které tvoří mřížku. Mřížka může být i víceúrovňová, záleží na struktuře dat. Nad touto mřížkou jsou prováděny všechny operace. Doba shlukování je závislá pouze na počtu buněk, ne na počtu objektů, proto jsou tyto metody používány pro velký objem dat.

Obecně lze říct, že průběh těchto metod probíhá rozdělením objektů do buněk. Následně se provede operace nad buňkami, ze kterých je vypočtena informace všech objektů v každé buňce. Pomocí informací obsažených v buňkách dochází ke shlukování více buněk a výsledkem jsou shluky objektů.

3.4.4 Metody založené na modelech

Pro tyto metody slouží jako vzor (model), nějaká matematická funkce, většinou se jedná o funkci generovanou nějakou pravděpodobnostní distribuční funkcí. Tyto metody se snaží pomocí tohoto vzoru nalézt shluky, které by co nejvíce odpovídaly vzoru.

3.4.5 Metody pro shlukování vysoce-dimensionálních dat

Předešlé metody jsou navrženy na malý počet atributů (dimenzí). S rostoucím počtem atributů dochází k řídnutí dat. Jednotlivé objekty jsou si stále vzdálenější a je velice složité najít shluky.

Pro vyřešení těchto problémů se nejčastěji používá výběr atributů. Ponecháme si pouze potřebné atributy pro naši úlohu. Další možností je transformovat data do menší dimenze. Při tomto kroku se využívá sumarizace některých atributů.

3.4.6 Ostatní metody

Dalším typem metod je shlukování založené na omezení, jedná se o typ úloh kde, uživatel klade omezení pro výsledné shluky.

Další velikou skupinou jsou metody založené na hustotě, které budou popsány dále v kapitole 4.

4 Metody shlukování založené na hustotě

Jsou metody, které vytváří shluky v místech s vysokou hustotou bodů. Naopak oblasti s nízkou hustotou bodů jsou považovány za šum. Na rozdíl od dříve popsaných metod, jako jsou například rozdělovací metody, nemusí objekt patřit do nějakého shluku. Díky tomu se metody založené na hustotě dají využít i k analýze odlehlých hodnot.

Nejznámějším představitelem je algoritmus DBSCAN a jeho vylepšení OPTICS. Oba tyto algoritmy byly vybrány k implementaci, navíc byl vybrán algoritmus DENCLUE, který umožňuje popsat shluky i pro vysocedimensionální data. Všechny zmíněné algoritmy budou popsány v této kapitole.

4.1 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) byl navržen roce 1996, za jeho vznikem stojí Martin Ester, Hans-Peter Kriegel, Jörg Sander a Xiaowei Xu. Algoritmus je založen na jednoduché myšlence, shlukovat body, které jsou blízko sebe.[10]

Pro pochopení principu algoritmu je nutné vysvětlit následující pojmy:

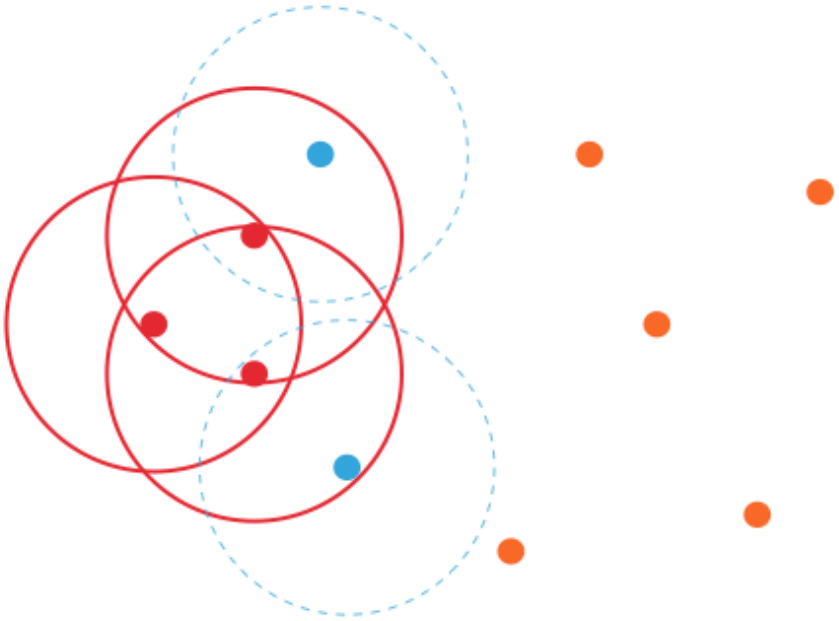
- Okolí bodu o poloměru ϵ, se nazývá jako ϵ-okolí bodu.
- Pokud je v ϵ-okolí bodu MinPts bodů, je tento bod označován jako jádrový.
- Jestliže o bodu p z množiny D řekneme, že je přímo dosažitelný na základě hustoty z objektu q, pak je bod p v ϵ-okolí bodu q a bod q je zároveň jádrový bod.
- Bod p je dosažitelný na základě hustoty z bodu q v množině D, jestliže existuje posloupnost bodu x_i, ..., x_n taková, že x_i je přímo dosažitelný na základě hustoty z bodu x_{i+1}. Pro $1 \leq \mathrm{i} \leq \mathrm{n}$, $x_i \in \mathrm{D}$.

- Bod p je spojený na základě hustoty s bodem q v množině D, jestliže existuje bod o, který je dosažitelný na základě hustoty s body p i q.

Průběh algoritmu je popsán v následujícím pseudokódu, do kterého vstupují tři parametry. D - pole bodů, ϵ a MinPts.

```
DBSCAN(D, eps, MinPts)
   for each unvisited point P in dataset D
      mark P as visited
      N = getNeighbors (P, eps)
      if sizeof(N) < MinPts
         mark P as NOISE
      else
         C = next cluster
         expandCluster(P, N, C, eps, MinPts)

expandCluster(P, N, C, eps, MinPts)
   add P to cluster C
   for each point P' in N
      if P' is not visited
         mark P' as visited
         N' = getNeighbors(P', eps)
         if sizeof(N') >= MinPts
            N = N joined with N'
      if P' is not yet member of any cluster
         add P' to cluster C

getNeighbors(P, eps)
   return all points within P's eps-neighborhood (including P)
```

V hlavní smyčce se procházejí všechny body, pokud je bod označen jako jádrový vzniká shluk. Tento shluk je rozšířen o další body v metodě expandCluster. V této metodě se iteruje skrz všechny sousední body, zde platí stejné pravidlo jako v hlavní smyčce, a to že pokud je některý bod vyhodnocen jako jádrový, jsou všechny jeho okolní body přidány do shluku.



Obrázek 4.1: DBSCAN, minPts = 4, ϵ = kružnice okolo bodu. Červené body jsou označeny jako jádrové. Modré body patří do shluku, ale nejsou jádrové. Oranžové body jsou označovány jako šum.

Hlavní nevýhodou DBSCANU je, že nedokáže najít smysluplné shluky v datech s různou hustotou. Dalším problémem je volba vstupních parametrů, kde i malá změna může vést k výrazné změně výsledku.

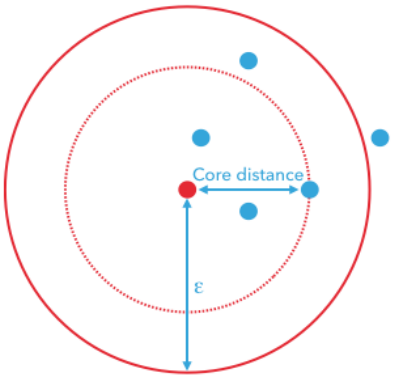
4.2 OPTICS

OPTICS (Ordering Points to Identify the Clustering Structure) byl představen čtveřicí autorů: Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander. Jeho základní myšlenka je stejná jako u DBSCANU, shlukovat body, které jsou blízko sebe. Na rozdíl od DBSCANU se snaží řešit problém s hledáním smysluplných shluků v datech s různou hustotou bodů. Proto data nejdříve setřídí do lineární posloupnosti, kde spolu sousedí vždy nejbližší body. Dále ke každému bodu uloží pomocné atributy, které používá během třídění i následného vyhodnocování.[11],[12]

- **Core distance:** (jádrová vzdálenost) Podobně jako u DBSCANU algoritmus hledá jádrové body. Pro tyto body je vypočtena core distance.

$$CoreDistance_{\epsilon,MinPts}(P) = \begin{cases} NULL & N(P,\epsilon) < MinPts \\ d(P, P_{MinPts}) \end{cases} \tag{4.1}$$

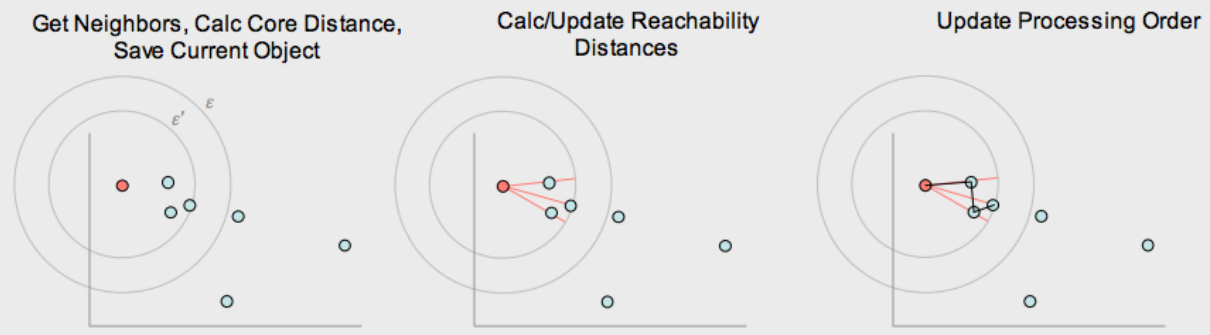
Kde N(P, ϵ) je počet sousedních bodů ve vzdálenosti ϵ a P_{MinPts} je MinPts-tý nejvzdálenější bod od zkoumaného bodu P.



Obrázek 4.2: Core distance pro MinPts = 3

- **Reachability distance:** (dosažitelná vzdálenost) je vzdálenost dvou bodů i a j daná vztahem:

$$reachabilityDistance = max(coreDistance(i), d(i,j)) \tag{4.2}$$



Obrázek 4.3: Seřazení bodů podle reachability distance (převzato z [12]).

Seřazení bodů podle dosažitelné vzdálenost můžeme rozdělit do tří hlavních kroků, které jsou patrné z obrázku 4.3.

1. V prvním kroku se pro zkoumaný bod P, naleznou okolní body ve vzdálenosti ϵ. Dále je pro bod P vypočtena jádrová vzdálenost ϵ'.

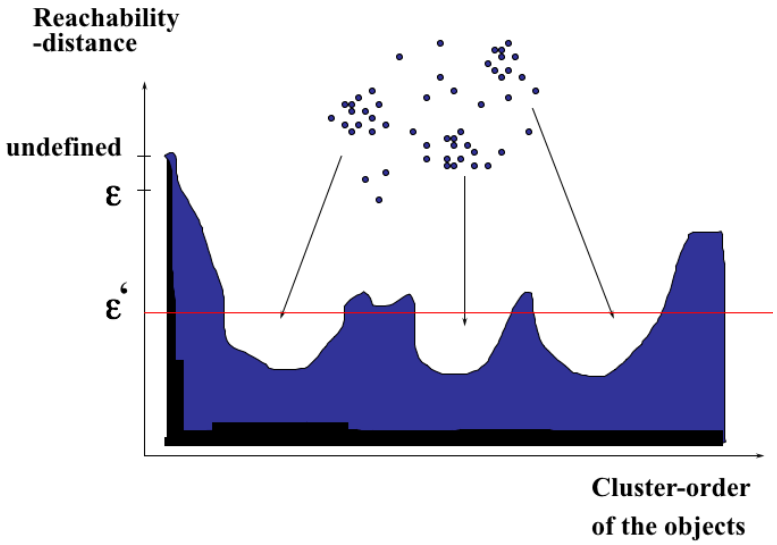
2. V druhém kroku je vypočtena reachability distance, mezi bodem P a jeho okolními body ve vzdálenosti ϵ'.

