Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Porovnání metrik pro shlukovací algoritmy

Plzeň 2021 Jakub Schenk

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub SCHENK**
Osobní číslo:	**A18B0311P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Porovnání metrik pro shlukovací algoritmy**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s problematikou shlukování pro nehomogenní vstupní data.
2. Vytipujte různé metriky vhodné pro 2D a 3D body s různým rozložením a s dalšími atributy a implementujte je.
3. Otestujte implementace vybraných metrik na umělých i reálných datech.
4. Dosažené výsledky popište v textu práce a zhodnoťte.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Prof. Dr. Ing. Ivana Kolingerová**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **5. října 2020**
Termín odevzdání bakalářské práce: **6. května 2021**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 26. října 2020

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 6. května 2021

Jakub Schenk

Abstract

This Bachelor thesis deals with clustering methods used on real and artificial data while using different metrics.

The aim was to test different settings of clustering algorithm and observe differences between outcomes. By settings is meant switching between chosen metrics, classify for different numbers of clusters etc. on various input data. The outcome of observation is mentioned in this document.

Key words: Clustering, metrics

Abstrakt

Tato bakalářská práce se zabývá použitím metody shlukování na umělých i reálných datech, s použitím různých metrik.

Cílem práce je otestovat různá nastavení shlukovacího algoritmu a pozorovat, jak se liší výsledky. Nastavením je myšleno střídání vybraných metrik pro výpočet, klasifikovat pro různé množství shluků apod. na různých typech vstupních dat. Výsledky pozorování jsou uvedeny v tomto dokumentu.

Klíčová slova: Shlukování, metrika

Poděkování

Chtěl bych poděkovat Prof. Dr. Ing. Ivaně Kolingerové za její podporu, rady, čas a trpělivost, se kterou se mnou vedla konzultace. Rovněž děkuji Ing. Ondřeji Kaasovi za jeho odbornou pomoc a doc. T. Bayerovi z PřF UK, za výřezy reálných dat použitých při práci. Rád bych poděkoval rovněž své rodině, která mě v době studia a vypracování této práce podporovala.

Obsah

1 Úvod **1**
- 1.1 Definice problému . . . 1
- 1.2 Vstupní a výstupní data . . . 1

2 Shlukování **3**
- 2.1 Rozdělení metod shlukování . . . 3
  - 2.1.1 Hierarchické a nehierarchické shlukování . . . 3
  - 2.1.2 Connectivity models - Spojové modely . . . 3
  - 2.1.3 Centroid models - Modely s těžištěm . . . 4
  - 2.1.4 Distribution models - Distribuční modely . . . 5
  - 2.1.5 Density models - Hustotní modely . . . 7
- 2.2 Metrika . . . 9
  - 2.2.1 Euklidovská metrika . . . 9
  - 2.2.2 Manhattanská metrika . . . 9
  - 2.2.3 Geodetická metrika . . . 10
  - 2.2.4 Kosinová metrika . . . 10

3 Řešení **11**
- 3.1 Vybraný algoritmus a způsob měření vzdálenosti . . . 11
  - 3.1.1 K-means . . . 11
  - 3.1.2 Implementace geodetické vzdálenosti . . . 12
- 3.2 Python a jeho knihovny . . . 13
- 3.3 Implementace . . . 14

4 Experimenty a výsledky **16**
- 4.1 Technické prostředky . . . 16
- 4.2 Vlastnosti užitých dat . . . 16
  - 4.2.1 Umělá data . . . 16
- 4.3 Testy jednotlivých metrik na umělých datech . . . 20
  - 4.3.1 Testování euklidovské metriky . . . 20
  - 4.3.2 Testování geodetické metriky . . . 21
  - 4.3.3 Testování manhattanské metriky . . . 22
  - 4.3.4 Testování funkčnosti algoritmu k-means . . . 23
- 4.4 Kombinace metrik . . . 25
- 4.5 Demonstrace funkčnosti metrik na reálných datech . . . 27
- 4.6 Výsledky pokusů na reálných datech . . . 29

4.7 Časové nároky na výpočet 30

5 Závěr **35**

Literatura **36**

A Uživatelská dokumentace **39**
A.1 Classifier 39
A.1.1 Implementace 39
A.1.2 Parametry spuštění 40
A.1.3 Příklady spuštění 42
A.1.4 Výstupní soubory 43
A.2 Combiner 44
A.2.1 Výstupní soubory 44
A.3 Generator 45
A.3.1 Parametry spuštění 45
A.3.2 Výstupní soubory 45
A.4 SimplifyFile 45
A.5 Vizualizer 47

1 Úvod

Shlukování je pojem, který se s modernizací technologií rychle rozšiřuje. Čím více dat získáváme, tím rychleji je potřebujeme zpracovávat. Jedna z metod, kterou je k tomu možné využít, je právě shlukování. Cílem metody shlukování je roztřídit jednotlivé elementy do skupin na základě jejich podobnosti. Díky tomu je možné se zaměřit pouze na nějaké konkrétní elementy nebo velké množství zkoumaných prvků zjednodušit na několik elementů reprezentující celé skupiny těchto prvků. Díky těmto metodám jsme schopni zpracovat i tak velká data, že bychom s nimi za normálních podmínek nebyli schopni manipulovat.

Základní myšlenkou práce je zpracování velkých dat metodou shlukování a následné testování různých metrik využitých při shlukování. Slovu metrika můžeme zatím rozumět jako pravidlu, podle kterého určujeme, jaký element patří do jaké skupiny (podrobnější informace v kapitole 2).

Vysvětlení, co je to shlukování, a popis několika užívaných metod je uveden v kapitole 2. Samotný výpočet a řešení grafického zobrazení výsledků jsou popsány v kapitole 3. Testování nejprve funkčnosti programu a následně různých metrik je shrnuto v kapitole 4. Poslední částí je kapitola 5, která práci uzavírá.

1.1 Definice problému

Úkolem bylo vytvořit program, který by umožňoval měnit metriku, a pomocí shlukovacího algoritmu a této metriky shlukovat geodetická data. Pomocí programu by mělo být možné měnit „váhy“ jednotlivých prvků směrových vektorů elementů (kde pojem váha znamená většinou koeficient, kterým je hodnota daného prvku násobena), měnit samotný způsob výpočtu vzdálenosti (metriky) nebo i kombinovat několika metrik a vah. K tomu je potřeba se vyznat v problematice teorie shlukování pro nehomogenní data a v algoritmických řešeních, která jsou v praxi pro tyto účely využívána. Následně je nutné vybrat jedno řešení a to implementovat.

1.2 Vstupní a výstupní data

Vstupními daty do mého programu mohou být například geodetická nebo umělá data. Tato data jsou zapsána jako trojice souřadnic x, y, z. Jedná

se tedy o tři čísla oddělená mezerou, kdy každý řádek s touto trojicí čísel představuje jeden bod.

Výstupem tohoto programu pak je soubor, který obsahuje informaci o výsledném rozdělení bodů do jednotlivých shluků (nikoliv však původní data). Tato data lze následně interpretovat například tak, jak je tomu ve čtvrté kapitole. Více k tomu, jak je toho docíleno, je popsáno v kapitole 3.

2 Shlukování

Kapitola byla vypracována podle pramenů [9] [8] [10] [7].

Shlukování je způsob rozřazování prvků do skupin podle jejich vzájemné podobnosti do tzv. shluků. Pro člověka se nejedná o těžkou úlohu, pokud ji má řešit pro malé množství prvků. Pokud by ji měl ale řešit pro velkou množinu prvků s malými rozdíly, stává se tato úloha téměř neřešitelnou. Právě pro tyto úlohy slouží metody shlukování a jejich implementace v moderních zařízeních.

2.1 Rozdělení metod shlukování

2.1.1 Hierarchické a nehierarchické shlukování

Metod shlukování je mnoho, a lze je tedy dělit hned několika způsoby, např. na hierarchické a nehierarchické. Hierarchické metody vytvářejí shluky postupným spojováním prvků do větších celků, jsou ale náchylnější na chyby v datech a mohou být nestabilní (v porovnáním s nehierarchickým shlukováním). Jedná se o shlukování náročnějším způsobem, ale většinou tyto metody produkují čitelnější výsledky než nehierarchické metody.

Nehierarchické shlukování vytváří shluky pomocí spojování nebo naopak rozdělování jiných shluků podle určitých parametrů. Jedná se o rychlejší, stabilnější a průměrně přesnější oblast metod, jejíž výsledky jsou, na rozdíl od hierarchického shlukování, méně čitelné.

Stále populární a zároveň užitečné pro utvoření jasnější představy jsou dělení podle výsledného modelu nebo podle algoritmu, se kterým metoda pracuje. Populárnějším způsobem z těchto dvou se jeví dělení podle modelu, ale jelikož výsledné rozdělení do skupin je podobné u obou druhů dělení (a u některých pramenů se dokonce mísí), následuje pouze popis dělení podle výsledného modelu.

2.1.2 Connectivity models - Spojové modely

Spojové modely přistupují k problematice dvěma různými způsoby. Prvním je každý prvek klasifikovat jako jednu skupinu a pak podle podobnosti tyto skupiny slučovat, dokud nedosáhneme výsledku, např. 10 velkých shluků. Druhý způsob přistupuje k problematice opačně. Začínáme s jedním shlukem

všech dostupných prvků a poté zvyšujeme nároky na podobnost, dokud se náš shluk nerozdělí na dva, případně více menších shluků.

Nejznámějším algoritmem spadajícím do této skupiny je algoritmus hierarchického shlukování, který k prvkům přistupuje prvním ze zmíněných přístupů. Ostatní algoritmy řadící se do této skupiny nejsou zdaleka tak populární a většinou z hierarchického shlukování vycházejí.

Hierarchické shlukování

Tyto algoritmy se dělí do dvou hlavních skupin, podle přístupu k datům. Aglomerativní přístup pracuje zdola nahoru, kdy prvky začínají samostatně a postupně je shlukujeme do větších celků. Rozdělující přístup pracuje shora dolů, kdy začínáme se všemi prvky v jednom shluku a ten postupně dělíme.

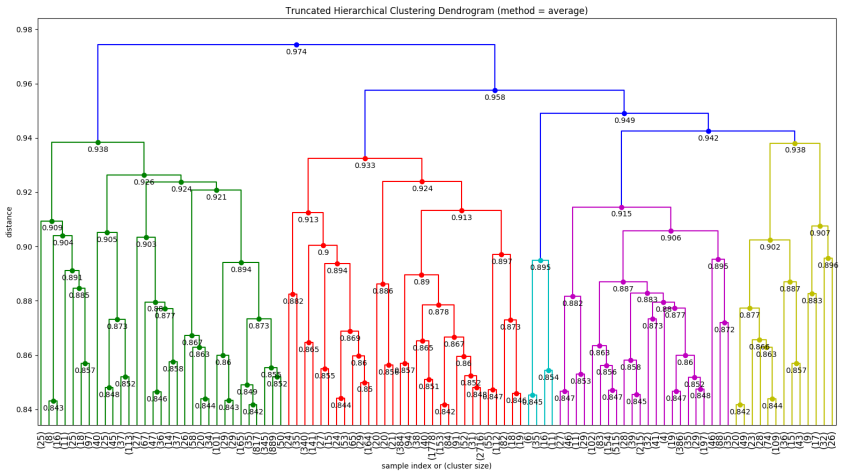
Pokud mluvíme o konkrétním algoritmu s tímto názvem, většinou máme na mysli algoritmus s následujícím postupem. Najdu dva prvky, které jsou k sobě „nejblíž“ (více o vzdálenosti mezi prvky v dalších podkapitolách) a tyto dva prvky sloučím do jedné skupiny. Tento krok je nutné opakovat, dokud nejsou splněny podmínky ukončení shlukování. Ukončení shlukování většinou nastává tehdy, když je dosažen počet požadovaných shluků, nebo když vzdálenost mezi jakýmikoliv dvěma prvky překračuje povolenou mez pro jejich spojení, případně při kombinaci obojího.

Celý výsledek pak lze zakreslit jako stromovou strukturu. Podle podmínky lze vybrat, jakou vrstvu tohoto stromu potřebujeme a budeme prezentovat jako výsledek shlukování, jak je vidět na obr. 2.1. Zde je zobrazen výsledek, při kterém bylo dosaženo pěti shluků různé velikosti, které jsou v grafické reprezentaci odděleny různými barvami (s výjimkou tmavě modré, která naznačuje, jak by vypadal výsledek s méně výslednými shluky).

Toto shlukování je jednoduché na implementaci a velice dobře čitelné. Zároveň není složité upravit ukončovací podmínku, a tím získat například jiné množství shluků s jinými vlastnostmi.

2.1.3 Centroid models - Modely s těžištěm

Pro tyto modely je nutné dopředu znát požadovaný počet shluků ve výsledku. Podle toho daná metoda přidá do modelu body, které označí jako těžiště nebo středy jednotlivých shluků. Tato kategorie metod rozřazuje prvky podle toho, jak blízko mají k těžištím. Tato akce rozřazení se následně několikrát opakuje, přičemž těžiště jsou po každé iteraci přemisťována do středů jejich shluků. Ukončující podmínkou většinou bývá to, že dvě po sobě jdoucí iterace jsou identické. Tím je myšleno, že žádný z pozorovaných prvků nebyl přiřazen k jinému těžišti oproti poslední iteraci.



Obrázek 2.1: Ukázka hierarchického shlukování s ukončující podmínkou [3]

Nejznámějším zástupcem této kategorie je algoritmus k-means. I tento algoritmus má několik svých variant, ale obecně funguje tak, že je zvoleno k různých středů (těžišť) shluků v modelu (náhodně nebo systematicky) a prvky se klasifikují do jednotlivých shluků na základě jejich pozice vůči pozicím těžišť. Dalším krokem je přemístění těžišť do středů svých shluků. Tím je zapříčiněno přiblížení těžiště k jiným prvkům a zároveň oddálení těžiště od prvků jiných. Proto je tento algoritmus několikrát opakován, dokud se všechna těžiště neustálí nebo dokud mezi dvěma iteracemi nedojde ke změně shluků.