3. V posledním kroku jsou body seřazeny do pole (orderSeeds) podle nejmenší reachability distance. Bod je označen jako zpracovaný a jeho stav se již nemůže měnit.

   Následně se pokračuje prvním bodem z orderSeeds, tedy tím s nejmenší reachability distance.

Pole orderSeeds slouží jako zásobních okolních bodů. Z tohoto pole je v každé iteraci odebrán jeden bod, který je následně přidán. Do pole jsou postupně přidávány okolní body právě zkoumaného bodu P. Pokud je pole orderSeeds prázdné pokračuje algoritmus prvním nezpracovaným bodem v datech.

Na obrázku 4.4 je graf znázorňující seřazené pole, na ose x jsou body a na ose y je reachability distance. Shluky jsou zde reprezentovány jako prohlubně a jsou hledány pomocí parametru ϵ'. Pomocí tohoto atributu můžeme ovlivnit výsledný počet shluků a jejich velikost. Přitom platí stejné pravidlo jako u DBSCANU, shluk musí mít minimálně MinPts bodů.



Obrázek 4.4: OPTICS, 2D graf seřazeného pole bodů s reachability distance, prohlubně pod ϵ' značí shluky (převzato z 7).

4.3 DENCLUE

DECLUE (DENsity-based CLUstEring) je algoritmus automatické klasifikace založený na hustě. Na rozdíl od jiných algoritmů na bázi hustoty (DBSCAN, OPTICS), je založen na matematickém základu. Tím je statická metoda KDE (Kernel Density Estimation - jádrový odhad hustoty).

Statická metoda KDE popisuje vliv okolních objektů na zkoumaný objekt. Pomocí jádrové funkce vlivu KDE počítá vliv jednotlivých bodů a určuje tak hustotu v okolí zkoumaného objektu. Jádrová funkce určuje jak velký vliv má jeden objekt na druhý v závislosti na jejich vzdálenosti.

- **Jádrová funkce vlivu** popisuje vliv bodu x na bod y, jedná se o vzdálenostní funkci, která musí být reflexivní a symetrická. Příklady takovýchto funkcí jsou:
  - Obdélníková

$$f_{square}(x,y) = \begin{cases} 0 & d(x,y) > \sigma \\ 1 & d(x,y) \leq \sigma \end{cases} \tag{4.3}$$

  - Gaussovská

$$f_{Gauss}(x,y) = e^{\frac{d(x,y)^2}{2\sigma^2}} \tag{4.4}$$

- **Funkce hustoty** bodu x je definována jakou součet všech vlivů bodů z databáze na bod x.

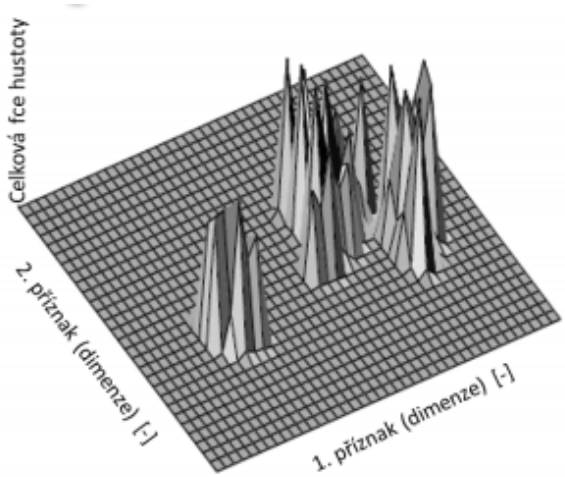
- **Atraktor na základě hustoty** je bod x reprezentující lokální maximum. Okolní body jsou přitahovány tímto atraktorem pokud existuje posloupnost bodů y_0, ..., y_n, kde $y_0 = y$ a $y_k = x$. Dále platí, že pro každé následující y_{i+1} v takové posloupnosti je funkce hustoty vyšší než y_i .

Algoritmus DENCLUE pracuje se dvěma vstupními parametry ϵ a σ, jejichž nastavení zásadně ovlivňuje tvar a velikost nalezených shluků pomocí tohoto algoritmu.

4.3.1 Parametr σ

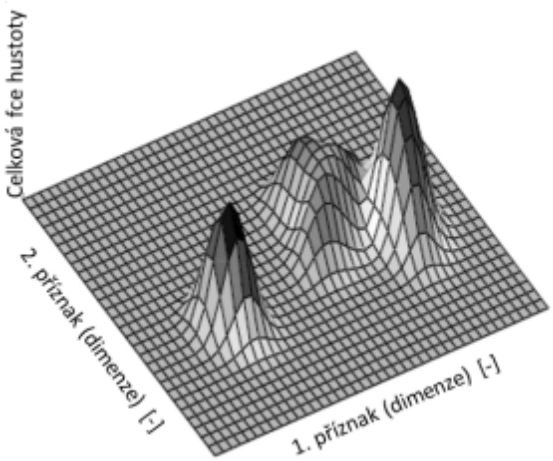
Je koeficient hladkosti a udává jak veliký vliv bude mít objekt na své okolí. Se zvýšením parametru σ působí objekt na vzdálenější objekty, a zároveň má vyšší vliv na bližší objekty. Díky tomu vzniká méně lokálních maxim, což znamená menší počet nalezených shluků. Při snižování parametru σ naopak přibývá lokálních maxim, jejich maxima mohou být příliš nízká. Správné nastavení parametru σ je tedy pro DENCLUE klíčové. Jeho vliv na celkovou funkci hustoty je vidět na obrázku 5.5 a 5.6

Příklad celkové hustoty pro parametr $\sigma = 0.2$



Obrázek 4.5: Na ose x a y jsou bezrozměrné příznaky, na ose z celková funkce hustoty. Parametr σ je poměrně nízký, a tak vzniká více lokálních maxim s vyšší hodnotou těchto maxim. (převzato z [13], upraveno).

Příklad celkové hustoty pro parametr $\sigma = 0.6$

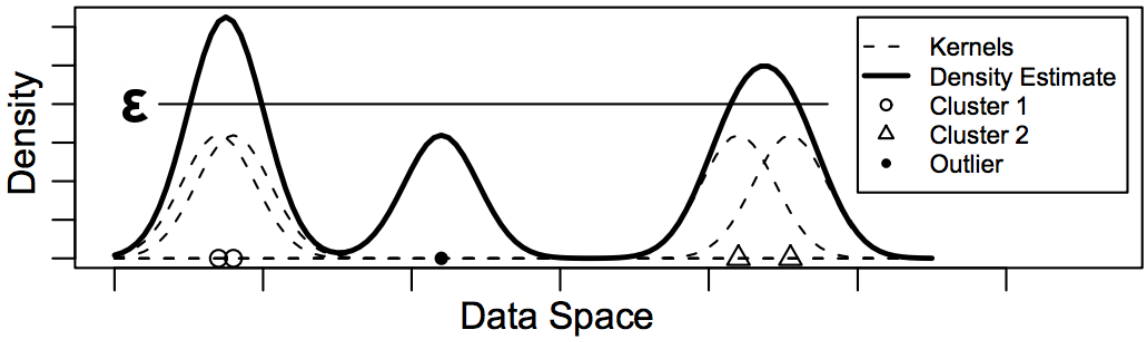


Obrázek 4.6: Na ose x a y jsou bezrozměrné příznaky, na ose z celková funkce hustoty. Parametr σ je výrazně vyšší než na obrázku 5.5. Vliv bodu na své okolí je tedy nižší, ale jeho vliv působí ve větší vzdálenosti, proto vzniká méně lokálních maxim. (převzato z [13], upraveno).

4.3.2 Parametr ϵ

Lze definovat jako koeficient šumu, vztahuje se k celkové funkci hustoty. Udává při jaké velikosti funkce hustoty lze lokální maximum považovat za shluk. Pokud je tedy lokální maximum nižší než ϵ jsou objekty patřící lokálnímu maximu považovány za šum. Jestliže nastavíme parametr ϵ na nulu, může každý samostatný objekt tvořit shluk. Nastavení parametru ϵ udává kolik shluků algoritmus DENCLUE nalezne, viz obrázek 5.7.

Příklad vlivu parametru ϵ na výsledné vyhodnocení shluků.



Obrázek 4.7: Pro objekty (Cluster 1, Cluster 2 a Outiler) je zobrazena funkce vlivu v prostoru (Kernels). Celková funkce vlivu (Density Estimate) je součet funkcí vlivů všech bodů. Z celkové funkce vlivu jsou hledány lokální maxima, pokud je lokální maximum vyšší než zadaný parametr ϵ vznikne shluk. Pro zvolený parametr ϵ jsou nalezeny dva shluky - Cluester 1 a Cluster 2 (převzato z [13], upraveno).

4.3.3 Průběh algoritmu DENCLUE

Průběh algoritmu DENCLUE je možné rozdělit do dvou kroků.

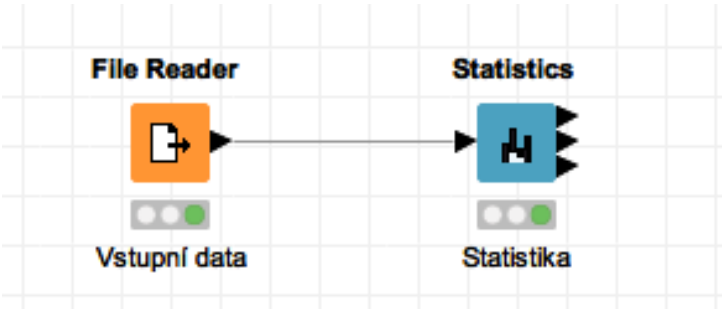
1. **Předshlukovací krok**: ze vstupních dat je sestavena mapa z d-rozměrných hyperkostek. Každá taková hyperkostka obsahuje pouze relativní vzorek dat. Tato mapa slouží k urychlení výpočtu funkce hustoty, při kterém je potřeba efektivně přistupovat k blízce sousedícím bodům.

2. **Shlukovací krok**: jsou hledány atraktory a k nim přitahované body za použití mapy vytvořené v prvním kroku. Každý atraktor a jím přitahované body značí nalezený shluk.

5 Analýza dat

Před implementací analytického nástroje bylo nutné data nejprve analyzovat. Nástroj je v podstatě šitý na míru dostupným datům, ale měl by počítat s různými vstupy. Pro základní analýzu dat jsem použil open source program KNIME, především díky dostupnosti. Nabízeli se i jiné lepší varianty jako IBm SPSS Modeler, ovšem tento program je placený a zkušební verze je pouze čtrnáctidenní.

KNIME má k dispozici rozsáhlou knihovnu funkcí, které lze využít jak pro transformaci dat, tak i pro jejich analýzu. Základní uzel File Reader slouží pro načtení vstupního csv souboru, z tohoto uzlu můžeme navíc zjistit základní strukturu souboru. Vstupní soubor obsahuje 50 sloupců (Příloha A), kde většina nabývá textových hodnot. Některé nabývají samozdřejmě číselných hodnot jako GPS pozice, hustota dopravy, počet zraněných osob, atd.



Obrázek 5.1: Analýza vstupního souboru v KNIME.

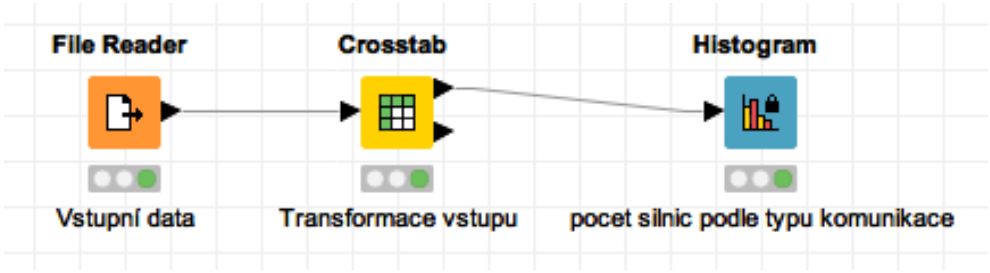
Zajímavější je uzel Statistic (Obrázek 5.1), pro který je nutné definovat zkoumané sloupce. Pro zkoumání jsem vybral pouze některé zajímavé atributy jako je: den nehody, druh pozemní komunikace, zavinění nehody a alkohol u viníka nehody. Na obrázku 5.2 jsou vidět jednotlivé atributy, jakých hodnot nabývají a jaká je jejich četnost zastoupení.

Den	Count...	Druh pozemní komunikace	Count...	Zavinění nehody	Count...	Alkohol u viníka nehody	Count...
pátek	125905	komunikace místní	228256	řidičem motorového vozidla	632914	ne	466537
pondělí	115258	silnice 1.třídy	162338	lesní zvěří, domácím zvířectvem	64102	nezjišťováno	226135
čtvrtek	109730	silnice 2.třídy	146238	řidičem nemotorového vozidla	22034	ano, obsah alkoholu v krvi do 0,99‰ (2)	18572
středa	109271	silnice 3.třídy	117091	chodcem	8910	ano, obsah alkoholu v krvi 1,5‰ a více	14684
úterý	106637	dálnice	29318	technickou závadou vozidla	4586	ano, obsah alkoholu v krvi od 1,0‰ do 1,5‰	3735
sobota	94776	komunikace účelová - ostatní (parkoviště apod.)	28229	jiné zavinění	3635	ano - obsah alkoholu v krvi 1‰ a více (2)	3011
neděle	78474	komunikace sledovaná (ve vybraných městech)	18271	závadou komunikace	2783	ano, obsah alkoholu v krvi do 0,99‰	1900
?	?	uzel (křižovatka sledovaná ve vybraných městech)	7587	jiným účastníkem silničního provozu	1087	ano, obsah alkoholu v krvi od 0,5‰ do 0,8‰	1570
?	?	komunikace účelová - polní a lesní cesty atd.	2723	?	?	ano - obsah alkoholu v krvi 1‰ a více	1462
?	?	?	?	?	?	pod vlivem drog	1153
?	?	?	?	?	?	ano, obsah alkoholu v krvi od 0,8‰ do 1,0‰	1108
?	?	?	?	?	?	pod vlivem drog a alkoholu	184

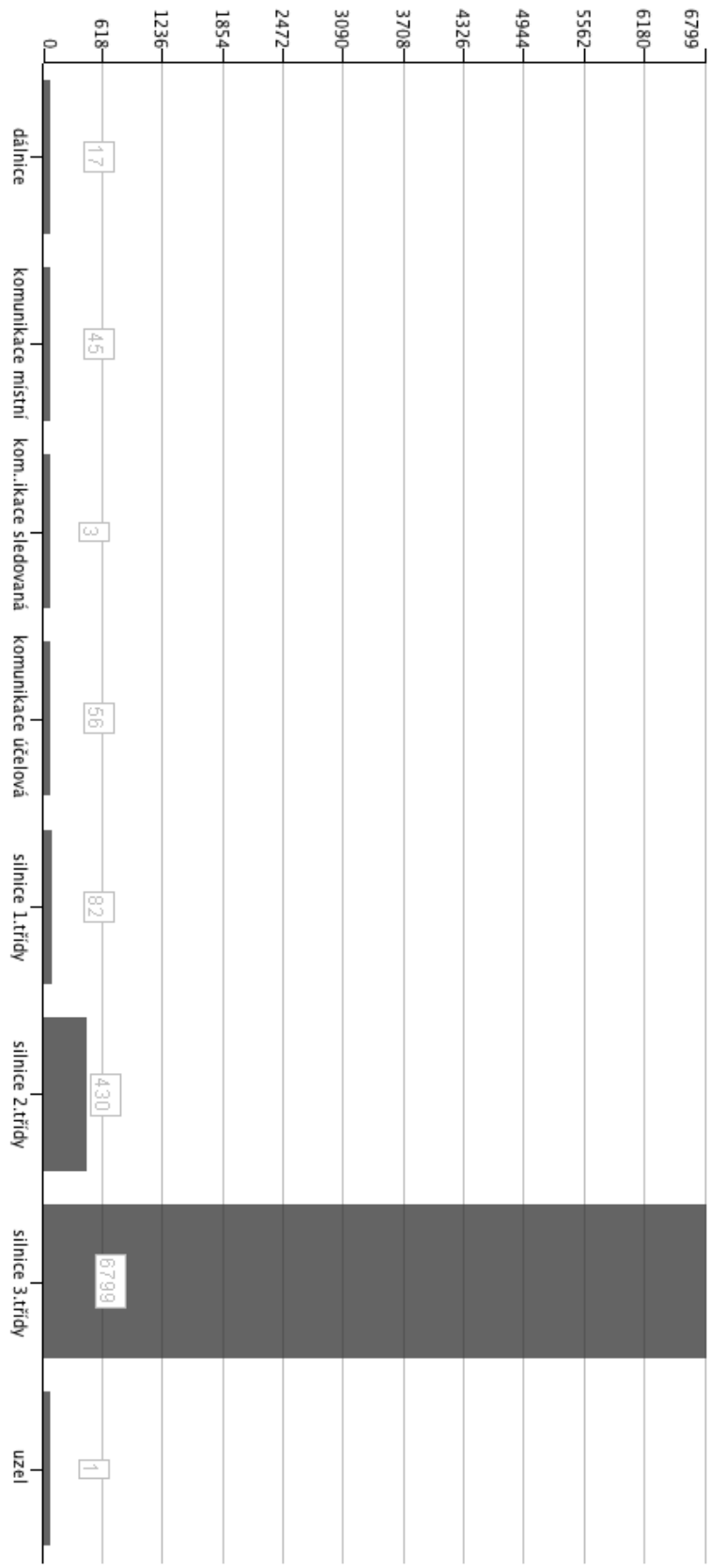
Tabulka 5.1: Vybrané sloupce analyzované pomocí uzlu Statistics.

Pokud se podíváme na den nehody, není zde nic překvapivého - nejvíce nehod se stává v pátek. Další prvenství v počtu dopravních nehod má (Zavinění nehody) řidič dopravního vozidla. Poměrně alarmující je atribut alkohol u viníka nehody, kde viníkem nehody byla v 47 379 případech osoba pod vlivem alkoholu nebo drog. Je to téměř 10% nehod při kterých byla provedena dechová zkouška nebo test na omamné látky.

Pokud se podíváme na atribut Druh pozemní komunikace, nejvíce nehod se stalo na místních komunikacích. Tak jsou běžně označovány parkovišti, menší cesty v městech a na vesnicích. Jak ale můžeme zjistit, na kterém typu komunikace se stává nejvíce nehod? Co do počtu jasně vedou místní komunikace, ale ty jsou také nejčastější a většina nehod na těchto komunikacích není příliš vážná. Pomocí KNIME lze jednoduše zjistit kolik silnic určitého type je obsaženo v datech pomocí atributu Číslo pozemní komunikace, viz obrázek 5.3 a obrázek 5.4.

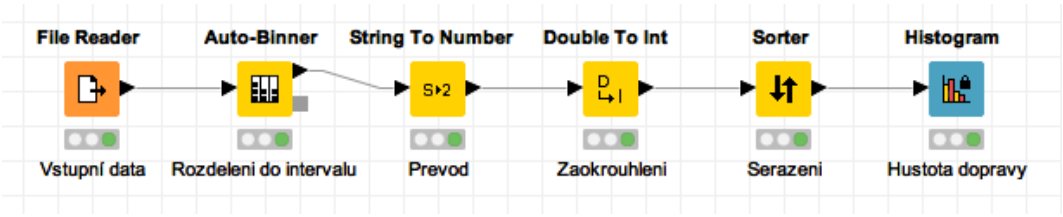


Obrázek 5.2: Transformace dat pomocí KNIME a vytvořený histogram četností čísla pozemní komunikace pro typ komunikace.

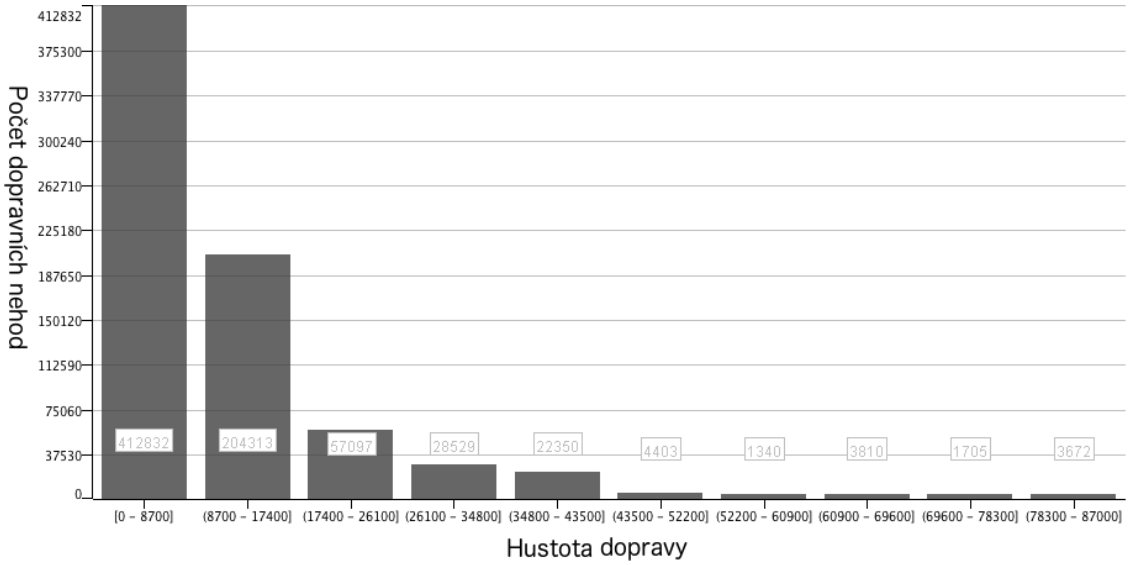


Obrázek 5.3: Histogram četností čísel komunikace podle druhu pozemní komunikace.

Četnost komunikací podle čísla komunikace není úplně směrodatná, jelikož data neobsahují údaje o délce komunikace, nebo o úseku komunikace. Je ale patrné, že nejvíce komunikací je silnic 3. třídy, ovšem počet dopravních nehod je vyšší na méně častých silnicích, jako jsou silnice 1. třídy. Tyto komunikace jsou frekventovanější a je zde větší riziko dopravní nehody.



Obrázek 5.4: Vytvoření histogramu zobrazující počet nehod v závislosti na hustotě dopravy v KNIME.



Obrázek 5.5: Histogram zobrazující počet nehod v závislosti na hustotě dopravy.