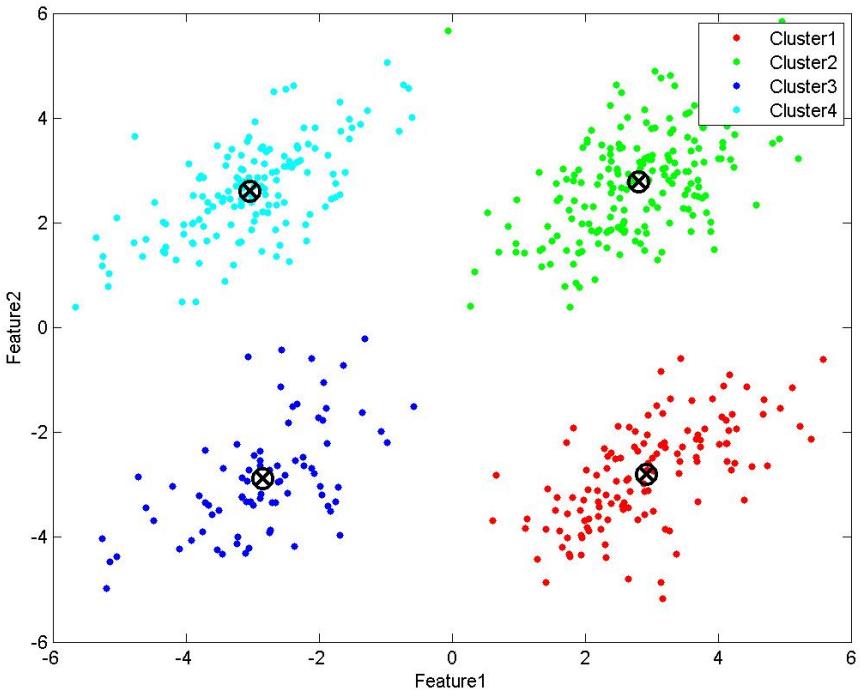
Případný výsledek takového algoritmu by mohl vypadat například tak, jak je znázorněno na obr. 2.2. Zde je znázorněn stav algoritmu klasifikace pro modely s těžištěm se čtyřmi středy shluků po ukončení výpočtu. Tyto shluky jsou barevně odlišeny a středy těchto shluků jsou znázorněny černou kružnicí s křížem.

K-means

Tomuto algoritmu se věnuji podrobněji v kapitole 3.1.1. Je to z toho důvodu, že jsem si tento algoritmus vybral pro implementaci při práci na tomto projektu.

2.1.4 Distribution models - Distribuční modely

Pro modely této kategorie je typické, že prvky nenáleží čistě k jednomu shluku. Naopak u každého prvku držíme informace o tom, jak pravděpo-



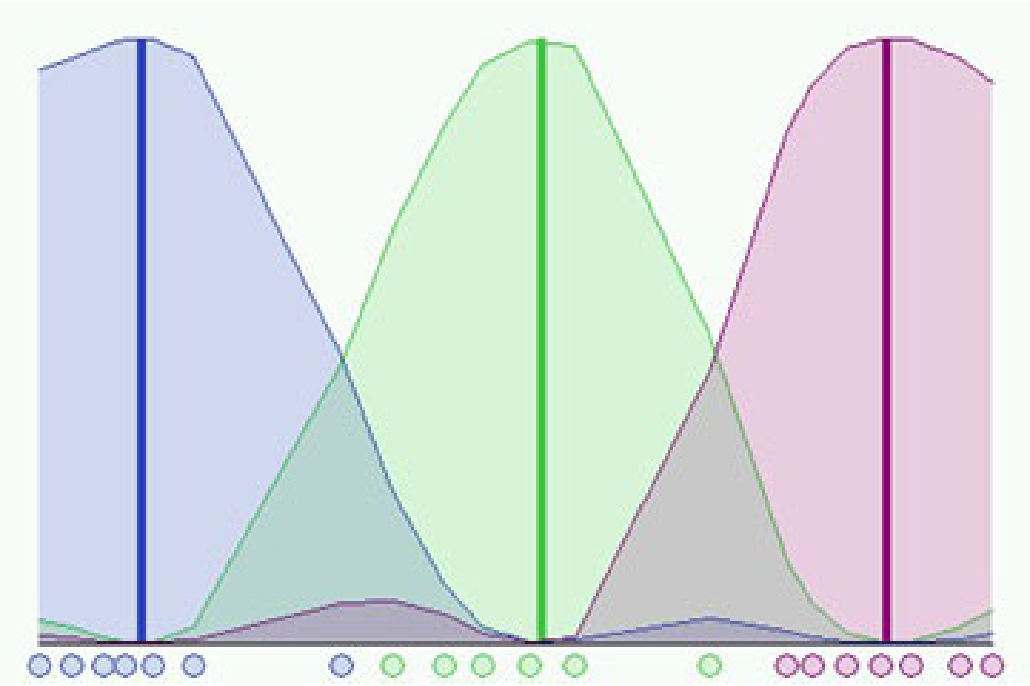
Obrázek 2.2: Ukázka shlukovaní podle modelu s těžištěm [1]

dobné je, že patří k nějakému shluku. Prvky, které jsou uprostřed mezi dvěma shluky, mají menší pravděpodobnost, že k jednomu z nich patří, než prvky, které jsou těsně vedle pomyslného středu shluku. Mezi nejznámější algoritmy řadící se do této kategorie řadíme například c-means nebo expectation-maximization algoritmy. Například expectation-maximization algoritmus funguje tak, že máme kroky E a M, kdy v kroku E vytvoříme podle odhadu a dat nějaké očekávání, a v kroku M vytvoříme podle tohoto očekávání odhad pro krok E v další iteraci.[11]

C-means

Algoritmus vycházející z k-means. K jeho zahájení je potřeba zadat požadovaný počet shluků. Následně je vytvořen příslušný počet středů a ty jsou postupně upravovány. Hlavním rozdílem od k-means je ten, že prvky nemusí náležet pouze jednomu shluku. Při každé iteraci je u každého prvku přepočítáváno, jak je pravděpodobné, že patří ke každému shluku. Čím je prvek blíže ke středu shluku, tím pravděpodobnější je, že k tomuto shluku náleží. Naopak, pokud se prvek nachází uprostřed mezi dvěma shluky, nemusí být rozhodnuto, ke kterému z těchto shluků náleží. Je jednoduše určeno, že

k oběma shlukům náleží se stejnou, resp. podobnou pravděpodobností.



Obrázek 2.3: Ukázka C-means shlukovaní [2]

Na obr. 2.3 je znázorněn výsledek klasifikace pomocí takového algoritmu. Ve spodní části grafu jsou znázorněny klasifikované body, zatímco v horní části je rozložení pravděpodobnosti s jakou body náleží k jakému shluku. Shluky jsou opět barevně odděleny, a jelikož není jednoduché výsledek zakreslit, tak klasifikované body mají barvu shluku, ke kterému mají nejblíže, nebo ke kterému náleží s největší pravděpodobností.

C-means je označován jako přednostní algoritms tzv. fuzzy shlukování, které je často označováno za soft clustering nebo soft k-means (měkké shlukování nebo měkké k-means). Existuje dokonce i možnost dělení všech shlukovacích algoritmů na soft clustering a hard clustering, což by opět c-means a algoritmy s podobnou myšlenkou stavělo do velice odlišné kategorie od ostatních druhů shlukování. Podle popisů těchto algoritmů se však domnívám, že se jedná jen o jiné označení celé této kategorie, tedy distribution models - distribuční modely.

2.1.5 Density models - Hustotní modely

Základní myšlenkou této kategorie metod je, že metoda hledá oblasti s neobvykle mnoha prvky. Taková místa označí za shluk. Výsledkem ale často může být, že spousta relativně izolovaných bodů nebude náležet k žádnému

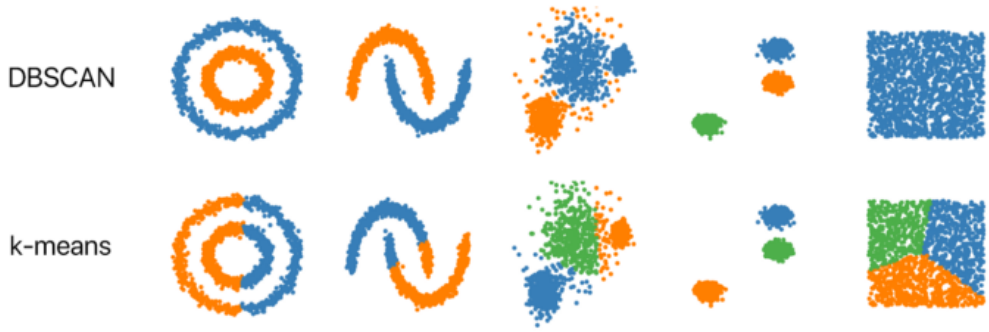
shluku, resp. ony samy budou svým shlukem. Druhým úskalím, které tato kategorie metod přináší, je, že pokud se nějaké shluky protínají, jsou ve výsledku klasifikovány jako jeden shluk. Přesto může být tato kategorie metod velice užitečná, pokud se vyvarujeme příliš hustým grafům, nebo např. pokud nechceme, aby vzdálenost mezi dvěma libovolnými prvky přesáhla určitou hranici. Populárním algoritmem z této kategorie je např. DBSCAN nebo OPTICS.

Density-based spatial clustering of applications with noise (DBSCAN)

V překladu to znamená něco jako „Prostorové shlukování aplikací s šumem založené na hustotě“ a jedná se o populární způsob shlukování využívaný hlavně při strojovém získávání dat nebo strojovém učení.

Algoritmus na základě vzdáleností mezi prvky určí, kde přibližně je nejvíce podobných prvků, a vytvoří tam shluk. Těchto shluků vytváří několik podle toho, kde mají prvky mezi sebou malé vzdálenosti a je jich tam dostatečné množství. Místa, kde prvků není dostatečné množství, označí za izolované prvky nebo skupiny prvků.

Na obr. 2.4 jsou znázorněny rozdíly ve výsledcích algoritmu DBSCAN a k-means, jakožto „zástupců“ kategorií „Hustotní modely“ a „Modely s těžištěm“. Tyto rozdíly jsou znázorněny na pěti různých datasetech v rovině a lze z nich snadno vypozorovat odlišnost přístupu obou kategorií.



Obrázek 2.4: Ukázka DBSCAN shlukovaní [9]

Pro spuštění algoritmu je tedy potřeba zadat dva parametry. Jedním z nich je maximální vzdálenost mezi dvěma prvky, které můžeme zařadit do jednoho shluku, a druhým je minimální počet prvků ve skupině, aby tato skupina mohla být označena za samostatný shluk.[6]

2.2 Metrika

Slovem metrika je myšlen způsob určení vzdálenosti mezi jednotlivými prvky. Jedná se o základní nástroj klasifikace, který umožňuje matematicky určit velikost rozdílů mezi dvěma zkoumanými prvky. Pojmem vzdálenost v tomto kontextu není nutně myšlen fyzický rozdíl poloh dvou prvků jako spíš rozdíl atributů, se kterými pracujeme. Tohoto lze využít při klasifikaci tak, že prvky s menšími rozdíly řadíme do jednotlivých skupin a prvky s většími rozdíly řadíme do skupin jiných. Při klasifikaci prvků je vzdáleností myšleno to, jak moc jsou dané prvky rozdílné. Ta se ale dá interpretovat různě. Mezi nejznámější řadíme euklidovskou metriku, manhattanskou (nebo taky City block) metriku, geodetickou metriku, kosinovou metriku a další.

2.2.1 Euklidovská metrika

Jedná se o nejčastěji používanou metriku. Euklidovská metrika je délka úsečky mezi dvěma body. Jedná se o tu metriku, kterou používáme všichni, pokud v běžné konverzaci zmíníme vzdálenosti bodů.

$$d_e(a,b) = \sum_{i=1}^{N} \sqrt{(a_i - b_i)^2} \tag{2.1}$$

Zde představuje člen $d_e(a,b)$ výslednou vzdálenost mezi body a a b. N představuje počet složek vektoru (a tedy i velikost dimenze).

2.2.2 Manhattanská metrika

Tato metrika je také známa pod názvem City block metric. Jedná se totiž o metriku, která se počítá jako součet vzdáleností dvou bodů ve všech osách souřadnicového systému, neboli vzdálenost dvou bodů, pokud ji mohu měřit pouze ve směrech osy.

Výpočet této metriky je procesově nenáročný, protože se jedná o sčítání a odčítání, na rozdíl od předchozí metriky. Proto v úvahu připadá následující vzorec:

$$d_m(a,b) = \sum_{i=1}^{N} |a_i - b_i| \tag{2.2}$$

Kde $d_m(a,b)$ je výslednou vzdáleností, N je opět dimenze prostoru a prvky a a b jsou body, mezi kterými vzdálenost měříme.

2.2.3 Geodetická metrika

Geodetická metrika je méně obvyklá, ale zejména pro tuto práci velice důležitá metrika. Užívá se při výpočtech povrchových vzdáleností. Žádný jednoduchý jednokrokový vzorec pro tuto metodu neexistuje, není tedy možné vzdálenost libovolného prvku od svého středu vypočítat bez znalosti vlastností ostatních prvků. Metrika mezi dvěma prvky je totiž definována jako nejmenší součet vzdáleností všech prvků mezi nimi. Jedná se o netriviální hledání nejkratší lomené čáry mezi dvěma body (s využitím bodů mezi nimi) a triviální výpočet její délky, kde body zlomu této čáry jsou prvky měření a úsečky mezi nimi jsou vzdálenosti mezi těmito prvky. Pokud bychom tuto metriku chtěli vypočítat, pravděpodobně bychom použili vzorec pro výpočet délky lomené čáry, což je v podstatě suma jednotlivých úseček obsažených v této lomené čáře spočítané podle již výše uvedeného vzorce pro výpočet euklidovské metriky.

$$d_g(a,b) = \sum_{i=1}^{N-1} d_e(c_i, c_{i+1}) \tag{2.3}$$

Kde $d_g(a,b)$ je výslednou vzdáleností, N je počet úseček, ze kterých se skládá měřená lomená čára, $a = c_1$ a $b = c_M$ a jsou tedy počáteční a konečný bod, c_j popř. c_{j+1} jsou mezilehlé body v lomené čáře. Prvek d_e je vzorec 2.1 pro výpočet euklidovské metriky.

2.2.4 Kosinová metrika

Kosinová metrika je velice odlišná od ostatních. Vzdáleností definovanou touto metrikou totiž myslíme úhel svíraný vektory měřených bodů. Jedná se o kosinus tohoto úhlu, tedy výsledek leží v intervalu -1 a 1. Často se z praktických důvodů výsledek převádí na absolutní hodnotu, tedy do intervalu 0 až 1. Pro výpočet existuje následující vzorec:

$$d_c(a,b) = \frac{\sum_{i=1}^{N}(a_i b_i)}{(\sqrt{\sum_{i=1}^{N} a_i^2}\sqrt{\sum_{i=1}^{N} b_i^2})} \tag{2.4}$$

Zde představuje člen $d_c(a,b)$ výslednou vzdálenost (resp. odchylku) mezi vektory bodů a a b. N představuje počet složek vektoru (a tedy i velikost dimenze vektorového prostoru).[4]

Tato metrika je velice užitečná při měření podobnosti textů. Tedy je možné (a oblíbené) ji v tomto oboru využívat.

3 Řešení

Po dostatečném nastudování problematiky úkolu mi bylo navrženo, abych si s úkolem zkusil poradit za pomoci již implementovaných Python knihoven (této části vypracování se věnuji v následující podkapitole). Počáteční experimenty s knihovnami v jazyce Python ukázaly, že jejich úprava do podoby, kterou jsme chtěli, nebude triviální, a protože by stávající podoba byla pro zamýšlenou práci příliš omezující, rozhodl jsem se pro vlastní implementaci v jazyce Java.

3.1 Vybraný algoritmus a způsob měření vzdálenosti

Pro implementaci jsem vycházel z již zmíněného algoritmu k-means. Je to z toho důvodu, že je snadno pochopitelný, jednoduchý a zároveň velice efektivní. V neposlední řadě se jedná také o algoritmus, který se na podobný typ úloh aplikuje v běžné praxi. Začal jsem shlukováním pomocí euklidovské metriky a postupně programu dodával různé možnosti počítání vzdálenosti (metriky). Program nyní disponuje možnostmi pro výpočet vzdálenosti pomocí euklidovské, geodetiké a manhattanovské metriky. Po drobných úpravách je možné výsledky těchto výpočtů kombinovat, a získat tak kombinace jednotlivých metrik. Výsledky klasifikace jsou zobrazovány pomocí knihoven Pythonu (které byly zmíněny v úvodu této kapitoly a je jim věnována ještě jedna z následujících podkapitol), takže formát výsledků tomu odpovídá.Výstupem programu je několik souborů, které mohou být následně zpracovány a zobrazeny právě pomocí těchto Python knihoven.

3.1.1 K-means

Algoritmus k-means je jedním z velice populárních algoritmů řadících se do skupiny Centroid models nebo Modely s těžištěm, blíže popsané v kapitole 2. Jedná se o algoritmus, který manipuluje s několika přidanými body a následně k nim pak přiřazuje shlukované elementy.

Nejprve je nutné načíst informace o prvcích do paměti stroje. Následuje přidání několika bodů jako budoucích středů shluků. Počet těchto bodů musí být zadán před provedením shlukování a jsou do modelu přidány většinou náhodně. Následně je potřeba projít veškeré elementy určené ke klasifikaci a

podle metriky zjistit, ke kterému ze středů jsou nejblíže, a k tomuto středu je klasifikovat. Dalším krokem je pak přemístění středů do centra svých shluků. Tímto krokem se od některých bodů vzdálí nebo naopak se k nim přiblíží. Z toho důvodu jsou tyto kroky (určení příslušnosti elementů a změna pozice středu shluku) opakovány, dokud se středy neustálí na jednom místě, nebo dokud není zaznamenána nulová změna, co se týče příslušnosti jednotlivých elementů ke shlukům. Tehdy můžeme tyto elementy prohlásit za klasifikované a shlukování za ukončené.

Proč k-means

Pro svou práci jsem zvolil k-means, protože se jedná o metodu shlukování, která je nenáročná na implementaci a zároveň je velice efektivní a flexibilní, co se týče dat a práce s nimi. Jelikož v datech není potřeba vyhledávat nějaké extrémy nebo izolované body a pracuje se s velkým množstvím dat, je potřeba využít takového algoritmu, který je rychlý a dělá s daty to, co je potřeba. Jelikož rychlost základního k-means je oproti ostatním algoritmům poměrně vysoká, jednalo se o jasnou první volbu. Volba algoritmu mohla být ovlivněna ještě jedním faktorem, a to sice tím, že k-means je jedna z metod, o kterých jsem věděl, a díky poznatkům získaných během studia si dokázal představit i to, jak pracují.

3.1.2 Implementace geodetické vzdálenosti

Každý z bodů si nese informaci o jejich nejbližších sousedech. Abych se vyhnul nepříjemnému zanořování při prohledávání do hloubky nebo do šířky, rozhodl jsem se body postupně obarvovat podle toho, jestli jsou obarveni jejich sousedé či nikoliv. To vede k opakovanému procházení bodů a jejich sousedů, až nakonec zůstanou pouze klasifikované body (body náležící nějakému ze středů). Body si zároveň uchovávají informaci o vzdálenosti ke středu, kterému náleží. Tato vzdálenost je počítána jako součet vzdáleností jednotlivých bodů, přes které je nejblíže k jejich středu, neboli se jedná o délku lomené čáry se zlomy v bodech. Počátečním bodem je střed shluku, ke kterému náš bod klasifikuji, a koncovým bodem je klasifikovaný bod. Během výše popsané klasifikace tedy dochází ještě ke kontrole hraničních bodů (bodů, jejichž sousedé se „neshodnou“ na jednom středu) a kontroluje se, jestli „se hlásí“ ke správnému středu. Tento algoritmus navíc pracuje pouze s předchozí iterací, tedy nedochází zde ke zkreslení, které by mohlo způsobit klasifikování části dat a následné počítání s těmito výsledky. Z tohoto důvodu je toto řešení mnohem přesnější, ale zároveň o něco náročnější na výpočet.

Jsem si vědom toho, že toto řešení rozhodně není optimální a je poměrně pomalé. Přistoupil jsem k němu, protože je šetrnější na paměť a to je při manipulaci s velkými daty často velice důležité. Zkoušel jsem i dříve zmíněné algoritmy s prohledáváním do šířky a do hloubky, ale jelikož jsem si zvolil programovací jazyk, který není pro tyto aplikace optimální, tak jsem měl problémy s již zmíněnou pamětí.

Toto řešení je navíc zajímavé tím, že pokud bychom měli prvky pouze v rovině, byly by rovnoměrně rozmístěné a graf by byl dostatečně hustý, výsledky by mohly připomínat Voroného diagramy, resp. byly by jejich podmnožinou. To platí i pro veškeré iterace.

Další důvod, proč jsem se rozhodl pro tento způsob výpočtu, je, že většinou se programy s podobnou funkcionalitou implementují a počítají s pomocí meshe. Toto řešení nic takového nemá a počítá jen s nejbližšími sousedy každého bodu. To může být občas nevýhodou, ale lze díky tomu získat trochu jiné výsledky, které mohou být v některých případech vhodnější. Jedná se o jinou metodu získání geodetické metriky, která stejně jako mesh dokáže sjednotit body do shluků na základě jejich polohy bez toho, aby je shlukovala „vzduchem“.

3.2 Python a jeho knihovny

První doporučenou cestou, jak program vytvořit, bylo využít již implementovaných Python knihoven a algoritmus s jejich pomocí implementovat v Python Notebooku. Tato varianta vypadala původně velice dobře, protože knihovny pracovaly velice rychle, efektivně a byly v nich už implementované algoritmy, které byly potřeba. Nevýhodou ovšem je, že takto získané algoritmy se velice těžko upravují podle toho, jak autor chce. Nebylo časově přijatelné upravit implementaci algoritmu, a změnit tak možnosti úpravy metriky využívanou v jejich výpočtu.

Vzhledem k dlouhodobému onemocnění Ing. Ondřeje Kaase, který mi měl odborně pomoci s implementací v tomto prostředí a programovacím jazyce, bylo rozhodnuto, že výsledný algoritmus bude implementován v programovacím jazyce Java bez použití cizích knihoven. Dosavadní výsledky byly využity na implementaci programu, který zobrazuje výsledky pomocí grafických knihoven.

3.3 Implementace

Základní program je tvořen čtyřmi třídami, přičemž parametry jsou zadány při spuštění. Nejprve jsou zpracovány příchozí parametry. Poté jsou načtena poskytnutá data o prvcích, která jsou zpracována podle potřeby, během čehož jsou zjištěny parametry celé testované skupiny prvků. Pak jsou podle parametrů vygenerovány středy shluků a v případě geodetické vzdálenosti spočítáni nejbližší sousedé každého z prvků. Následuje první klasifikace všech bodů k jednotlivým středům shluků podle zvolené metriky. V případě zvolení využití algoritmu k-means jsou přepočítány pozice středů shluků a poslední dva kroky jsou opakovány, dokud nedojde ke splnění ukončovací podmínky. Tou může být to, že proběhlo příliš mnoho operací, nebo že v posledních dvou iteracích neproběhla žádná změna. Posledním krokem je zápis výsledku do souborů.

Tyto výsledné soubory je nakonec možné zpracovat a zobrazit druhým, již zmíněným programem, napsaným v Python Notebooku. Ten načte informace o bodech a následně výsledky a to vykreslí jak v rovině, tak následně i v prostoru.

Při práci na projektu vznikly ještě tři malé třídy v Javě, které by tu měly být zmíněny. Všechno to jsou samostatně spustitelné programy manipulující se soubory, se kterými pracuje hlavní program.

První z těchto programů byl vytvořen za účelem generování vlastních umělých dat, takže vytváří soubor podle metody, která je v něm spuštěna. V tomto souboru jsou popsány body třemi souřadnicemi na každé řádce. Tímto programem byla vytvořena všechna umělá data popsaná níže v tabulce 4.1

Druhý z těchto programů umožňuje úpravu vstupních dat pro hlavní program. Data buď zmenší tak, že rovnoměrně odebere nějaké množství bodů, nebo data upraví tak, aby měly jednotlivé body vždy jen tři souřadnice. To proto, že hlavní program vždy zpracuje pouze první tři souřadnice a program pro vizualizaci výsledku vyžaduje právě tři souřadnice.

Třetí z těchto programů umožňuje kombinace různých výsledků pokusů provedených na jedněch datech s různě zvolenou metrikou. Je tedy možné zkombinovat např. výsledky pokusů s euklidovskou metrikou a výsledky pokusů s geodetickou metrikou v určitém poměru. Pro spuštění tohoto programu musí být známy výsledky všech kombinovaných pokusů. Výsledky jsou zapsány do stejného souboru, do kterého zapisuje výsledky i hlavní program, můžou být tedy zobrazeny stejným způsobem.

Algoritmus k-means, který je implementován v hlavní třídě programu, má dvě různé ukončovací podmínky. První z nich je ukončení po dosažení

optimálního výsledku, tedy ve dvou po sobě jdoucích iteracích nedošlo k žádnému rozdílu. Žádný ze středů shluků se v tomto případě nesmí přemístit oproti předchozí iteraci. Z důvodu dosažitelnosti výsledku v přijatelném časovém horizontu je do programu implementována dodatečná, tedy druhá ukončovací podmínka, která ukončí činnost k-means po maximálně 40 iteracích, kdy by měl být výsledek už dostatečně přesný. Většina z testovaných dat této podmínky nedosáhne a výpočet je ukončen podmínkou o ustálení pohybu středů shluků.

4 Experimenty a výsledky

4.1 Technické prostředky

Při své práci jsem využíval vlastního techniku. Jedná se o notebook Dell G3 15 3500, bez úprav na hardware. Operačním systémem při vývoji byl Windows 10. Program byl vyvíjen výhradně v programu IntelliJ Idea Cominity Edition 2019.2.4 v programovacím jazyce Java v. 11.

Reálná data využitá při experimentech byla poskytnuta doc. Ing. Tomášem Bayerem, Ph.D. [5]. Data umělá jsem vytvořil pro účel demonstrace a testování sám.

4.2 Vlastnosti užitých dat

Data přijímána programem jsou zapsána v textovém souboru v požadovaném formátu. Z důvodu vlastností poskytnutých dat je ignorován první řádek tohoto souboru. Všechny další řádky obsahují jednotlivé souřadnice jednoho z bodů. Poslední řádka souboru je prázdná.

Data nejsou zobrazována v kartézské soustavě souřadnic, mohou být tedy zkreslená měřítkem. Je tomu proto, aby byly výsledky lépe vidět. Červenou barvou jsou vyznačeny středy shluků (ty ovšem můžou být zakryté jinými body, takže nemusí být vždy dobře viditelné), zatímco ostatními barvami jsou oddělené jednotlivé shluky. U reálných dat jsou pak jednou barvou označeny body nedostupné, pokud byly klasifikovány geodetickou metrikou (většinou žlutě).

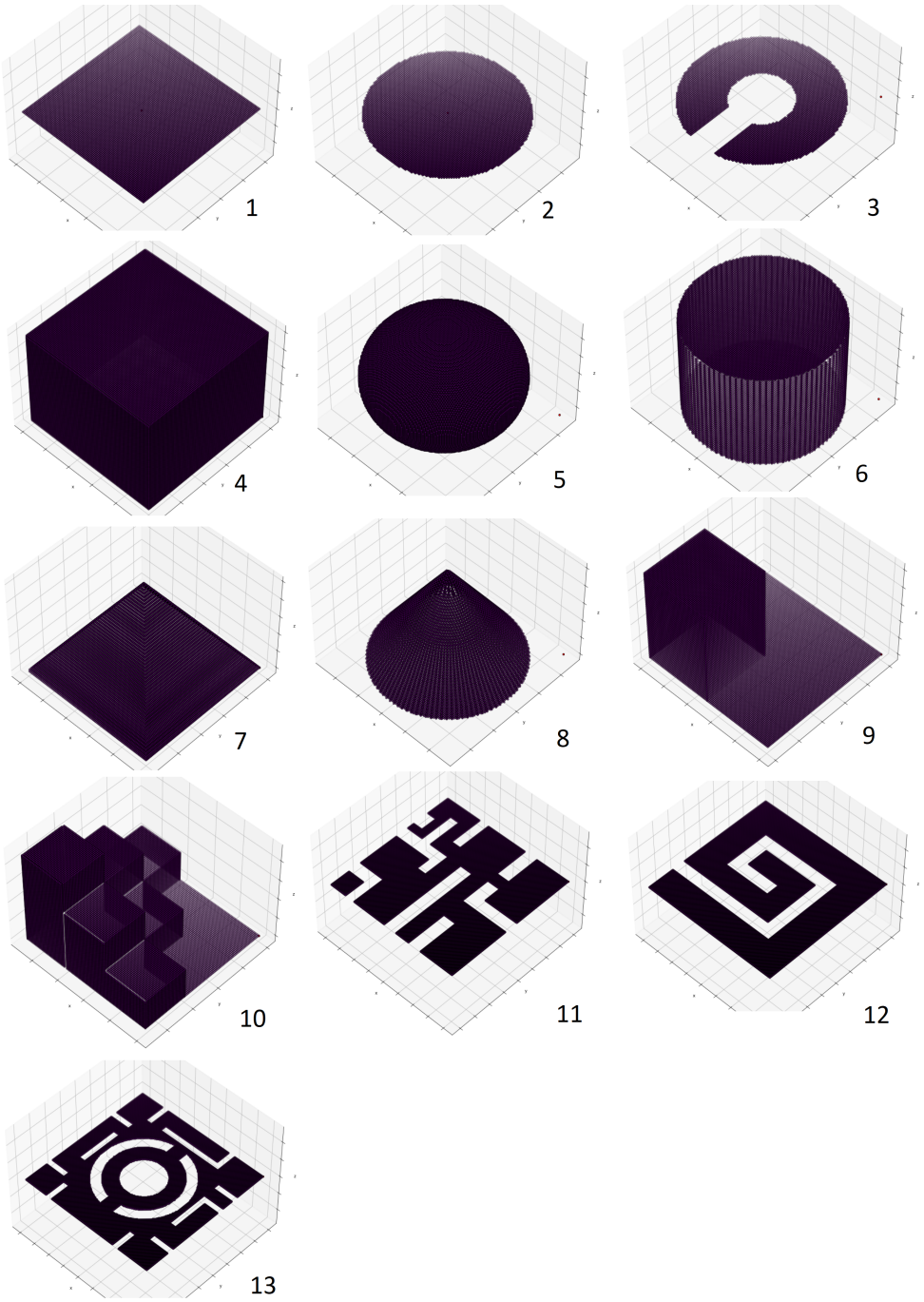
4.2.1 Umělá data

V tabulce 4.1 jsou základní informace o umělých datech, která byla při práci použitá. Na obr. 4.1 je pak vidět, jak tato data vypadají zobrazena, přičemž číslo u každého zobrazeného objektu koresponduje s číslem datasetu zmíněným v tabulce.