Svoji úlohu může hrát i hustota dopravy v místě nehody, její vliv na počet nehod je vidět na obrázku 5.5. Většina nehod se stala na málo frekventovaných komunikacích, kterých je ovšem nejvíce. Jak je vidět v příloze B, na silnicích 3. třídy je většina nehod na komunikacích s hustotou menší než 5900 aut za den. Pokud to porovnáme s komunikací 1. třídy, viz příloha C, větší počet nehod je na frekventovanějších komunikacích. Je tedy zřejmé, že ač je silnic 3. třídy více než 1. třídy, více nehod se stalo na méně častém druhu komunikace, ovšem s vyšší hustotou dopravy. Proto by bylo vhodné hustotu dopravy zohledňovat v analytické nástroji.

6 Implementace

Tato část ze zabývá návrhem a implementací aplikace pro hledání shluků dopravních nehod a následné analýze těchto shluků. Jedním z klíčových požadavků na aplikaci je možnost zpracovat jakýkoliv vstupní soubor s validními GPS pozicemi. Důraz je proto kladen na dynamické ukládání a filtrování vstupních dat. Jsou zde popsány tři vybrané algoritmy určené ke shlukování a metody, které výrazně zrychlují práci s daty v řádech statisíců záznamů. Každý nalezený shluk je možné analyzovat pomocí algoritmu Apriori, díky kterému je možné najít společné příčiny vzniku dopravních nehod.

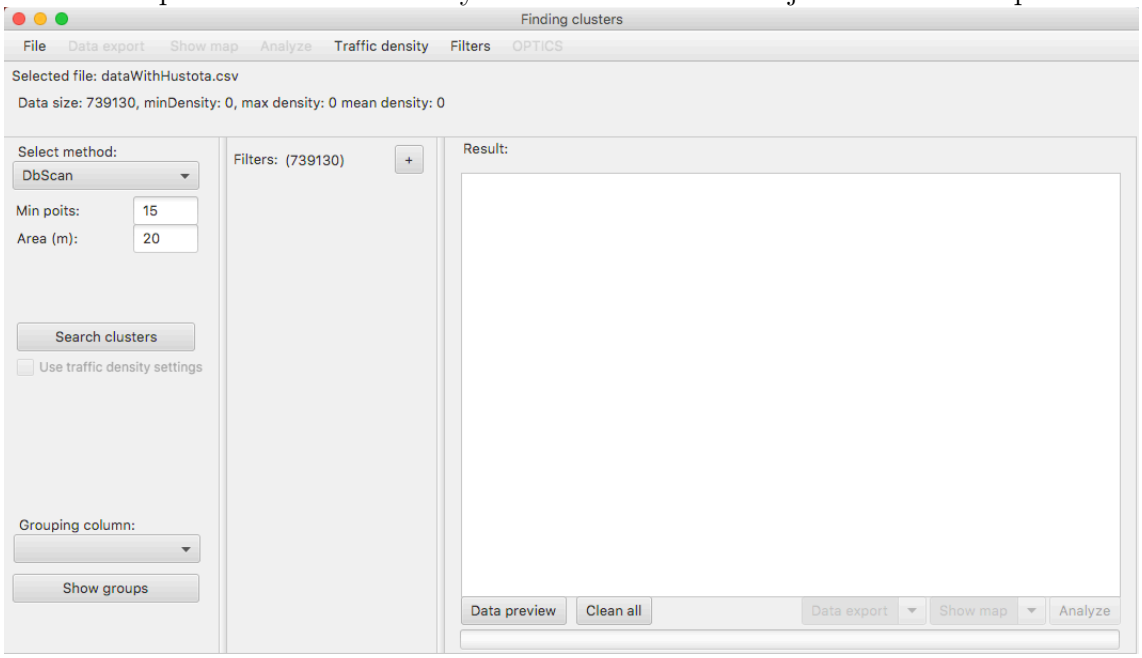
6.1 Použité technologie

Celá aplikace je naprogramována v jazyce Java ve verzi 8. Java 8 s sebou mimo jiné přinesla prvky funkcionálního programování, především lambda funkce, které jsou často použity pro jejich snadnou implementaci a lepší čitelnost kódu. Pro vývoj grafického prostředí je použita knihovna JavaFX 2.0 a nástroj na návrh grafického prostředí Java FX scene builder. Velikou výhodou JavaFX je návrh okenní aplikace definované pomocí FXML(nástavba XML), tato definice je mnohem komfortnější, než je tomu u grafických knihoven AWT a Swing, které používají takzvaný LayoutManager. Další výhodou je možnost pro jednotlivé komponenty definovat vlastní CSS styly.

6.2 GUI

Na obrázku 6.1 je hlavní okno aplikace, v jeho záhlaví se nachází menu, kde lze provádět většinu operací, jako je výběr souboru, export dat, analýza výsledků, atd. Pod záhlavím se nachází informační panel, zde je zobrazen počet vyfiltrovaných bodů, informace o hustotě, nebo chybové hlášení. V levé části lze vybírat meto-

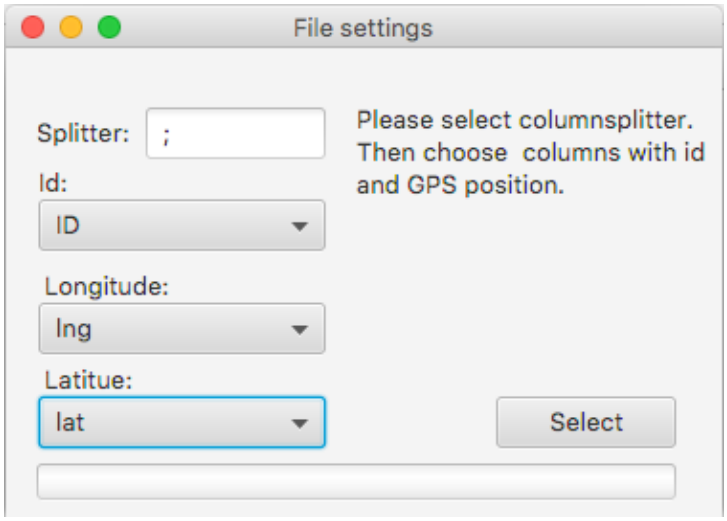
du pro shlukování a nastavit pro ni vstupní parametry. Dále je zde možnost shlukovat objekty podle určitého sloupce, tato možnost je spíše pro usnadnění práce. Po grupování se v části result zobrazí všechny hodnoty, kterých vybraný soubor nabývá, jejich počet a možné akce, které s nimi lze provést. Dále jsou zde uživatelem vybrané filtry na sloupce ze vstupního souboru. Pravá část slouží k zobrazení dat. Může se jednat o přehled vyfiltrovaných dat, zgrupovaných hodnot, ale hlavně pro zobrazení nalezených shluků a možností jak s nimi dále pracovat.



Obrázek 6.1: Hlavní okno grafického rozhraní aplikace, s 981 nalezenými shluky na 739 130 nehodách v ČR.

6.3 Vstupní data

Pro chod celé aplikace je nutné nejdříve načíst csv soubor s nehodami, tento soubor by měl obsahovat hlavičku s popisem sloupců. Vybraný soubor (File -> Open), musí obsahovat nezbytné informace pro shlukovou analýzu: id, latitude, longitude. Celá aplikace je navržena tak, aby bylo možné vložit jakýkoliv soubor obsahující sloupce id, latitude, longitude. Ostatní sloupce slouží pouze k filtrování dat, nebo analýze a již vzniklých shluků (obrázek 6.2).



Obrázek 6.2: Definice vstupních proměnných.

6.3.1 Uložení dat do paměti

Po výběru souboru se data nahrají do paměti. Pro každý řádek v souboru je vytvořen nový objekt Point, který uchovává všechny vstupní parametry a navíc má některé pomocné atributy pro algoritmy. Všechny body jsou uloženy do listu ve třídě DataSource. Tato třída slouží jako databáze, jsou zde uchovávána data, provádí se zde filtrace dat a jsou zde uchovány všechny dostupné hodnoty pro každý sloupec.

6.3.2 Dostupné hodnoty

Jelikož každý sloupec může být použit pro filtrování dat, bylo nezbytné uložit všechny hodnoty sloupců a jaké body těchto hodnot nabývají. S malým souborem dat bylo možné ke každému bodu uchovávat všechny jeho hodnoty a následně podle nich snadno filtrovat. Ovšem po vložení velkého souboru, kde bylo cca 740 000 dopravních nehod a každá nehoda měla asi 50 atributů, byl tento přístup nemožný. Do paměti se nepodařilo nahrát všechna data a aplikace zabírala přes 2 GB paměti a následně vyvolala OutOfMemoryError. Hlavní příčinou byla duplicita dat, protože většina sloupců nabývá pouze několika hodnot. Tyto hodnoty se ale uložily ke každému bodu, což je v takto velikém počtu instancí a sloupců paměťově náročné. Proto jsem navrhl následující strukturu:

```
HashMap < Integer, HashMap < String ,List < Point >>> avaliableValues
```

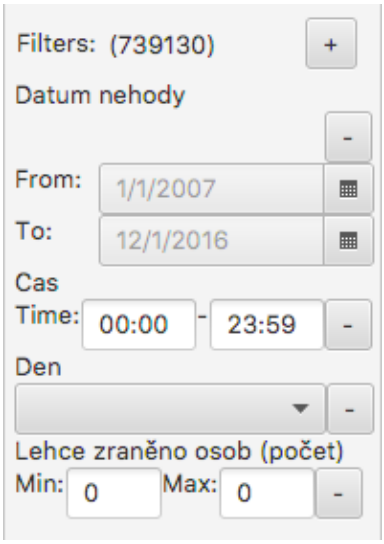
V první HashMap je jako klíč Integer, který reprezentuje pořadí sloupce a jako

objekt uchovává další HashMapu. V té je klíčem jedna hodnota daného sloupce a k ní je přiřazen list bodů, které v daném sloupci nabývají této hodnoty.

Díky této struktuře je v paměti text hodnoty uložen pouze jednou pro jeden sloupec. Přiřazené body jsou pouze reference na již vytvořené objekty. Tato struktura tak zabírá pro stejný vstupní soubor pouze necelých 100MB.

6.3.3 Filtrace dat

Filtrovat data lze pomocí jakéhokoliv sloupce ze vstupního souboru. Aplikace nabízí čtyry druhy filtrů, podle datového typu hodnot ve filtrovaném sloupci. Pro sloupce obsahující pouze datum je k dispozici klasický výběr data v rozmezí od - do. Podobně je tomu tak v případě času, kdy musí být korektně zapsány časy. Pokud sloupec obsahuje pouze číselné hodnoty je uživatel použita filtrace rozsahem. V ostatních případech je možné vybrat *1* až *n* hodnot ze všech hodnot ve vybraném sloupci. Na obrázku 6.3 jsou ukázky těchto filtrů. Filtry lze dynamicky přidávat nebo odebírat.



Obrázek 6.3: Filtr s datovým rozsahem (Datum nehody), rozsahem časů (Cas), textovými hodnotami (Den) a filtr pouze s číselnými hodnotami (Lehce zraněno osob)

Zajímavá je především kombinace více filtrů s textovými hodnotami sloupců. Pokud budeme filtrovat například den a typ komunikace zajímá nás průnik bodů, které mají zvolené hodnoty. Vybrané filtry jsou uloženy do proměnné filter typu HashMap <Integer, Set<String>>, kde pro každý sloupec je set vybraných hodnot.

Při filtraci dat je skrz proměnnou filter iterováno a z avaliableValues (kapitola 6.3.2) jsou do listu vyfiltrvaných bodů vkládány body, které obsahují vybraně texty.

Zajímavá je především metoda filterDuplicatesOnly(), která je volána po každé iteraci sloupce. V ní jsou vybrány pouze duplicitní hodnoty tedy ty, které splňují všechny podmínky filtrů z dosud iterovaných sloupců. Tato metoda je mnohonásobně rychlejší než iterovat skrz všechna data a zjišťovat pro každý bod, jestli splňuje všechny podmínky.