Stejně tak tomu je pro reálná data v tabulce 4.2 a na obr. 4.2.

Název	Počet bodů	Popis	Číslo datasetu
Čtverec	10000	Body v rovině uspořádané rovnoměrně do tvaru čtverce.	1
Kruh	7825	Body v rovině uspořádané rovnoměrně do tvaru kruhu.	2
C	5991	Body v rovině uspořádané rovnoměrně do tvaru znaku C.	3
Krychle	60000	Body v prostoru uspořádané rovnoměrně do tvaru krychle.	4
Koule	60374	Body v prostoru uspořádané rovnoměrně do tvaru koule.	5
Válec	20150	Body v prostoru uspořádané rovnoměrně do tvaru válce.	6
Jehlan	20200	Body v prostoru uspořádané rovnoměrně do tvaru jehlanu.	7
Kužel	15412	Body v prostoru uspořádané rovnoměrně do tvaru kužele.	8
Schod 1	20000	Body v prostoru uspořádané rovnoměrně do tvaru krychle v rohu čtvercové podložky.	9
Schod 2	29520	Body v prostoru uspořádané rovnoměrně do složitějšího pravidelného tvaru.	10
Místnost 1	29185	Body v rovině uspořádané rovnoměrně tak, aby aby bylo možné otestovat některé vlastnosti metrik.	11
Místnost 2	31279	Body v rovině uspořádané rovnoměrně tak, aby aby bylo možné otestovat některé vlastnosti metrik.	12
Místnost 3	28227	Body v rovině uspořádané rovnoměrně tak, aby aby bylo možné otestovat některé vlastnosti metrik.	13

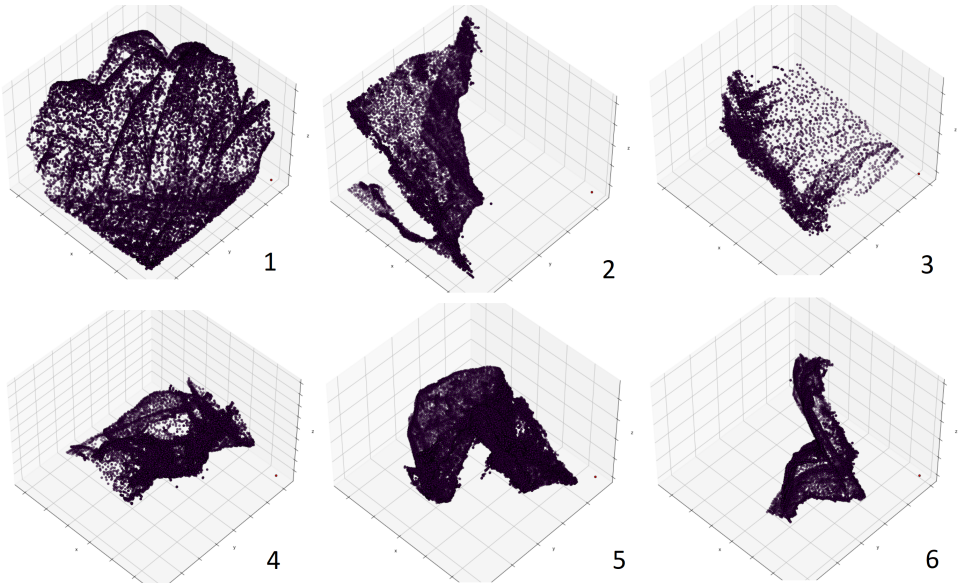
Tabulka 4.1: Umělá data



Obrázek 4.1: Umělá data použitá při testování

Název	Počet bodů	Popis	Číslo datasetu
Data 1	23582	Reálná data	1
Data 2	16927	Reálná data	2
Data 3	6170	Reálná data	3
Data 4	17139	Reálná data	4
Data 5	39961	Reálná data	5
Data 6	10414	Reálná data	6

Tabulka 4.2: Reálná data



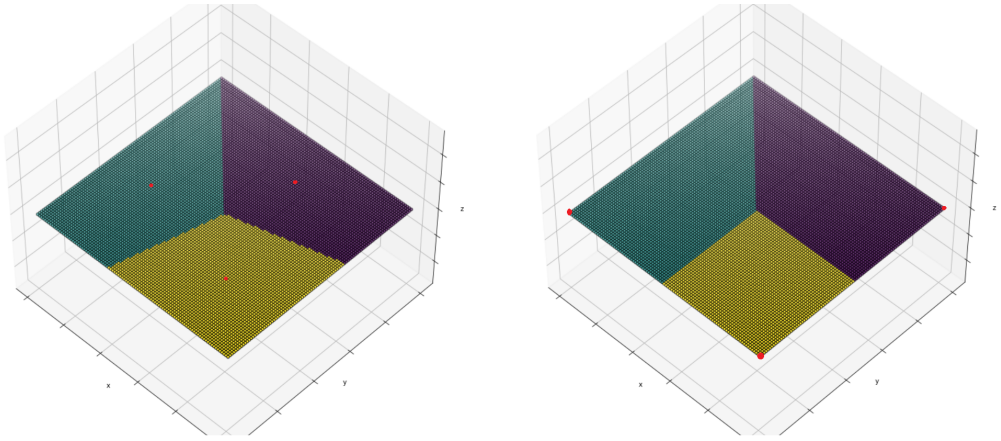
Obrázek 4.2: Reálná data použitá při testování

4.3 Testy jednotlivých metrik na umělých datech

Následuje popis výsledků jednotlivých testů, které mají ukázat fungování programu na různých datech. Tyto výsledky jsou zobrazovány již zmíněným způsobem. Při vizualizaci výsledku dochází ke zkreslení z toho důvodu, že zobrazená data nejsou prezentována v kartézské soustavě souřadnic. Souřadnicový systém je ve všech osách roztažen tak, aby výsledek byl na grafu vidět co možná nejlépe.

4.3.1 Testování euklidovské metriky

Nejlepším způsobem, jak zjistit a následně demonstrovat správnost implementace je programem zkusit klasifikovat data, u kterých je předvídatelný správný výsledek. Data prezentována v této kapitole jsou uměle vygenerované kvantum bodů v rovině nebo v prostoru. Data generovaná v prostoru jsou množiny bodů rovnoměrně vygenerovaných pouze na površích jednoduchých geometrických objektů.



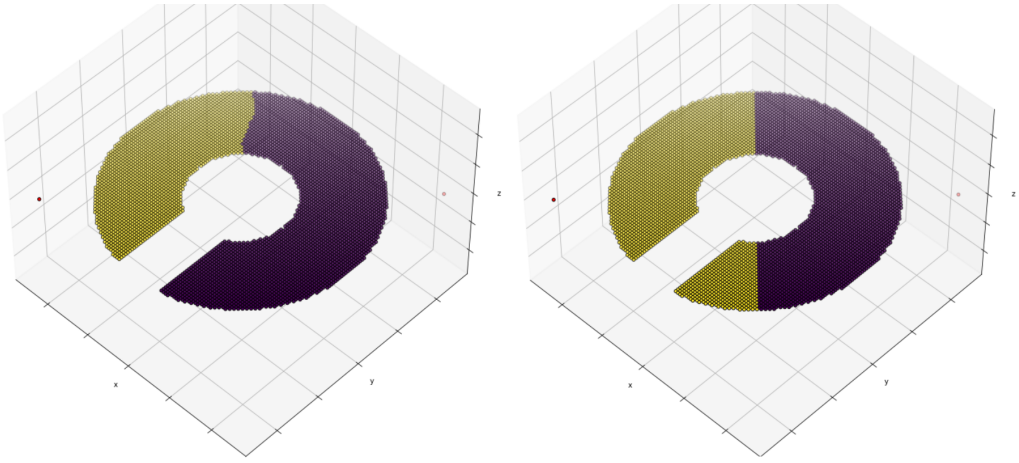
Obrázek 4.3: Test euklidovské metriky na datech Čtverec (4.2.1)

Na obrázku obr. 4.3 je zobrazeno porovnání dvou měření jednoduchého chování euklidovské metriky. Oba testy byly provedeny na stejných datech se třemi středy s počáteční pozicí ve třech různých rozích čtverce (4.2.1). Test vlevo byl spuštěn a zpracován pomocí deseti iterací algoritmu k-means. Liší se tím, že byl spuštěn pouze pro otestování správnosti implementace euklidovské metriky, tudíž neproběhly žádné iterace algoritmem k-means, tedy počáteční poloha středů nebyla v průběhu měření změněna.

Jelikož byl tento výsledek očekáván, můžeme předpokládat, že se funkce pro výpočet chová tak, jak by měla. Další ukázky klasifikace pomocí této metriky lez vidět v následujících odstavcích, kdy jsou jednotlivé výsledky porovnávány s výsledky získané pomocí této metriky. Je tomu tak, protože tato metrika je pro nás nejvíce přirozená a jednoduchá na pochopení. Rozdíly ostatních metrik tak lze v porovnání s ní snadno ukázat.

4.3.2 Testování geodetické metriky

To, jestli výpočet pomocí geodetické metriky funguje dle zadání, lze zjistit pomocí modelu dat s nějakou mezerou, přes kterou by se body neměly shlukovat, jako by tomu bylo s výpočtem pomocí metriky např. euklidovské.

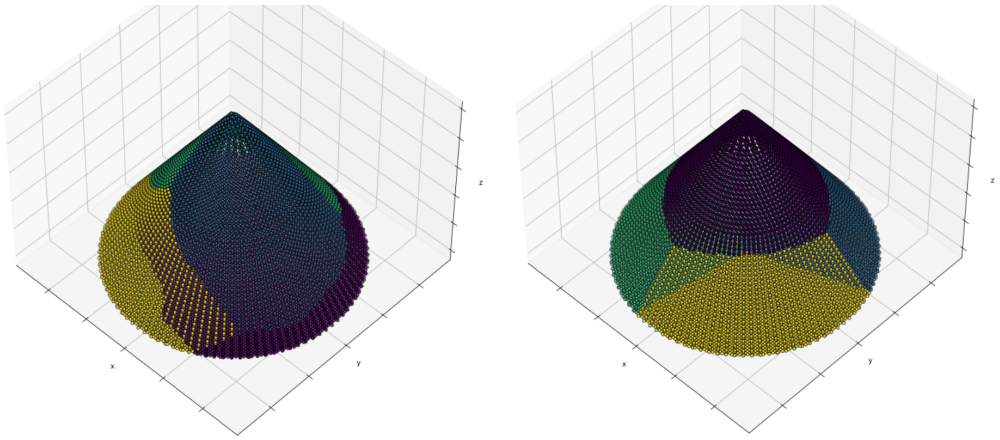


Obrázek 4.4: Rozdíly mezi geodetickou a euklidovskou metrikou v rovině

Toto lze pozorovat na obr. 4.4, kdy shlukujeme se dvěma středy shluků na umělých datech s pracovním názvem C (4.2.1). Vlevo je zobrazen výsledek pokusu pro výpočet pomocí geodetické metriky a vpravo pokus provedený za stejných podmínek jen s využitím euklidovské metriky. Oba pokusy proběhly bez přesunů středů shlukování, nebyl tedy využit algoritmus k-means, aby byly ve výsledku zřetelně vidět rozdíly mezi metrikami.

Na obr. 4.5 je zobrazeno porovnání výsledků pro podobný pokus. Shlukování bylo prováděno na datech Kužel (4.2.1). Body jsou shlukovány do čtyř shluků a tentokrát bylo nastaveno, že algoritmus k-means přemístí středy shluků do optimálních pozic v maximálně čtyřiceti iteracích.

Na levé straně je výsledek pokusu spuštěného s geodetickou metrikou. Díky přesunům středů podle k-means jsou všechny shluky přibližně stejně velké. Na pravé straně je výsledek vypočítaný pomocí euklidovské metriky.



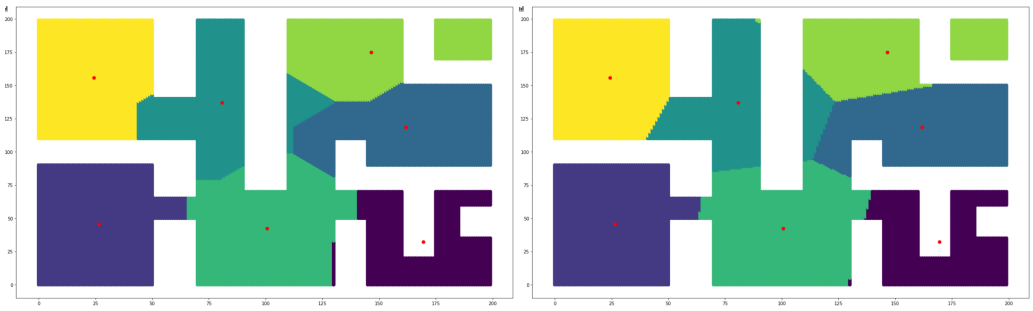
Obrázek 4.5: Rozdíly mezi geodetickou a euklidovskou metrikou v prostoru

Shluky jsou mnohem pravidelnější a díky k-means opět rovnoměrně rozložené a podobně rozsáhlé.