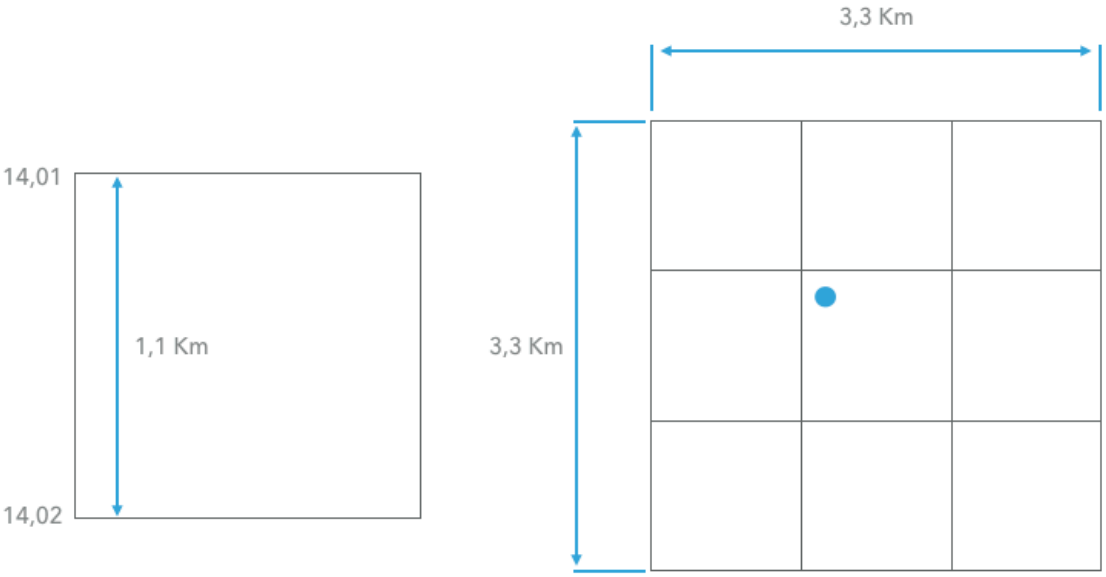
```
private void filterData() {
     for (Entry<Integer, Set<String>> entry : filter.entrySet()) {
        for (String string : entry.getValue()) {
           if (data.size() == 0)
              data = new
                  ArrayList<>(avaliableValues.get(entry.getKey()).get(string));
           else
              data.addAll(avaliableValues.get(entry.getKey()).get(string));
        }
        filterDuplicatesOnly();
     }
  }

private void filterDuplicatesOnly() {
     Set<Point> allItems = new HashSet<>();
     Set<Point> duplicates = data.stream().filter(n ->
         !allItems.add(n)).collect(Collectors.toSet());
     if (!duplicates.isEmpty())
        data = new ArrayList<>(duplicates);
  }
```

Uložení aktuálního nastavení

Nástroj umožňuje uložení aktuálního nastavení všech atributů. Toto nastavení může být kdykoliv obnoveno, nahráním uloženého nastavení. Tyto dvě volby jsou dostupné z menu aplikace. Jedná se spíše o pomocnou funkcionalitu, díky které je nástroj více user-friendly. Lze si tak jednoduše uchovávat zajímavé výsledky pro další zkoumání v budoucnu.

6.4 Rozdělení dat

Jelikož všechny implementované algoritmy mají podobný průběh, iterují skrz všechny body, je možné používat pro všechny stejná vstupní data. Při průchodu jakéhokoliv algoritmu se program dostane do bodu, kdy je nutné nad právě zkoumaným bodem provést nějakou operaci se všemi ostatními body. Pokud ale hledáme shluky, zajímají nás pouze okolní body, které se nachází v blízkosti zkoumaného bodu. Právě výpočet vzdálenosti by byl nejčastější operací, která však v drtivé většině případů rozhodne bod dále nezkoumat. Z tohoto důvodu jsou data už před začátkem algoritmu rozdělena do čtvercové sítě, kde každý bod zapadne do jednoho ze čtverců podle latitude a longitude.



Obrázek 6.4: Bod je podle GPS souřadnic zařazen do jednoho ze čtverců, které jsou rozděleny po setině zeměpisné délky a šířky. Jeden stupeň zeměpisné šířky měří přibližně 111 120 metrů, jedna setina je tudíž cca 1 111 metrů.

Tyto čtverce lze zapsat do následující HashMapy:

```
HashMap<Double, HashMap<Double, LinkedList<Point>>> dataMap
```

Kde první Double je latitude a druhý je longitude. Výpočet do jakého listu bodů bude bod zařazen je prostým zaokrouhlením latitude, případně longitude.

```
double lan = Math.round(point.getLat() * 100.0) / 100.0;
double lon = Math.round(point.getLon() * 100.0) / 100.0;
```

Díky této struktuře se pro zkoumaný bod počítá pouze s body, které jsou ve stejném a všech okolních čtvercích (obrázek 6.4). Díky tomu jsou prováděny operace jen s blízkými body, a to ve vzdálenosti minimálně 1,1 kilometru, což je dostatečná vzdálenost při hledání shluků dopravních nehod.

6.5 Hledání shluků

V rámci aplikace, byly implementovány tři algoritmy DBSCAN, OPTICS a DENCLUE. Na obrázku 6.1 je v levém panelu pole na výběr metody, pod tímto výběrem jsou povinné atributy pro jednotlivé algoritmy. Defaultně obsahují hodnoty, které se mi nejlépe osvědčili při hledání shluků na všech datech. Pro hledání specifických shluků na základě některých filtrů se budou jevit jako optimální jiné hodnoty.

6.5.1 DBSCAN

DBSCAN byl implementován podle pseudokódu z kapitoly 4.1. Tento algoritmus je na první pohled primitivní a nijak nepracuje s hustotou dat. Jeho výsledky jsou přesné především na menších datech, jako je použití filtru na typ komunikace (například silnice 3. třídy).

Vstupní parametry jsou MinPts (minimální počet bodů ve shluku) a area (vzdálenost, ve které se musí MinPts bodů vyskytovat). Kde area je zadávána v metrech, následně se provede přepočet, tak aby se s ní mohlo počítat v GPS souřadnicích. Jedná se v podstatě o převod na setiny stupně zeměpisné šířky.

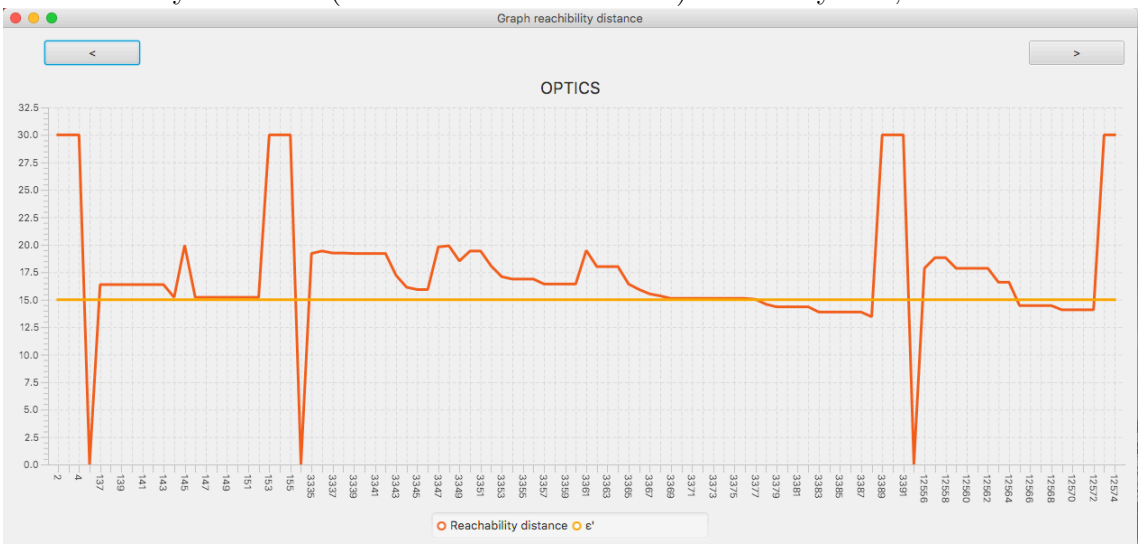
```
area = area / 111120
```

Pomocí takto převedeného čísla lze jednoduše vyhodnocovat, jestli je vzdálenost mezi dvěma body větší nebo menší.

6.5.2 OPTICS

Jak již bylo napsáno v kapitole 4.2 OPTICS vychází ze stejné myšlenky jako DBSCAN. Na rozdíl od něj tvoří seřazenou posloupnost bodů, kde vedle sebe jsou vždy nejbližší body. Toto vyhodnocení probíhá na základě stejných vstupních parametrů jako je u DBSCANu. Rozdíl je v tom, že pokud máme takto seřazenou posloupnost, můžeme pomocí vstupního parametru ϵ' jednoduše měnit výsled-

né shluky. Parametr ϵ' lze měnit i po výpočtu a tím pouze přepočítat výsledné shluky. Pro vizualizaci a ověření výsledků je v aplikaci možné zobrazit graf s reachability distance (dosažitelnou vzdáleností) a zvoleným ϵ', viz obrázek 6.5.



Obrázek 6.5: V grafu je zobrazeno vždy pouze 100 hodnot, kvůli přehlednosti. V horní části se lze libovolně posouvat na další části grafu. Dále jsou v grafu zobrazeny pouze tři po sobě jdoucí body s nekonečnou reachability distance, také z důvodu přehlednosti.

6.5.3 DENCLUE

Při implementaci DENCLUE bylo zásadní vybrat správnou jádrovou funkci. Nabízela se možnost využít čtvercovou funkci, při jejím užití se ale algoritmus chová dost podobně jako DENCLUE. Výpočet hustoty bodu je v podstatě součet všech bodu v nějakém okolí (viz rovnice 4.3). Algoritmus má sice jiný průběh i výsledky, ale nelze se spolehnout na základní výhodu DENCLUE, a to hledání shluků v datech s různou hustotou.

Proto jsem jako jádrovou funkci zvolil Gaussovu funkci vlivu:

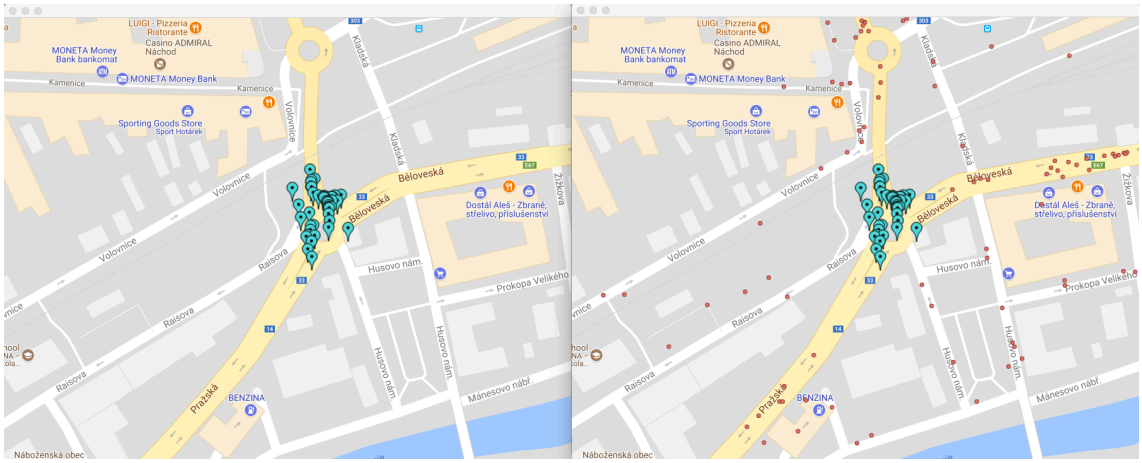
$$f_{Gauss}(x, y) = e^{\frac{d(x,y)^2}{2\sigma^2}} \tag{6.1}$$

Díky této funkci jsou výsledné shluky na datech s různou hustotou přesnější. Hledání lokálních maxim je velice citlivé na parametr σ, jeho nalezení je klíčové pro celý algoritmus. Pro různé filtry se může výrazně lišit, algoritmus je proto náchylný na špatně zvolené vstupní parametry σ a ϵ.