4.3.3 Testování manhattanské metriky

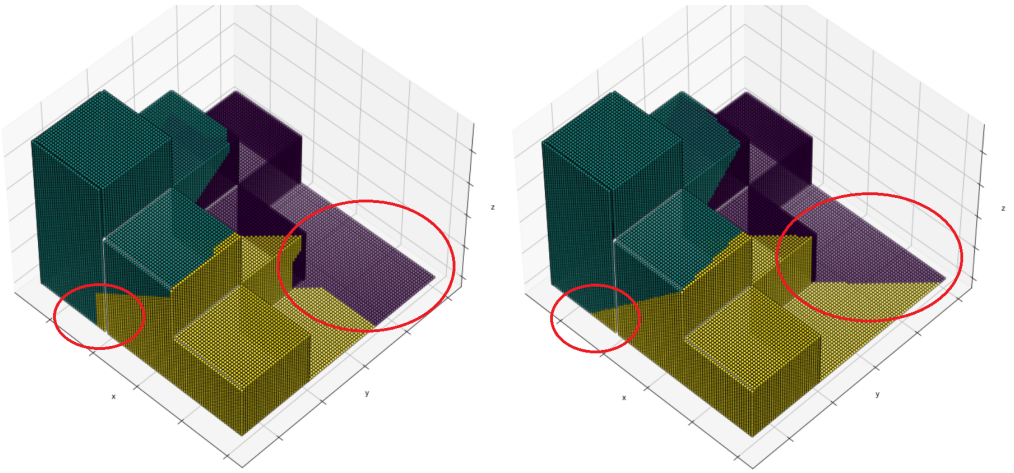
Rozdíly mezi manhattanskou a euklidovskou metrikou jsou mnohem menší oproti rozdílům s geodetickou metrikou. Je to způsobeno tím, že vzorce a výpočty použité při jejich výpočtu jsou podobné. Pro většinu zkoumaných bodů bude platit, že jejich manhattanská vzdálenost od středů shluků je větší než euklidovská, to se ale aplikuje na všechny body. Proto nelze očekávat nějaké zásadní rozdíly při porovnávání výsledků za pomocí těchto dvou metrik. Výpočet se ovšem řídí jiným vzorcem, takže lze rozdíly pozorovat.

Rozdíly výsledků získaných pomocí euklidovské a manhattanské metriky jsou lépe viditelné, pokud jsou body klasifikovány do většího množství shluků. Klasifikace do dvou shluků nemusí jevit žádné změny a klasifikace do tří shluků jen nepatrné (v porovnání s výpočty pomocí euklidovské metriky).



Obrázek 4.6: Rozdíly mezi manhattanskou a euklidovskou metrikou v rovině

Výsledky zobrazené na obr. 4.6 jsou získané pomocí klasifikace se sedmi shluky na datech Místnost 1 (4.2.1). Pro tyto klasifikace byly středy shluků přemístěny do optimálních pozic pomocí algoritmu k-means s maximálním množstvím iterací nastaveným na čtyřicet. Z důvodu podobnosti metrik se středy shluků přemístily z náhodně vygenerovaných míst na stejné pozice.



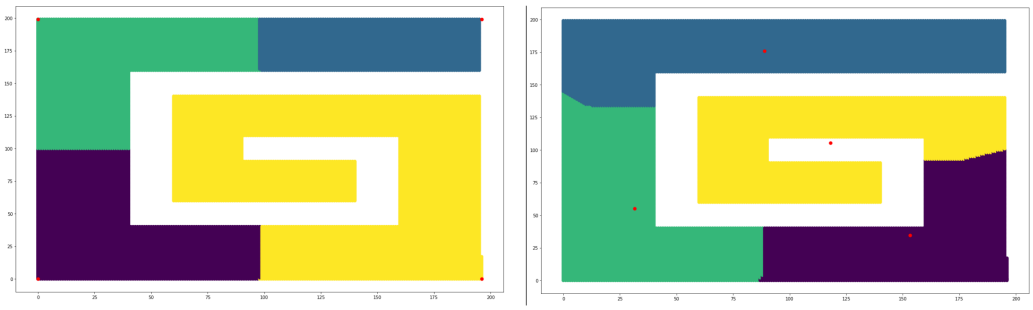
Obrázek 4.7: Rozdíly mezi manhattanskou a euklidovskou metrikou v prostoru

Na obr. 4.7 lze také pozorovat několik odlišností mezi výsledky. Jedná se o výpočet pro data Schod 2 (4.2.1) se třemi středy shluků. Na tato data byl aplikován algoritmus k-means, středy tedy jsou uprostřed svých shluků a maximální počet povolených iterací byl opět nastaven na hodnotu čtyřicet. Výsledek na levé straně byl vypočítán pomocí manhattanské metriky zatímco výsledek na pravé straně pomocí metriky euklidovské. Oblasti, ve kterých lze snadno pozorovat odlišnosti ve výsledku, jsou na obrázku vyznačeny červenou elipsou.

4.3.4 Testování funkčnosti algoritmu k-means

Algoritmus k-means má na starosti přesouvat středy shlukování do těžiště svého shluku. Pro zjištění, zda se středy shluků správně přesouvají, je nejjednodušší zobrazit stav před spuštěním algoritmu a porovnat jej se stavem po jeho ukončení.

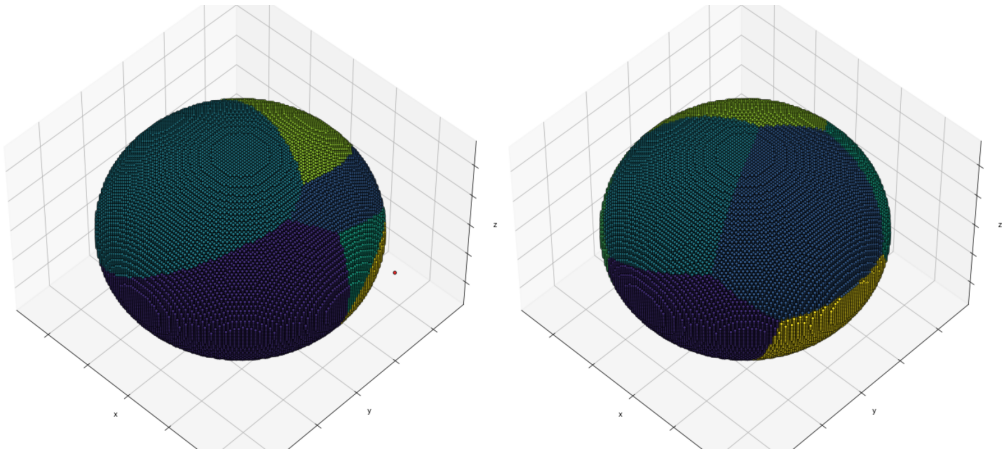
Na obr. 4.8 můžeme vidět dva výsledky. Oba pokusy byly provedeny s daty Místnost 2 (4.2.1) se stejnými parametry s tím rozdílem, že u pokusu na pravé straně neproběhl přesun středů shluků pomocí algoritmu k-means, tedy středy shluků zůstaly tam, kde byly vygenerovány. Naopak u výsledku



Obrázek 4.8: Demonstrace funkčnosti k-means v rovině

po levé straně proběhla celá klasifikace, včetně přemisťování středů a několika iterací algoritmu k-means. Oba pokusy byly spuštěny se čtyřmi středy shluků a geodetickou metrikou.

Jak lze z porovnání vidět, středy shluků se v případě výsledku vpravo všechny poměrně hodně přesunuly ze svých rohů, a utvořily tak přibližně stejně velké shluky. Naopak u výsledku nalevo lze pozorovat na žlutém shluku, jak se geodetická metrika chová u tohoto typu dat.



Obrázek 4.9: Demonstrace funkčnosti k-means v prostoru

Výsledky zobrazené na obr. 4.9 demonstrují výsledky chování k-means na prostorových datech. Při těchto pokusech byla využita data Koule (4.2.1). Pro tyto pokusy bylo vygenerováno osm náhodně rozmístěných středů shluků. Vlevo lze pozorovat výsledek shlukování bez přemisťování středů shluků pomocí k-means. Napravo je vidět výsledek po přemístění středů shluků pomocí algoritmu k-means. Tyto středy shluků měly identickou počáteční polohu jako v předchozím pokusu a maximální počet iterací byl nastaven na hodnotu čtyřicet. Z výsledků lze vyčíst, že po přemístění středů shluků jsou data do shluků rozdělena rovnoměrněji a shluky jsou pravidelnější. Při obou

pokusech byla počítána vzdálenost mezi body pomocí euklidovské metriky.

4.4 Kombinace metrik

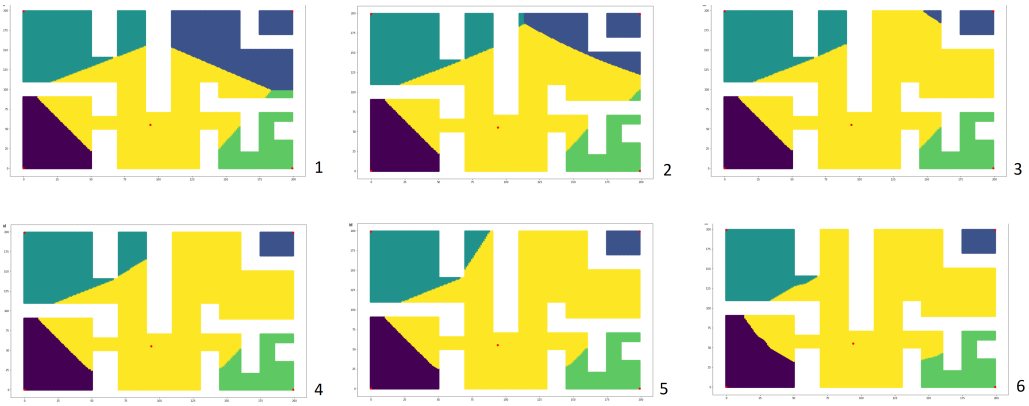
Možnost, která se po otestování naskytla, byla zkusit zkombinovat jednotlivé výsledky. Samozřejmě aby to mělo nějaký smysl, muselo by se jednat o experiment na jednom datasetu s většinou parametrů stejných, např. počet středů shluků. Nabízí se zkusit provést stejný pokus se stejnými daty, se stejným počtem shluků, ale se změněnými metrikami u různých měření a výsledky kombinovat v různém poměru.

Po úspěšném výpočtu jsou do externího souboru uloženy vzdálenosti každého bodu od každého středu shluků. Pro jednodušší představu: do souboru resultE.txt jsou uloženy vzdálenosti každého bodu ke každému středu shluku z posledního pokusu s euklidovskou metrikou. Do souboru resultG jsou uložena obdobná data, ale z pokusu s geodetickou metrikou. Výsledná kombinace vzniká tak, že tato data jsou násobena konstantou určující poměr dané metriky ve výsledku kombinace. Tyto konstanty jsou přepočítány tak, aby jejich součet byl 1. Po tomto přepočítání je každý bod přiřazen k nejbližšímu středu shluku podle nově spočítaných hodnot. Během implementace však vyvstala otázka, jak naložit se středy shluků, protože po obou klasifikacích budou na zcela odlišných místech, tedy kombinovat vzdálenosti k nim nemůže vrátit požadovaný výsledek.

První možností, jak je toto možné implementovat, je uložit si pozice středů po vygenerování (před jejich přesuny) a načíst je při následujících pokusech. Tato možnost by vracela pravděpodobně výsledky nejblíže tomu, co bychom chtěli vidět, ale je velice pomalá a v extrémních případech bychom mohli získávat výsledky, které budou na první pohled špatně (protože se např. středy shluků budou přesouvat opačnými směry z důvodu jiné metriky apod.).

Druhý způsob, který by mohl získat výsledky a zároveň se vyvarovat chyb toho prvního, by bylo neuplatnit přesouvání středů pomocí algoritmu k-means. Středy shluků by musely být zadány nebo vygenerovány na místech, o kterých už předem víme, že k nim chceme shlukovat. Tento způsob by mohl produkovat žádané výsledky, ale museli bychom znát pozice středů shluků před samotným výpočtem.

Na obr. 4.10 jsou zobrazeny výsledky takového pokusu. Pod indexem 1 je výsledek výpočtu využívající euklidovskou metriku. Výpočet byl spuštěn na datech s názvem Místnost 1 (4.2.1) s pěti středy shluků, přičemž čtyři ze středů byly vygenerovány v rozích a pátý byl vygenerován náhodně. Pokus



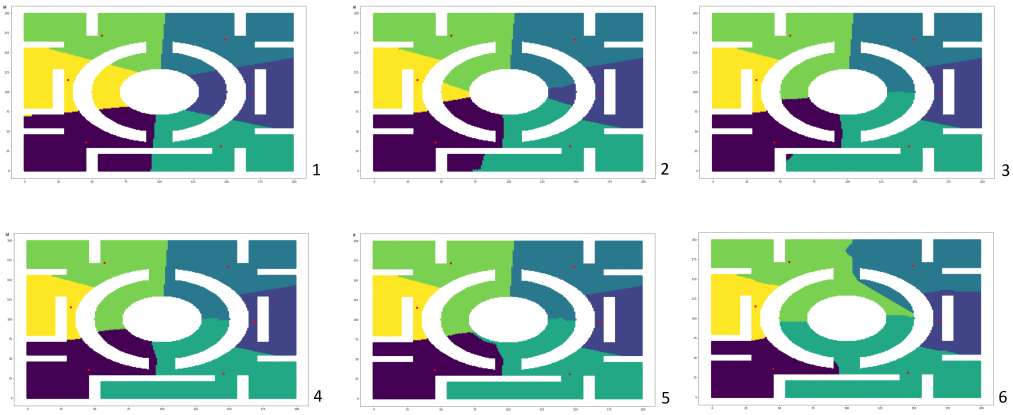
Obrázek 4.10: Kombinace euklidovské a geodetické metriky bez využití k-means

proběhl bez přesunu středů shluků. Stejným způsobem byl spuštěn výpočet, kterým byl získán výsledek pod indexem 6, s tím rozdílem, že tentokrát byly výpočty vzdálenosti počítány geodetickou metrikou a pozice středů shluků nebyly generovány, ale načteny. Zbylé výsledky 2 - 5 byly získány kombinací dat obdržených z těchto pokusů.

Výsledek s indexem 2 byl vytvořen v poměru 1:9, přičemž výsledky výpočtu s geodetickou metrikou byly násobeny konstantou o hodnotě 0,1 a výsledky výpočtu s euklidovskou metrikou byly vynásobeny konstantou o hodnotě 0,9. Výsledek s indexem 3 byl pak vytvořen v poměru 1:4, výsledek s indexem 4 v poměru 1:1 (tedy stejná část výsledku geodetické a stejná část euklidovské metriky) a výsledek s indexem 5 pak v poměru 4:1. Na všech šesti výsledcích je vidět postupná změna metriky, a to nejlépe podle ustupujícího modrého shluku a rozpínajícího se žlutého shluku.

Třetím způsobem je kombinace obou výše zmíněných. Pro získání možnosti kombinace výsledků musíme nejdříve tyto výsledky získat. Můžeme tedy nejprve spustit program s jednou z metrik a nechat algoritmus k-means přesunout středy shluků do optimálních pozic pro určitou metriku. Tyto pozice si můžeme uložit a následně je využít při spuštění dalších pokusů, tentokrát bez využití k-means, s jinými metrikami. Toto řešení není optimální, protože středy shluků nejsou pro většinu výpočtů s různými metrikami na správném místě, výjimkou je jen první výpočet. Rozdíly mezi výsledky jsou ale minimální. Tímto způsobem se také výrazně snížily časové nároky na výpočet a pozice středů shluků nemusí být známa před spuštěním výpočtu. Z těchto důvodů jsem se k tomuto způsobu přiklonil.

Pomocí tohoto algoritmu byly získány výsledky, které jsou zobrazeny na obr. 4.11. Tento výpočet byl proveden na datech Místnost 3 (4.2.1) s šesti



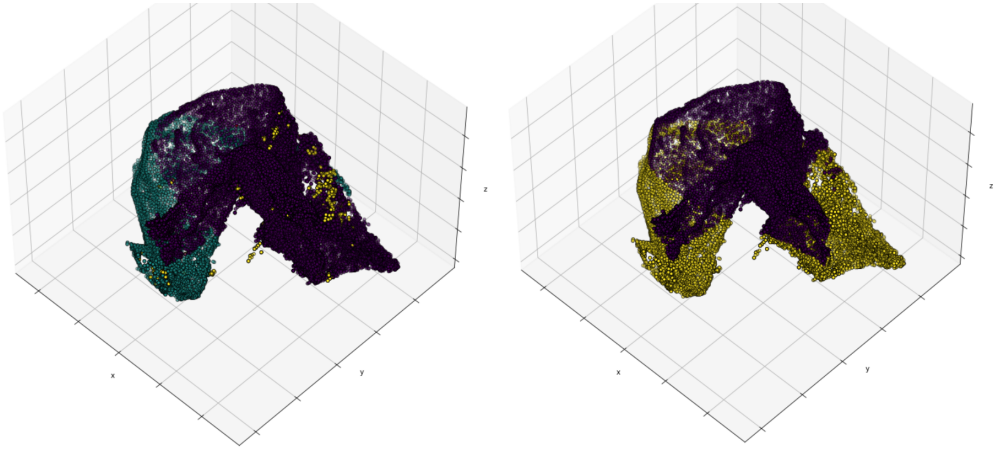
Obrázek 4.11: Kombinace euklidovské a geodetické metriky s využitím k-means pro první z výpočtů

středy shluků. Obdobně jako u předchozího obr. 4.10 odkazuje výsledek s indexem 1 na výpočet s využitím euklidovské metriky a výsledek s indexem 6 na výpočet s využitím geodetické metriky. Při klasifikaci pomocí euklidovské metriky mimo jiné proběhlo i přemisťování středů shluků pomocí k-means. Při klasifikaci s geodetickou metrikou toto už neproběhlo, a proběhla klasifikace ke stejným středům, ke kterým proběhl předchozí výpočet. Na výsledcích 2 - 5 jsou opět vidět výsledky kombinace těchto dvou výpočtů obdobným způsobem jako u předchozího příkladu. Výsledek s indexem 2 byl získán pomocí výsledků klasifikace v poměru 1:4, přičemž výsledky z klasifikace geodetickou metrikou byly vynásobeny konstantou o hodnotě 0,2 a výsledky z klasifikace euklidovskou metrikou byly vynásobeny konstantou o hodnotě 0,8. Podobně pak výsledek s indexem 3 byl vytvořen pomocí výsledků v poměru 2:3, výsledek s indexem 4 byl vytvořen pomocí výsledků v poměru 3:2 a výsledek s indexem 5 byl vytvořen pomocí výsledků v poměru 4:1. Na výsledcích je zajímavé sledovat postupnou ztrátu klasifikace bodů přes prázdná místa v datech a následné rozpínání tyrkysového a světle zeleného shluku.

4.5 Demonstrace funkčnosti metrik na reálných datech

Při práci s reálnými daty, které nejsou nutně rovnoměrné, se může vyskytnout problém s izolovanými body. Tyto body jsou nedosažitelné výpočty využívajícími geodetické metriky, proto jsou označeny jako nedostupné a budou ze shlukování vynechány. Pokud data takové prvky obsahují, výpočet

je výrazně zpomalen z důvodu pokusů o klasifikaci takových bodů.



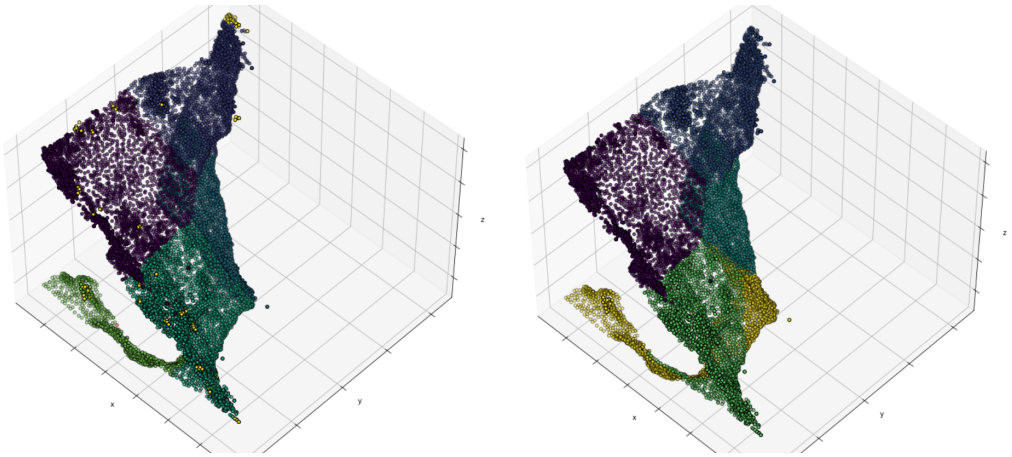
Obrázek 4.12: Rozdíly mezi euklidovskou a geodetickou metrikou na reálných datech

Na obr. 4.12 jsou zobrazeny dva výsledky shlukování provedené na datech s názvem Data 5 (4.2.1). Pro tyto pokusy bylo zvoleno nastavení, kdy je shlukováno ke dvěma středům shluku bez využití algoritmu k-means k přesunu středů shluků. Na levé straně je vidět výsledek shlukování s využitím geodetické metriky k výpočtu vzdálenosti bodu od středů shluků. Body úspěšně přiřazené k nějakému shluku jsou obarveny tyrkysovou nebo fialovou barvou. Prvky vyznačeny barvou žlutou byly klasifikovány jako prvky nedostupné. Na pravé straně je pak pro porovnání vidět, jak vypadá výsledek klasifikovaný pomocí euklidovské metriky. Zde jsou prvky shlukovány opět do dvou shluků a jsou obarvovány žlutou nebo fialovou barvou podle toho, k jakému shluku byly přiřazeny.

Výsledky zobrazené na obr. 4.13 byly získány klasifikacemi dat Data 2 4.2.1 pro pět středů shluků. Na středy shluků byl aplikován algoritmus k-means, tedy byly přesunuty na optimální pozice. Výsledek vlevo byl získán výpočtem využívajícím geodetickou metriku. Jelikož tato data obsahují izolované body, tyto body jsou ve výsledku označeny žlutou barvou. Výsledek vpravo byl získán s použitím euklidovské metriky. I když můžeme pozorovat podobnost v horních částech grafu, zřetelné odlišnosti se vyskytují mezi body níže.

Podobnost v horní části je způsobena tím, že body, které měly být přiřazeny do shluků, jsou v rovině. Na jejím povrchu je malé množství deformací. V takovém případě výpočet s geodetickou metrikou získá velice podobné výsledky jako výpočet s metrikou euklidovskou.

Naprosto odlišné výsledky lze ovšem pozorovat ve spodní části grafu.



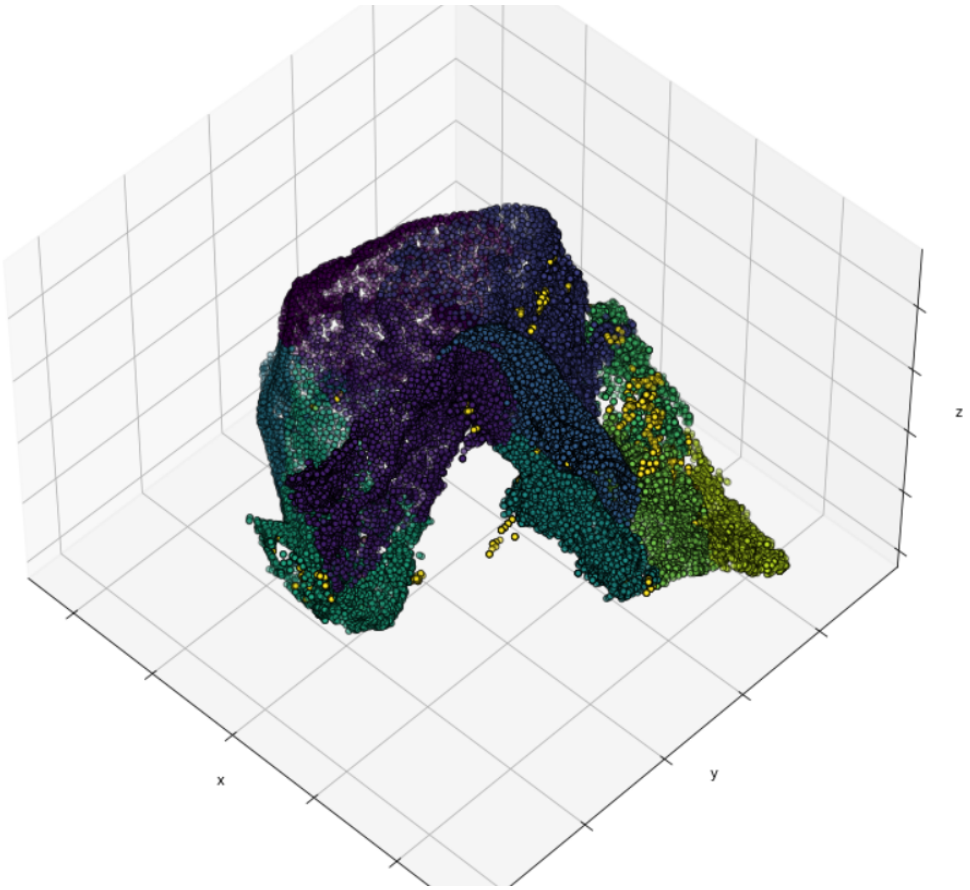
Obrázek 4.13: Rozdíly mezi euklidovskou a geodetickou metrikou na reálných datech s využitím k-means

Zatímco u výsledku výpočtu pomocí euklidovské metriky pozorujeme, že se v datech v těchto místech mísí dva shluky, ve výsledku výpočtu pomocí geodetické metriky tomu tak není. Toto je způsobeno klasifikací bodů ke shluku „vzduchem“, tedy bez vlivu okolních bodů. Jelikož jsou tyto body spojeny se zbytkem jen úzkým přechodem, byly výpočtem s využitím geodetické metriky klasifikovány jako jeden shluk. Výpočet využívající euklidovskou metriku naopak klasifikuje body bez ohledu na podobné okolnosti, a proto můžeme získat i takové výsledky, ve kterých se shluky téměř prolínají.

4.6 Výsledky pokusů na reálných datech

Kvůli vlastnostem některých z dat se některé výpočty protáhly i na několik minut. To je dáno hlavně již zmíněnými izolovanými body. Pokud dataset obsahuje takové body, které nejsou geodetickou metrikou dosažitelné, označí je jako body nedosažitelné nebo izolované. Kvůli opakované snaze těchto bodů dosáhnout a klasifikovat je, nabývá výpočet znatelné časové náročnosti. Toto je ale problém jen geodetické metriky v kombinaci s daty obsahující takové body.

Výpočet výsledku, který je vidět na obr. 4.14, byl uskutečněn na datasetu Data 5 (4.2.1). Zvolená metrika byla geodetická a počet středů shluků deset. Žlutou barvou jsou opět označeny body, které program nebyl schopné klasifikovat. Při tomto výpočtu bylo využito algoritmu k-means a lze vidět podobnost s výsledkem na levé straně obr. 4.12.



Obrázek 4.14: Využití geodetické klasifikace na reálných datech

4.7 Časové nároky na výpočet

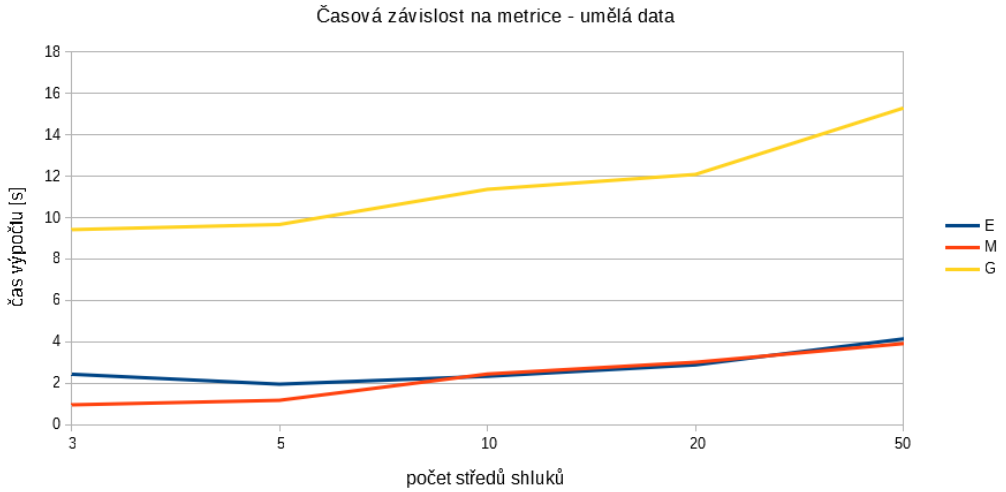
Časy uvedené v tabulce 4.3 jsou jen orientační. Výpočty proběhly pouze jednou a jen na jednom počítači. Slouží pouze k ucelení představy o fungování algoritmu. Ve sloupci „Metrika“ je uveden znak reprezentující metriku použitou při výpočtu. Znak „E“ značí euklidovskou metriku, znak „G“ metriku geodetickou a znak „M“ značí metriku manhattanskou. Do počtu středů shluků, v případě geodetické metriky, nejsou započítány izolované body jako samostatný shluk.