6.6 Zobrazení výsledků

Po hledání shluku podle jakéhokoliv algoritmu se v pravé části zobrazí všechny zobrazené shluky (obrázek 6.1). Jeden řádek reprezentuje jeden nalezený shluk, k němu i základní informaci, kolik obsahuje dopravních nehod. Každý shluk lze zobrazit v mapě jednotlivě, nebo ho analyzovat (kapitola 6.5). Pro podrobnější informace o všech shlucích lze exportovat výsledek do souboru csv. Každý řádek v souboru reprezentuje jednu dopravní nehodu, ke každé nehodě je také číslo shluku, do kterého patří. Lze exportovat soubor pouze s nalezenými shluky, nebo exportovat soubor i s šumem, kde je ve sloupci s číslem shluku šum označen jako -1.

Stejný způsobem lze zobrazit mapu s body, kde jsou jednotlivé shluky barevně odlišeny a šum je značen jako malý bod na mapě.



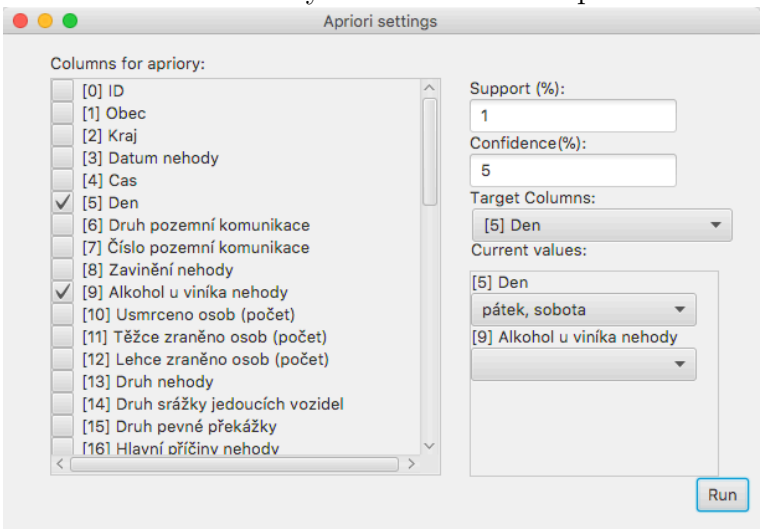
Obrázek 6.6: Jeden shluk nehod ve městě Náchod, MinPts = 15, Area = 20m. Na levém obrázku jsou pouze shluk, na pravém je zobrazen i šum.

6.7 Analýza shluků pomocí Apriori

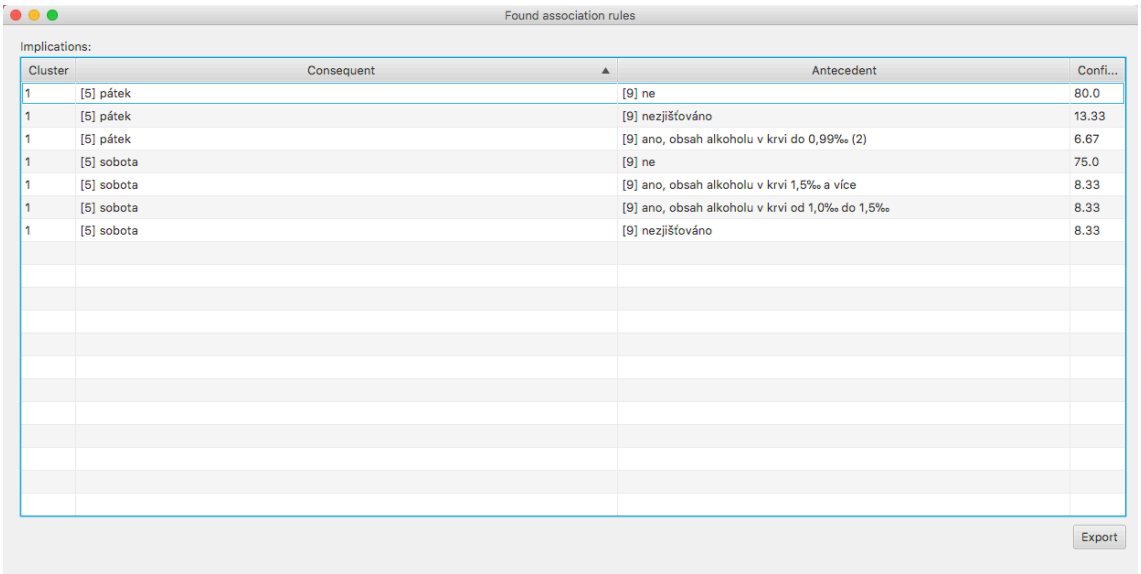
Každý nalezený shluk v aplikaci lze analyzovat. Při analýze jsou hledány podobné vlastnosti nehod v jednotlivých shlucích. V těchto shlucích hledáme asociační pravidla podle algoritmu Apriori. Jako vstup do tohoto algoritmu jsou sloupce, na kterých chceme hledat asociační pravidla a vstupní parametry support a confidence (kapitola 3).

Nástroj umožňuje vybrat sloupce, které nás zajímají jako předpoklad. Pro vybrané předpoklady můžeme vybrat konkrétní hodnoty. To-

ho lze využít například v situaci kdy, chceme hledat v jakých dnech se nejčastěji bourá v podnapilém stavu. Nastavení pro algoritmus Apriori je vidět na obrázku 6.7 a výsledná asociační pravidla na obrázku 6.8.



Obrázek 6.7: Výběr sloupců ke zkoumání (Den a Alkohol u viníka nehody), kde nás zajímá Den jako předpoklad (Target), kenkrétně dvě hodnoty pátek a sobota (Current values). Support a confidence jsou poměrně nízké, jelikož jsou hledány spíše extrémní hodnoty s nízkou pravděpodobností výskytu.



Obrázek 6.8: Vypočtená asociační pravidle podle nastavení z obrázku 6.7. Z asociačních pravidel je zřejmé, že nehody způsobené alkoholem se častěji stávají v sobotu, a to v 16,66% nehod.

6.8 Zlepšení pomocí hustoty

V průběhu psaní této práce, jsem se často dostal do bodu kdy pro různé typy silnic nebo jejich úseky nebyly nalezeny některé shluky. Například na některých silnicích prvních tříd bylo nalezeno výrazně více, či méně shluků dopravních nehod. Stejný problém nastával i u jedné silnice, kde na různých částech byl vyšší, nebo nižší počet dopravních nehod.

Po bližším zkoumání bylo zřejmé, že části silnic procházející skrz města obsahují výrazně více dopravních nehod, než je tomu na venkově. Důvod je zřejmý, hustota dopravy ve městě je výrazně vyšší a tudíž je větší pravděpodobnost vzniku dopravní nehody. Z tohoto důvodu jsem se pokusil optimalizovat algoritmy DBSCAN a OPTICS, tak aby dokázali reagovat na hustotu dopravy.

6.8.1 Zisk dat s hustotou dopravy

Poslední měření hustoty dopravy proběhlo v roce 2010 a jeho výsledky jsou volně dostupné na webových stránkách geoportal.rsd.cz[17]. Rozhodl jsem se tato data stáhnout ke všem dopravním nehodám, které byli k dispozici.

Stahování dat pro jednu nehodu probíhalo pomocí jednoduchého Bash skriptu:

```
curl -H "Accept: application/json"
     -H "Content-Type: application/json"

     -X GET 'https://geoportal.rsd.cz/arcgis/rest/services/ScitaniDopravy/
      MapServer/0/query?f=json&where=1%3D1&returnGeometry=true&spatialRel=
      esriSpatialRelIntersects&geometry=%7B%22xmin%22%3A'$xmin'%2C%22ymin%
      22%3A'$ymin'%2C%22xmax%22%3A'$xmax'%2C%22ymax%22%3A'$ymax'%
      2C%22spatialReference%22%3A%7B%22wkid%22%3A102067%2C%
      22latestWkid%22%3A5514%7D%7D&geometryType=esriGeometryEnvelope
      &inSR=102067&outFields=*&outSR=102067' >> $data
```

Tento skript stáhne pro jednu dopravní nehodu všechny dopravní úseky v blízkém okolí definovaném rozsahem $<$ xmin; xmax $>$ a $<$ ymin; ymax $>$. Pokud v odpovědi není nalezen žádný dopravní úsek, skript provede přepočet rozsahů a pokusí o další dotaz. Některé úseky, především v centrech měst a na místních komunikacích nebyli zařazeny do měření hustoty dopravy, a proto mají hustotu 0. Skript je odolný i vůči situaci, kdy je dopravní nehoda na rozmezí dvou nebo více sčítaných úseků. V

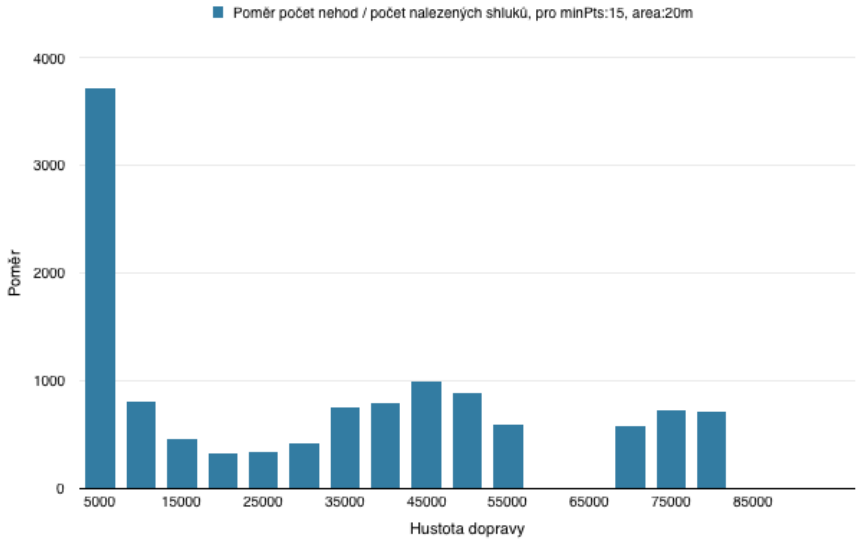
takovém případě je dopravní nehodě přidělena nejvyšší hustota z oblasti.

6.8.2 Analýza shluků podle hustoty dopravy

Pro zkoumání závislosti hustoty dopravy a počtem nalezených shluků jsem se rozhodl hledat shluky vždy v rozmezích po 5 000 hustoty dopravy. Přičemž nejnižší nalezená hustota je 17 aut za den. První rozmezí bylo tedy $< 17; 5\ 000 >$, kvůli odfiltrování nehod s nenalezenou hustotou dopravy. Jelikož každý interval obsahuje jiné množství dopravních nehod zaznamenal jsem poměr:

$$\text{poměr} = \frac{\text{počet nehod}}{\text{počet nalezených shluků}} \tag{6.2}$$

Tento poměr je zobrazen v grafu na obrázku 6.9.



Obrázek 6.9: Poměr počet nehod / počet nalezených shluků, pro minPts = 15, area = 20m, za použití shlukovacího algoritmu DBSCAN.

Z grafu je patrné, že nejlepší poměr je v rozmezí $< 15\ 000; 20\ 000 >$. Naopak v krajních rozmezích je tento poměr víc než třikrát horší. Dalším důležitým faktorem je, že počet nehod v intervalu $< 50\ 000; 85\ 000 >$ se pohybuje okolo 1 000 nehod na jeden pětitisícový interval. Poměry v tomto rozmezí mohou být výrazně zkresleny. V posledním intervalu je dostatečný počet dopravních nehod, takže by nemělo dojít ke zkreslení výsledku.

Toto rozložení je logické, na silnicích s nižší hustotou provozu se shluky hledají hůře, stejně tak na dálnicích kde je naopak vysoká hustota provozu. Pro tyto ob-

lasti je ideální upravovat vstupní parametry, tedy snížit parametr minPts a naopak rozšířit oblast area.

6.8.3 Úprava vstupních parametrů

Filtrovat neustále hodnoty pro jakékoliv hledání je dost nepraktické. Lepší metodou je dynamicky upravovat vstupní proměnné podle hustoty provozu, pro právě zkoumaný bod. Tím by se měl poměr ve všech intervalech narovnat na podobnou hodnotu.

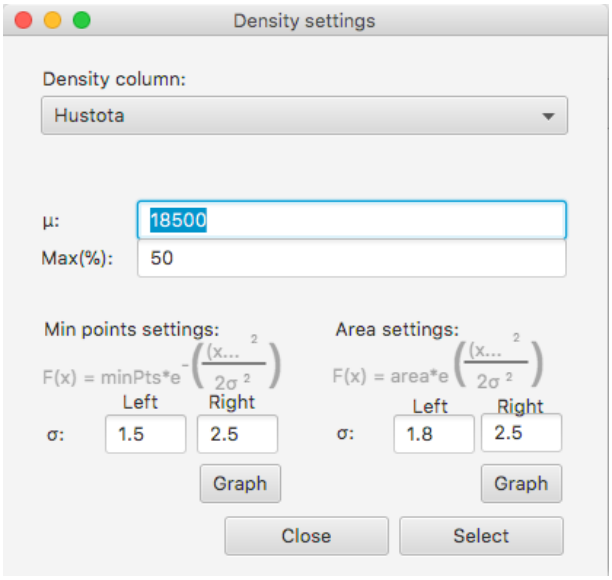
Na obrázku 6.9 je nejnižší poměr v intervalu < 15 000; 20 000 >. Nárůst poměru v levé části je mnohem strmější než v pravé části. Proto bylo nutné nalézt přesný střed, kde je poměr nejnižší. Tento střed byl nalezen rozdělením intervalu < 15 000; 20 000 > na menší intervaly po jednom tisíci. Zde byl nejnižší poměr v intervalu < 18 000; 19 000 >. Lze tedy předpokládat že střed je přibližně okolo hustoty dopravy 18 500.

Od této hustoty dopravy je potřeba dynamicky upravovat vstupní parametry. Obě poloviny připomínají kvadratickou funkci, nebo také obrácenou Gaussovu křivku. Právě Gaussova křivka byla pro přepočet vstupních parametr použita. Pro minPts byla otočena tak aby od středu měla sestupnou tendenci, naopak u area má vzrůstající tendenci.

$$minPts = minPts * e^{-(\frac{(x-\mu^2)}{2*\sigma^2})} \tag{6.3}$$

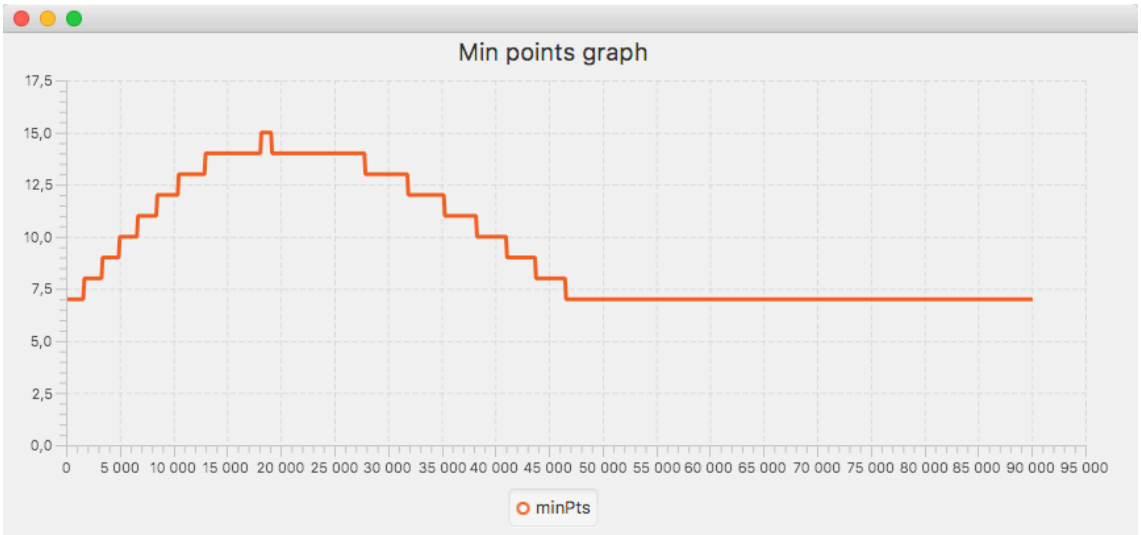
$$area = area * e^{(\frac{(x-\mu^2)}{2*\sigma^2})} \tag{6.4}$$

Kde μ je střední hodnota, tedy 18 500. Parametr σ udává strmost nárůstu nebo poklesu. Nastavení přepočtu lze v aplikaci ovládat pomocí položky traffic density v menu, viz obrázek 6.10.

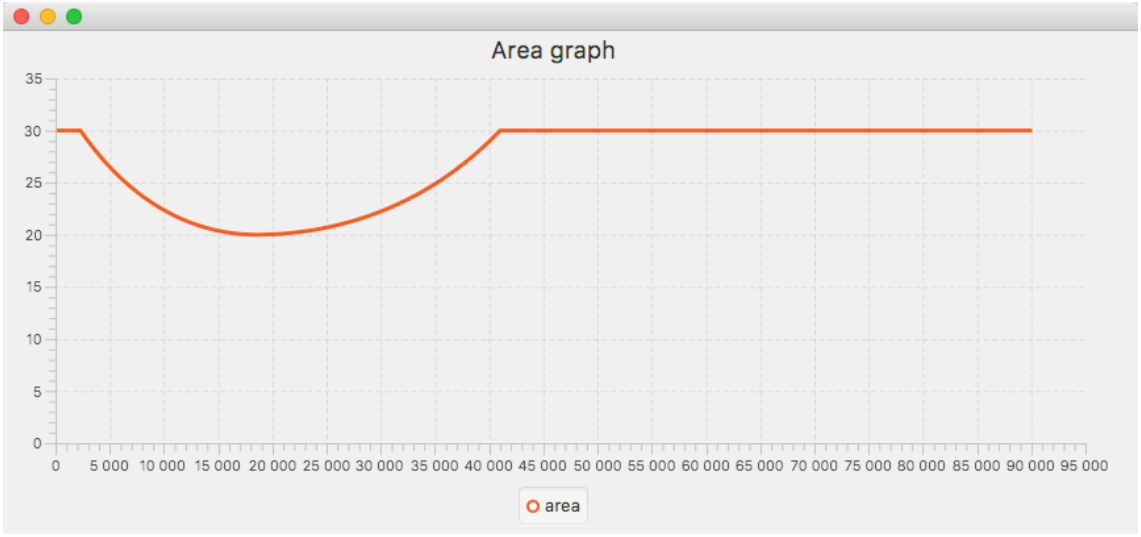


Obrázek 6.10: Nastavení parametru hustota pro zlepšení vyhledávání clusterů.

Je možné zvolit jiný parametr σ pro levou a pravou stranu, tak abychom mohli obě strany nastavit nezávisle na sobě. Dále je zde možnost zadat o kolik procent se může vstupní parametr změnit, jelikož u area by teoreticky mohl vystoupat až k nekonečnu a výsledkem by byl jeden velký shluk. Průběh obou parametrů je zobrazen na obrázcích 6.11 a 6.12.



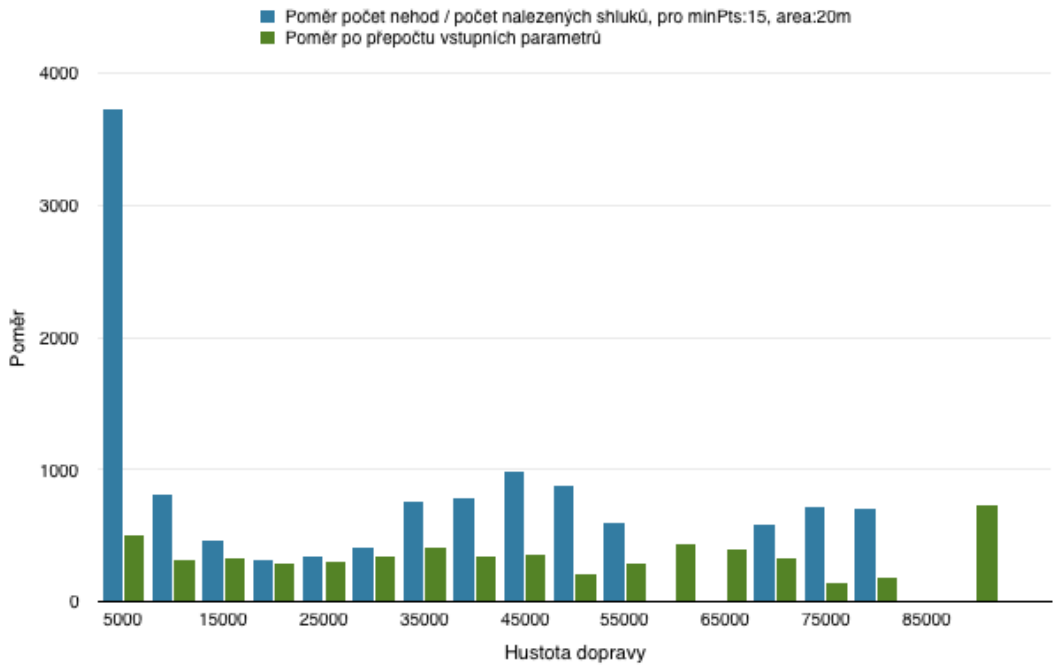
Obrázek 6.11: Průběh parametru minPts v závislosti na hustotě dopravy. MinPts=15, μ=18 500, σ left=1.5 a σ right=2.5 a maximální změnou o 50%. Na grafu jsou vidět skoky, vždy o jeden bod, minPts musí být vždy celé číslo.



Obrázek 6.12: Průběh parametru area v závislosti na hustotě dopravy. area=20, μ=18 500, σ left=1.8 a σ right=2.5 a maximální změnou o 50%.

6.8.4 Porovnání výsledků

Výsledky dosažené pomocí přepočtu vstupních parametrů jsou vidět na první pohled v aplikaci. Například pokud jsou vstupní parametry minPts = 15 a area = 20 metrů, počet nalezených shluků je 961 bez použití přepočtu vstupních parametrů. Jestliže je použit přepočet vstupních parametrů na základě hustoty dopravy podle nastavení z obrázku 6.11 a 6.12 je nalezeno 2068 shluků. Počet nalezených shluků je přibližně dvojnásobný, srovnání výsledků lze vidět na obrázku 6.13.