Název dat	Počet bodů	Metrika	Počet středů shluků	Čas výpočtu [sec]
Krychle	60000	E	3	3,43
Krychle	60000	E	5	4,84
Krychle	60000	E	20	5,64
Krychle	60000	M	3	1,67
Krychle	60000	M	5	2,57
Krychle	60000	M	20	5,41
Krychle	60000	G	3	41,03
Krychle	60000	G	5	49,04
Krychle	60000	G	20	45,19
Místnost 3	28227	E	3	2,43
Místnost 3	28227	E	5	1,95
Místnost 3	28227	E	10	2,33
Místnost 3	28227	E	20	2,88
Místnost 3	28227	E	50	4,14
Místnost 3	28227	M	3	0,95
Místnost 3	28227	M	5	1,17
Místnost 3	28227	M	10	2,44
Místnost 3	28227	M	20	3,01
Místnost 3	28227	M	50	3,91
Místnost 3	28227	G	3	9,42
Místnost 3	28227	G	5	9,67
Místnost 3	28227	G	10	11,37
Místnost 3	28227	G	20	12,09
Místnost 3	28227	G	50	15,30

Tabulka 4.3: Měření času v závislosti na datech a parametrech spuštění pro umělá data

V tabulce 4.3 jsou vidět výsledky měření pro data Krychle (4.2.1) a pro data Místnost 3 (4.2.1), jakožto zástupce pro umělá data v prostoru a v rovině. Z výsledků lze vypozorovat, že rozsah datasetu je poměrně důležitý z hlediska časové náročnosti. To, jak počet prvků ovlivňuje časové nároky na výpočet, lze poznat u všech metrik. Největší rozdíl je u geodetické metriky, zatímco rozdíly u manhattanské a euklidovské metriky jsou přibližně stejně velké.

Na grafu 4.15 je zobrazena závislost metriky a počtu shluků na čase výpočtu. Změny mezi manhattanskou a euklidovskou metrikou jsou velice malé nebo zanedbatelné oproti porovnání s metrikou geodetickou. Zvýšením počtu shluků dochází také k prodloužení času výpočtu, ale výrazně méně než



Obrázek 4.15: Graf závislosti metriky a počtu shluků na čase - umělá data

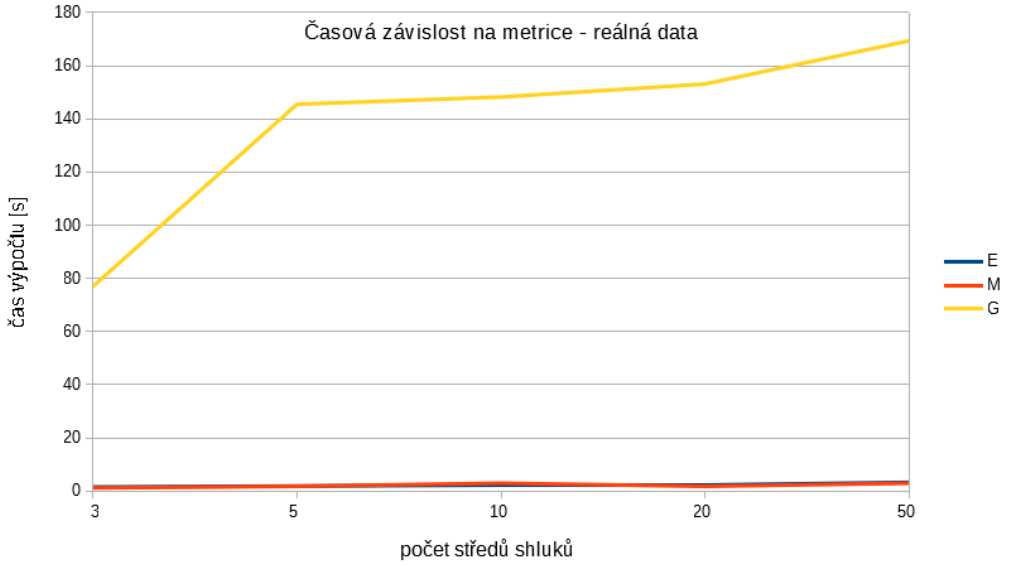
při změně metriky na metriku geodetickou. Dalším zajímavým poznatkem plynoucím z grafu je, že pro všechny metriky se s přibývajícími počty shluků zvyšuje časová náročnost přibližně stejně. Měření bylo prováděno na datech Místnost 3 (4.2.1) a je zaznamenáno v tabulce 4.3. Hodnoty jsou pouze orientační, protože měření probíhalo pouze jednou na jednom počítači.

Název dat	Počet bodů	Metrika	Počet středů shluků	Čas výpočtu [sec]
Data 1	23582	E	3	1,83
Data 1	23582	E	5	1,38
Data 1	23582	E	20	2,76
Data 1	23582	M	3	1,42
Data 1	23582	M	5	1,34
Data 1	23582	M	20	2,81
Data 1	23582	G	3	7,84
Data 1	23582	G	5	7,91
Data 1	23582	G	20	8,74
Data 4	17139	E	3	1,52
Data 4	17139	E	5	1,90
Data 4	17139	E	10	2,06
Data 4	17139	E	20	2,05
Data 4	17139	E	50	3,28
Data 4	17139	M	3	1,19
Data 4	17139	M	5	1,81
Data 4	17139	M	10	2,96
Data 4	17139	M	20	1,78
Data 4	17139	M	50	2,97
Data 4	17139	G	3	76,96
Data 4	17139	G	5	61,17
Data 4	17139	G	10	148,22
Data 4	17139	G	20	153,12
Data 4	17139	G	50	169,34
Data 5	39961	E	3	1,93
Data 5	39961	E	5	2,53
Data 5	39961	E	20	4,00
Data 5	39961	M	3	2,14
Data 5	39961	M	5	1,91
Data 5	39961	M	20	4,17
Data 5	39961	G	3	286,78
Data 5	39961	G	5	1016,86
Data 5	39961	G	20	1120,26

Tabulka 4.4: Měření času v závislosti na datech a parametrech spuštění pro reálná data

Jak je dle naměřených výsledků vidět, časovou náročnost nejvíce ovlivňuje výběr metriky. Samotný výpočet geodetickou metrikou je pak značně ovlivněn existencí izolovaných bodů. Ne tak razantně ovlivňuje čas výpočtu počet bodů v datech a počet středů shluků.

V různých měřeních se může vyskytnout abnormální výsledek z hlediska časové náročnosti. To je způsobeno tím, že středy shluků jsou v datech generovány náhodně a je možné, že se při tomto konkrétním pokusu vygenerovaly na pro algoritmus lepších nebo naopak horších pozicích. To může zapříčinit znatelně méně nebo více iterací potřebných k dosažení výsledku.



Obrázek 4.16: Graf závislosti metriky a počtu shluků na čase - reálná data

Graf na obr. 4.16 ukazuje časovou náročnost výpočtu. Zobrazená data jsou položky z tabulky 4.4 pro Data 4 (4.2.1). Oproti předchozímu grafu z obr. 4.15 je zde mnohem větší rozdíl mezi výpočty s geodetickou a s jinými metrikami. To je způsobeno izolovanými body, které značně zdržely výpočet. Časové rozdíly mezi euklidovskou a manhattanskou metrikou jsou oproti tomu zanedbatelné. Zlom u výpočtu s použitím geodetické metriky a pěti středů shluků je způsoben ukončovací podmínkou, která znemožňuje algoritmu přesáhnout 40 iterací. Pokud by k této podmínce nedošlo, pravděpodobně by čas potřebný pro výpočet stoupal stále stejným tempem, jako je vidět mezi tímto výpočtem a výpočtem se třemi shluky (také pro geodetickou metriku). Kdybychom zkoumali jak by vypadal graf, kdyby v datech nebyly žádné izolované body, tak bychom pravděpodobně dospěli k podobnému výsledku, jaký je vidět na grafu s umělými daty (obr. 4.15).

5 Závěr

V první části věnované zejména teorii byla vysvětlena problematika shlukování, algoritmy užívané v této oblasti a možnosti řešení zadaného problému. Bylo zde popsáno rozdělení metod shlukování, kategorie a příklady jednotlivých metod. Dále se tato část zabývala různými metrikami, stejně jako samotným významem tohoto slova.

V druhé části se práce věnuje zvolenému algoritmu, bližšímu popisu geodetické metriky a implementaci.

Výsledný program splňuje zadané parametry. Lze s jeho pomocí klasifikovat data s několika různými metrikami a výsledky následně porovnat. To je usnadněno možností jejich vizualizace. Takto získané výsledky je také možno kombinovat, a tím získat lepší představu o vlastnostech dat.

Program zpracuje data na základě parametrů, se kterými je spuštěn. Mezi různé parametry spuštění patří cesta k souboru s daty, počet středů shluků, metrika, způsob generace středů shluků a jestli výpočet má využít k-means a přemisťování středů shluků nebo ne. Více informací na toto téma je uvedeno v uživatelské dokumentaci.

Závěrečná část dokumentace se věnuje zejména výsledkům získaných díky tomuto programu a jejich vizualizacím. Krom popisu jednotlivých výsledků je zde i souhrnná statistika časů běhů pro různé parametry spuštění s různými daty a kapitola věnující se kombinaci výsledků.

Program funguje dle očekávání a lze pomocí něj získat požadované výsledky v akceptovatelném čase. V budoucnu je možné jej rozšířit například o další metriky nebo možná i jiný shlukovací algoritmus. Dalším možným rozšířením by mohlo být jednoduché GUI, které by uživateli usnadnilo práci s tímto nástrojem.

Literatura

[1] [online]. . Dostupné z: https://i.stack.imgur.com/7HFxw.jpg.

[2] [online]. . Dostupné z: https://matteucci.faculty.polimi.it/Clustering/tutorial_html/images/image049.jpg.

[3] [online]. . Dostupné z: https://i.stack.imgur.com/c959D.png.

[4] Jian Pei, J. H. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2012.

[5] Bayer, D. I. C. J. *Reálná geodetická data* [online].

[6] Prado, K. S. *How DBSCAN works and why should we use it?* [online]. 2017. Dostupné z: https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80.

[7] Kaas, B. O. *Využití shlukování pro GIS aplikace* [online]. 2013. Dostupné z: https://portal.zcu.cz/StagPortletsJSR168/PagesDispatcherServlet?pp_destElement=%23ssSouboryStudentuDivId_6452&pp_locale=cs&pp_reqType=render&pp_portlet=souboryStudentuPagesPortlet&pp_page=souboryStudentuDownloadPage&pp_nameSpace=G602506&soubidno=20319.

[8] Kaushik, S. *An Introduction to Clustering and different methods of clustering* [online]. 2016. Dostupné z: https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/.

[9] Pedregosa, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, 12, s. 2825–2830.

[10] Priy, S. *Clustering in Machine Learning* [online]. 2020. Dostupné z: https://www.geeksforgeeks.org/clustering-in-machine-learning/.

[11] Vichr, B. J. *Expectation-Maximization Algoritmus* [online]. 2013. Dostupné z: https://is.cuni.cz/webapps/zzp/detail/114194/.

Seznam obrázků

2.1 Ukázka hierarchického shlukování s ukončující podmínkou [3] 5
2.2 Ukázka shlukovaní podle modelu s těžištěm [1] 6
2.3 Ukázka C-means shlukovaní [2] 7
2.4 Ukázka DBSCAN shlukovaní [9] 8

4.1 Umělá data použitá při testování 18
4.2 Reálná data použitá při testování 19
4.3 Test euklidovské metriky na datech Čtverec (4.2.1) 20
4.4 Rozdíly mezi geodetickou a euklidovskou metrikou v rovině . 21
4.5 Rozdíly mezi geodetickou a euklidovskou metrikou v prostoru 22
4.6 Rozdíly mezi manhattanskou a euklidovskou metrikou v rovině 22
4.7 Rozdíly mezi manhattanskou a euklidovskou metrikou v prostoru 23
4.8 Demonstrace funkčnosti k-means v rovině 24
4.9 Demonstrace funkčnosti k-means v prostoru 24
4.10 Kombinace euklidovské a geodetické metriky bez využití k-means 26
4.11 Kombinace euklidovské a geodetické metriky s využitím k-means pro první z výpočtů 27
4.12 Rozdíly mezi euklidovskou a geodetickou metrikou na reálných datech 28
4.13 Rozdíly mezi euklidovskou a geodetickou metrikou na reálných datech s využitím k-means 29
4.14 Využití geodetické klasifikace na reálných datech 30
4.15 Graf závislosti metriky a počtu shluků na čase - umělá data 32
4.16 Graf závislosti metriky a počtu shluků na čase - reálná data 34

A.1 Příklad spuštění s šesti parametry ve správném formátu . . . 42
A.2 Příklad spuštění s šesti parametry v různých formátech a s chybami 43
A.3 Datasety, které lze vygenerovat pomocí Generatoru 46

Seznam tabulek

4.1 Umělá data 17
4.2 Reálná data 19
4.3 Měření času v závislosti na datech a parametrech spuštění pro umělá data 31
4.4 Měření času v závislosti na datech a parametrech spuštění pro reálná data 33

A Uživatelská dokumentace

V odevzdaných materiálech je pět spustitelných programů. Prvním z nich je Classifier.jar, který klasifikuje poskytnuté body podle zadaných parametrů. Jedná se o výsledek této práce, jelikož ostatní programy jsou pouze podpůrné aplikace, které usnadňují a zpříjemňují práci právě s tímto programem. Dalším programem je Combiner.jar, který kombinuje dosažené výsledky. Následuje Generator.jar, který dle zadaných parametrů vygeneruje dataset, který může být zpracován programem Classifier.jar. Posledním .jar programem je SimplifyFile.jar, který dokáže data zmenšit nebo oříznout o přebytečné souřadnice. Pro vizualizaci výsledku je zde poslední spustitelný program Vizualizer.py.

Všechny výše zmíněné programy jsou popsány v kapitolách níže.

Pro spuštění .jar souborů je nutné mít na počítači nainstalovanou a zprovozněnou Javu. Tyto programy byly vyvíjeny ve verzi Java 11.

Pro zprovoznění Vizualizer.py je nutné mít na počítači nainstalovaný Python. Tento program byl vyvíjen ve verzi Python 3.9. Pokud jej nainstalovaný nemáte, lze využít program Vizualizer.exe, který by měl dosáhnout stejných výsledků s tím rozdílem, že nezobrazí interaktivní okna, pouze uloží výsledky jako obrázky do souborů. Pro možnost spuštění tohoto nástroje bude možná potřeba doinstalovat některé knihovny, které jsou při zobrazování použity. Jedná se o pandas, nltk.cluster, matplotlib.pyplot, numpy, mpl_toolkits.mplot3d, math, sys a os. Některé z těchto knihoven můžou být automaticky nainstalovány společně s Pythonem.