Obrázek 6.13: Porovnání poměrů počet nehod / počet nalezených shluků, před přepočtem vstupních parametrů (modrá) a po přepočtu (zelená).

Z grafu na obrázku 6.13 je patrné, že poměr nalezených shluků na počet nehod se výrazně narovnal. Tento přístup umožňuje zkoumání velkého množství dopravních nehod v místech s různou hustotou dopravy díky optimalizaci vstupních parametrů pro danou lokalitu. Přepočtené hodnoty vstupních parametrů jsou zaznamenány v exportu csv souboru, takže je možné dohledat jaké parametry byly pro shluky použity.

7 Závěr

Hlavním úkolem bylo vytvořit analytický nástroj pro hledání shluků dopravních nehod. Nástroj je navržen tak, aby mohl přijímat jakákoliv data s GPS pozicemi. Pomocí těchto pozic jsou shluky vyhledávány. Pro jejich hledání byly implementovány tři algoritmy DBSCAN, OPTICS a DENCLUE. Nalezené shluky lze exportovat do csv souboru samostatně nebo se šumem. Aplikace také umožňuje zobrazení v mapě a to i s šumem, což může být u většího počtu dopravních nehod poměrně pomalé, kvůli vykreslování. Z tohoto důvodu lze v mapě zobrazit pouze 50 000 bodů. Nástroj dokáže filtrovat vstupní data pomocí jakéhokoliv atributu ze vstupního souboru, kde podle hodnot ve sloupci lze filtrovat číselné hodnoty, datumy a časy rozsahem, nebo textové hodnoty výběrem.

Pro zkoumání nalezených shluků byl implementován algoritmus Apriori, který hledá asociační pravidla. Zde je možnost vybrat, nad kterými atributy budou asociační pravidla hledána a jaké mají být použity jako předpoklad k pravidlu. Analyzovat lze buď jednotlivé shluky, nebo všechny nalezené současně. Nalezená pravidla lze exportovat do csv souboru.

Dalším dílčím úkolem bylo stažení dat o hustotě dopravy. To bylo provedeno pomocí bash skriptu, který pro každou dopravní nehodu hledal komunikaci s údajem o hustotě dopravy. Zde byl problém především v částech velikých měst a vedlejších silnicích, kde sčítání neprobíhala. Z toho důvodu některé nehody tento údaj nemají. Pro zbylé byla navržena optimalizace. Algoritmy DBSCAN a OPTICS reagují na hustotu dopravy přepočtem jejich vstupních parametrů. Díky tomu je možné nalézt smysluplné shluky v oblastech s různou hustotou dopravy.

Výsledný nástroj byl navržen dynamicky, tak aby dokázal zpracovávat různá vstupní data, nejen ta o dopravních nehodách. Dokáže pracovat i s velkými daty, řádově statisíce záznamů, v čase maximálně několika vteřin.

Možností jak stávající řešení rozšířit je několik. Jednou z nich je pro nalezené shluky vytvořit webovou aplikaci s pokročilým filtrováním shluků. Dalším a mno-

hem zajímavějším způsobem je vytvořit inteligentní navigaci, která pomocí přednastavených atributů bude zobrazovat nebezpečná místa na trase. Navigace by také mohla reagovat na aktuální počasí, jelikož data obsahují také informaci, za jakého počasí k nehodě došlo.

Literatura

[1] ZENDULKA, J., BARTÍK, V., LUKÁŠ, R., RUDOLFOVÁ, I. *Získávání znalostí z databází - studijní opora* [online]. FIT VUT v Brně, 2006 [cit. 2017-09-25] Dostupné z: http://www.fit.vutbr.cz/~bartik/ZZN.pdf

[2] DOUG, Alexander. *Data Mining* [online]. 2017 [cit. 2017-09-25]. Dostupné z: https://www.laits.utexas.edu/~anorman/BUS.FOR/course.mat/Alex/

[3] Smart Vision Europe. *What is the CRISP-DM methodology?* [online]. 2016 [cit. 2017-09-25].
Dostupné z: http://www.sv-europe.com/crisp-dm-methodology/

[4] WIKIPEDIA: *SEMMA* [online]. 2017 [cit. 2017-09-29]. Dostupné z:https://en.wikipedia.org/wiki/SEMMA

[5] BERKA, Petr. *Dobývání znalostí z databází*, Praha: Academia, 2003. 366s. ISBN 80-200-1062-9

[6] sorry.vse.cz. *5.2 Asociační pravidla* [online]. 2009 [cit. 2017-09-25].
Dostupné z: http://sorry.vse.cz/~berka/docs/izi456/kap_5.2.pdf

[7] HAN, J. *Cluster Analysis — presentation slides.* [online]. [cit. 2017-09-25].
Dostupné z: http://www.cs.uiuc.edu/~hanj/bk2/07.ppt

[8] HAN, J., KAMBER, M *Data Mining: Concepts and Techniques. second edition*, Morgan Kaufmann Publishers, 2006, ISBN 1-55860-901-6, 770 s.

[9] CIOS, K. J.; PEDRYCZ, W.; SWINIARSKI, R. W., aj. *Data Mining: A Knowledge Discovery Approach.*, Springer, 2007, ISBN 978-0-387-33333-5, 606 s.

[10] ESTER, M., KRIEGEL, H-P., SANDER, J., XU, X. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* [online]. 1996

[cit. 2017-09-25].
Dostupné z: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

[11] WIKIPEDIA: *OPTICS algorithm — Wikipedia, The Free Encyclopedia* [online]. 2017 [cit. 2017-09-29]. Dostupné z:https://en.wikipedia.org/wiki/OPTICS_algorithm

[12] ANKERST, M., BREUNIG, M.M., KRIEGEL, H-P., SANDER, J. *OPTICS: Ordering Points To Identify the Clustering Structure* [online]. 2004 [cit. 2017-09-25]. Dostupné z: https://pdfs.semanticscholar.org/0157/f142bee7b462897424908cd6c73d84f225cc.pdf

[13] HINNEBURG, A., KEIM, D.A. *A General Approach to Clustering in Large Databases with Noise* [online]. 2003 [cit. 2017-09-25]. Dostupné z: https://link.springer.com/content/pdf/10.1007/s10115-003-0086-9.pdf

[14] Ministerstvo dopravy. *Jednotná dopravní vektorová mapa* [online]. 2017 [cit. 2017-09-25]. Dostupné z: http://www.jdvm.cz/

[15] *KDE BOURÁME* [online]. 2017 [cit. 2017-09-25]. Dostupné z: http://www.kdebourame.cz/cz/

[16] *Geoportál silnic)ční a dálniční sítě ČR* [online]. 2017 [cit. 2017-09-25]. Dostupné z: https://geoportal.rsd.cz/webappbuilder/apps/7/

[17] *Geoporál silniční a dálniční sítě ČR* [online]. 2017 [cit. 2017-09-25]. Dostupné z: http://geoportal.rsd.cz/web

8 Seznam příloh

Příloha A: Seznam všech atributů ve vstupních datech a jejich datové typy

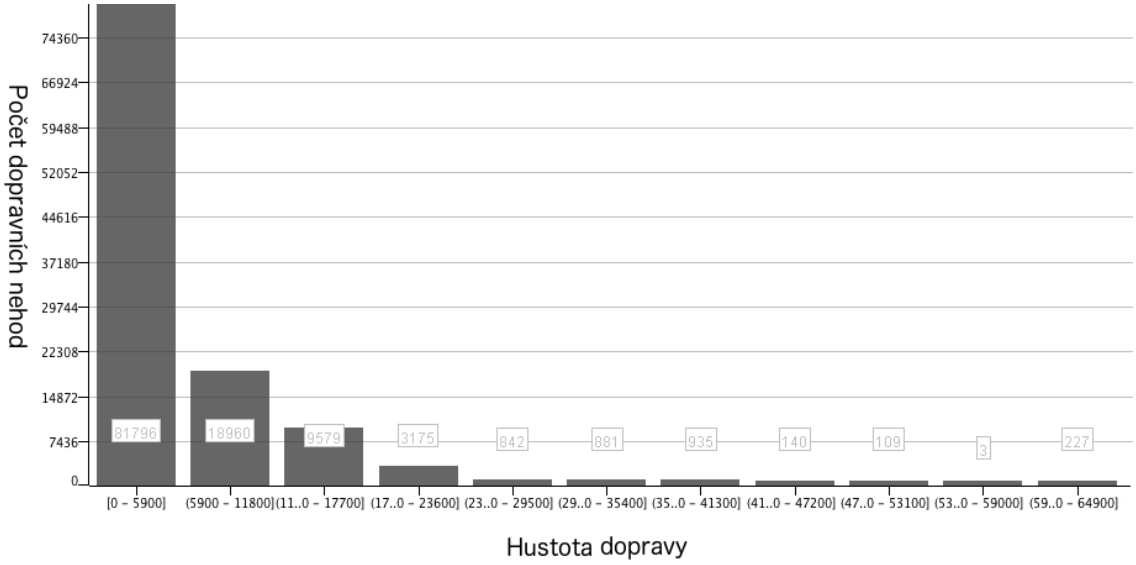
Příloha B: Histogram počtu nehod v závislosti na hustotě dopravy pro silnice 3. třídy

Příloha C: Histogram počtu nehod v závislosti na hustotě dopravy pro silnice 1. třídy

Příloha A: Seznam všech atributů ve vstupních datech a jejich datové typy

Columns: 50	Column Type	Column Index
Obec	String	0
Kraj	String	1
Datum nehody	String	2
Cas	String	3
Den	String	4
Druh pozemní komunikace	String	5
Číslo pozemní komunikace	Number (integer)	6
Zavinění nehody	String	7
Alkohol u viníka nehody	String	8
Usmrceno osob (počet)	Number (integer)	9
Těžce zraněno osob (počet)	Number (integer)	10
Lehce zraněno osob (počet)	Number (integer)	11
Druh nehody	String	12
Druh srážky jedoucích vozidel	String	13
Druh pevné překážky	String	14
Hlavní příčiny nehody	String	15
Druh povrchu vozovky	String	16
Stav povrchu vozovky v době nehody	String	17
Stav komunikace	String	18
Povětrnostní podmínky v době nehody	String	19
Viditelnost	String	20
Rozhledové poměry	String	21
Dělení komunikace	String	22
Situování nehody na komunikaci	String	23
Řízení provozu v době nehody	String	24
Místní úprava přednosti v jízdě	String	25
Specifické objekty v místě nehody	String	26
Směrové poměry	String	27
Místo dopravní nehody	String	28
Druh křižující komunikace	String	29
Smyk	String	30
Směr jízdy nebo postavení vozidla	String	31
Počet zúčastněných vozidel	Number (integer)	32
Druh vozidla	String	33
Výrobní značka motorového vozidla	String	34
Rok výroby vozidla	String	35
Vlastník vozidla	String	36
Celková hmotná škoda (100 Kč)	Number (integer)	37
Škoda na vozidle (100 Kč)	Number (integer)	38
Vozidlo po nehodě	String	39
Únik provozních, přepravovaných hmot	String	40
Způsob vyproštění osob z vozidla	String	41
Kategorie řidiče	String	42
Stav řidiče	String	43
Vnější ovlivnění řidiče	String	44
lng	Number (double)	45
lat	Number (double)	46
puv lng	Number (double)	47
puv lat	Number (double)	48
Hustota	Number (integer)	49

B: Histogram počtu nehod v závislosti na hustotě dopravy pro silnice 3. třídy



Příloha C: Histogram počtu nehod v závislosti na hustotě dopravy pro silnice 1. třídy