A.1 Classifier

Jedná se o hlavní program aplikace. Klasifikuje poskytnuté body podle zadaných parametrů. Povinnými vstupními hodnotami je cesta k souboru s daty a požadovaný počet shluků. Všem parametrům bude vyhrazena kapitola, ve které budou podrobně popsány.

A.1.1 Implementace

Program se skládá ze čtyř .java souborů, přičemž tři z nich podporují hlavní třídu Classifier. V té probíhá většina výpočtů a její implementace je blíže popsána v hlavním dokumentu této bakalářské práce.

Třídy Bod, Mean a Distance jsou převážně obalové třídy s minimální funkčností, kterých třída Classifier využívá ve své implementaci.

A.1.2 Parametry spuštění

Program je spouštěn se dvěma povinnými a až čtyřmi nepovinnými parametry. Povinnými parametry jsou: cesta k souboru, ze kterého bude program čerpat data pro výpočet, a požadovaný počet shluků. Nepovinnými parametry jsou: zvolená metrika pro klasifikaci, pozice generování středů shluků před první iterací, informace o tom, jestli má proběhnout klasifikace pomocí k-means (tedy jestli se mají středy shluků přemisťovat), a maximální povolený počet iterací.

Pokud je program spuštěn s méně než dvěma parametry, je ukončen s chybovou hláškou „Not enough arguments...“. Pokud je program spuštěn se dvěma a více parametry, jsou tyto parametry zkontrolovány a v případě, že byly zadány správně, jsou nastaveny. Nepovinné parametry, které nebyly zadány nebo byly zadány ve špatném formátu, jsou ignorovány a místo nich je nastavena základní hodnota. Pokud je zadáno více než šest parametrů, je bráno v potaz pouze prvních šest a zbytek je ignorován.

Parametry spuštění musí být zadány v následujícím pořadí: cesta k souboru, požadovaný počet shluků, zvolená metrika pro klasifikaci, pozice generování středů shluků před první iterací, informace o tom, jestli má proběhnout klasifikace pomocí k-means (tedy jestli se mají středy shluků přemisťovat), a maximální povolený počet iterací. Pokud chceme spustit program s nějakými parametry, je nutné zadat i parametry předchozí.

Cesta k souboru Jedná se o první parametr potřebný a povinný pro spuštění výpočtu. Je možné zadat cestu k souboru absolutní i relativní cestou. Z důvodu implementace jsou přijímány pouze soubory s koncovkou .txt. Pokud Váš soubor s daty obsahuje jinou koncovku, ale je ve správném formátu, zkuste doplnit .txt jako jeho druhou koncovku. Pokud soubor nelze otevřít nebo najít, je vypsána chybová hláška oznamující tuto skutečnost.

Soubor poskytovaný programu musí být v požadovaném formátu a koncovkou .txt, jinak dojde k chybě v jeho nalezení. Kvůli vlastnostem poskytnutých souborů s daty je ignorována první řádka souboru. Na každé další řádce souboru je nutné mít tři přirozená nebo reálná čísla oddělená mezerou, která reprezentují pozici bodu v prostoru. Poslední řádka souboru je prázdná.

Počet středů shluků Jde o celé číslo reprezentující počet středů shluků vygenerovaných před první iterací. V případě zadání záporného čísla je použita jeho absolutní hodnota. V případě zadání desetinného čísla nebo řetězce je program ukončen s chybovou hláškou „Second argument ...“.

Metrika Je možné zadat třetí argument, tedy zvolit, jaká metrika bude při výpočtu použita. Možnostmi pro euklidovskou metriku jsou „0“, „E“ a „e“. Tato metrika je také výchozí, pokud bude tento parametr zadán chybně nebo zadán nebude. Pro klasifikaci s geodetickou metrikou je možné zadat „1“, „G“ nebo „g“ a pro klasifikaci s metrikou manhattanskou jsou možnosti „2“, „M“ a „m“. Pokud je zadáno něco jiného, program vypíše chybovou hlášku a zvolí euklidovskou metriku.

Pozice středů shluků před první iterací Čtvrtý argument celkem a zároveň druhý nepovinný určuje výchozí pozici, na které budou vygenerovány středy shluků. První možností je náhodná pozice, která je také výchozí pozicí, pokud je parametr zadán chybně nebo není zadán vůbec. Možnosti parametru jsou „0“, „R“ a „r“.

Druhou možností je programu poskytnout soubor s pozicí středů shluku zadáním parametru „1“, „K“ nebo „k“. Program bude hledat soubor se stejným názvem, jako je soubor s body, ale se značkou na konci názvu „_means“. Takový soubor je generován po každém běhu programu, takže lze takto uschovat a znovu použít pozici středů shluků pro více běhů. Při použití této možnosti je automaticky nastaveno, že neproběhne přemisťování středů. Tomu se nelze vyhnout ani následujícím parametrem vstupu. Jedná se o bezpečnostní prvek, který lze změnit jen přímo v kódu třídy (odebráním řádky 366).

Třetí možností je „2“, „P“ nebo „p“. Při využití této možnosti rozdělujeme soubor s body na přibližně stejné části a z každé části získáme informaci o bodu, který je popsán uprostřed. Na této pozici vygenerujeme jeden ze středů shluků. Jinými slovy, generované pozice budou vždy mezi body, nikoliv náhodně vygenerované někde v prostoru.

Poslední možností je vygenerovat středy shluků v rozích datasetu. Pro tento výsledek použijeme „3“, „C“ nebo „c“. Nejprve jsou preferovány protilehlé rohy. Pokud je generováno více středů shluků, jsou vygenerovány na náhodných pozicích.

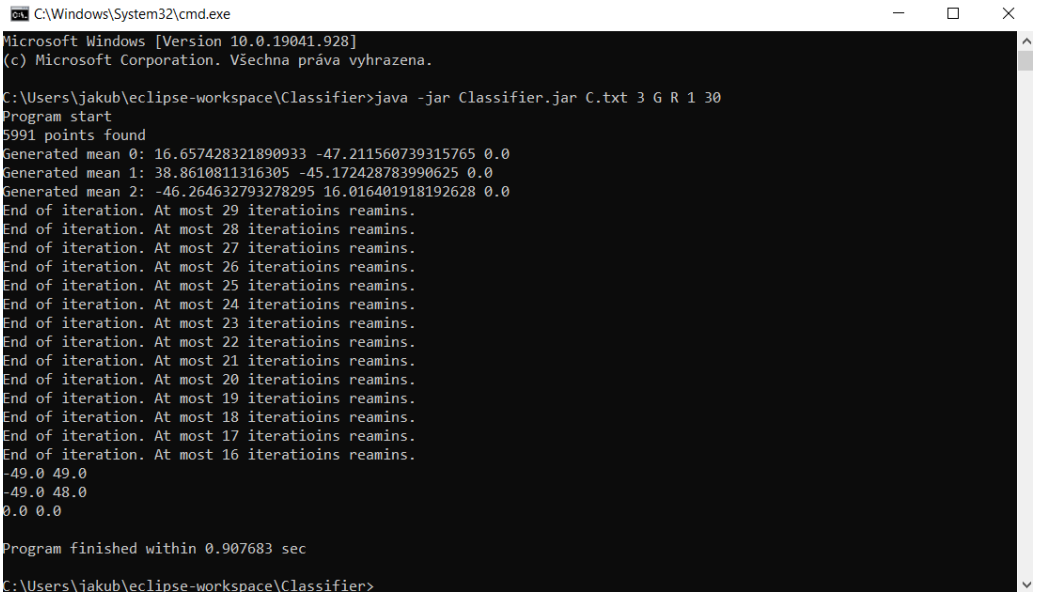
Povolení k přemisťování středů shluku Nastavení na „0“, pokud zakazujeme programu přemisťovat středy shluku, nebo naopak „1“, pokud

to chceme povolit. Pokud je zadáno něco jiného, je automaticky nastavena hodnota na „1“, tedy povoleno.

Maximální počet iterací Pro tento parametr je požadováno celé číslo, které určuje maximální počet iterací k-means v případě, že je předchozí parametr nastaven na „1“, tedy je povolen pohyb středů shluků. Pokud je tento parametr zadán jako záporné číslo, je získána jeho absolutní hodnota. Pokud se jedná o desetinné číslo nebo řetězec, je vypsána chybová hláška a je nastaven na výchozí hodnotu, kterou je 40.

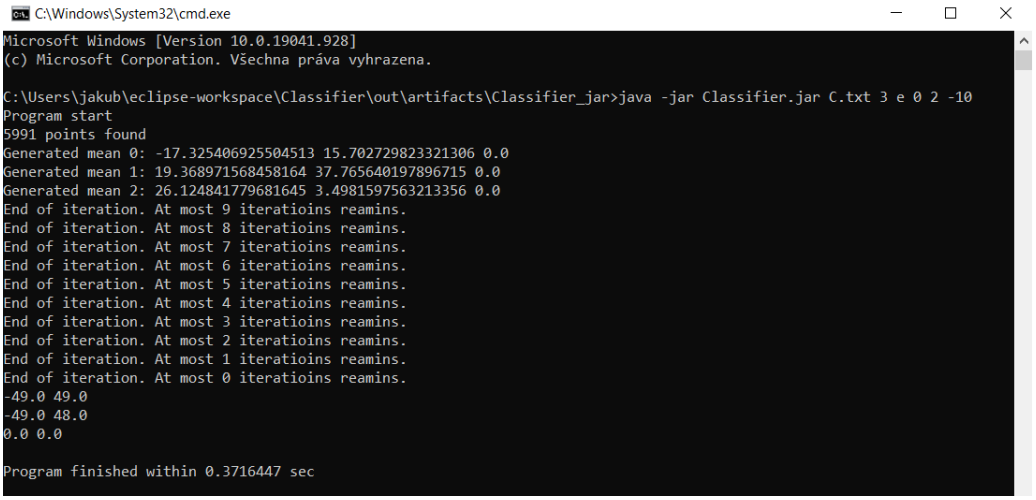
A.1.3 Příklady spuštění

Prvním příkladem spuštění je spuštění se všemi šesti parametry ve správném formátu „java -jar Classifier.jar C.txt 3 G R 1 30“. V tomto případě jsou informace o bodech uloženy v soubooru C.txt, který je ve stejné složce jako .jar soubor. Počet středů shluků je nastaven na tři, metrika na geodetickou, pozice středů shluků při generování je náhodná, středy jsou přemisťovány pomocí algoritmu k-means a maximální počet iterací je 30. Získaný výstup je zobrazen na obr. A.1. Je zde popsán průběh výpočtu. Počet načtených bodů, pozice, na kterých byly vygenerovány středy shluků před první iterací, a hlášení o ukončení jednotlivých iterací. Nakonec je vypsán rozsah datasetu a doba trvání běhu programu. Pokud je program spuštěn bez některých parametrů, výpis je ve stejném formátu.



Obrázek A.1: Příklad spuštění s šesti parametry ve správném formátu

Následuje příklad zadání s několika chybnými parametry, stejně jako zadání parametrů v různém formátu: „java -jar Classifier.jar C.txt 3 e 0 2 -10“. V tomto případě jsou oba povinné parametry zadány správně, metrika je zadána správně malým písmenem, pozice pro generování středů shluků je zadána správně číslicí, možnost pohybu se středy je zadána špatnou číslicí, a je tedy nastavena základní hodnota - povoleno, počet iterací je nastaven na -10, a je tedy použita absolutní hodnota tohoto čísla.



Obrázek A.2: Příklad spuštění s šesti parametry v různých formátech a s chybami

Posledním příkladem spuštění je spuštění s chybou v povinných parametrech. Pokud je program spuštěn následujícím způsobem „java -jar Classifier.jar C1.txt 3 G R 1 30“, přičemž C1.txt není správná cesta k souboru, program skončí chybou a vypíše chybovou hlášku „Couldnt’t find your file.“

A.1.4 Výstupní soubory

Po úspěšném výpočtu jsou ve stejné složce, ve které je soubor s informacemi o bodech, vytvořeny celkem tři soubory. Prvním je soubor s výsledkem klasifikace. Každý bod je zde popsán číslem shluku, ke kterému náleží dle výsledku klasifikace. Tento soubor se jmenuje stejně jako soubor s klasifikovanými body, jenom má za názvem dodatek „_labels“. Druhý soubor obsahuje pozice středů shluků po poslední iteraci výpočtu ve stejném formátu, jako je soubor s informacemi o klasifikovaných bodech. Lze jej poznat podle dodatku za názvem „_means“. Poslední soubor, který je vytvořen po běhu programu, obsahuje vzdálenost každého bodu ke každému středu shluků. Lze jej poznat tak, že má dodatek za názvem „_E“, „_M“ nebo

„_G“ podle toho, s jakou metrikou byl výpočet prováděn.

První z těchto souborů je výsledek klasifikace a je potřeba při zobrazení výsledků pomocí programu Vizualizer. Druhý ze souborů je potřeba také pro zobrazení výsledku pomocí programu Vizualizer, ale je také potřeba, pokud bychom chtěli spustit program se čtvrtým parametrem jako „1“, „K“ nebo „k“. Třetí ze souborů je nutný pro fungování Combineru pro kombinování výsledků.

A.2 Combiner

Tento program kombinuje výsledky dosažené výpočty prvního programu. Ke správnému běhu programu je nutné mít potřebné výsledky výpočtů, které chceme zkombinovat.

Program je spouštěn podobným způsobem jako Classifier.jar se čtyřmi povinnými parametry. Prvním je cesta k souboru s klasifikovanými prvky. Tento soubor sice nebude použit, ale soubory s dodatky „_E“, apod. budou. Dalšími třemi parametry jsou konstanty, které určují, v jakém poměru kombinace proběhne. Zadávají se jako reálná čísla a v programu jsou následně normalizovány. Jsou zadávány v následujícím pořadí: první je podíl euklidovské metriky, druhý je podíl manhattanské metriky a třetí je podíl geodetické metriky. Pokud je některá z hodnot nulová, není nutné mít odpovídající soubor s výsledky.

Pokud je program spuštěn správně a výpočet proběhne, program informuje uživatele, že parametry úspěšně přijal a do úspěšném výpočtu se ukončí. Příkladem takového spuštění je např. „java -jar Combiner.jar C.txt 0.5 0 0.9“, kdy ve stejném adresáři jako je Combiner.jar jsou i soubory C.txt, C_labels.txt, C_E.txt a C_G.txt ve správném formátu a se správnými daty. Pokud by nějaký soubor chyběl, argumenty by byly zadány chybně nebo se stal nějaký jiný problém zabraňující úspěšnému výpočtu, program informuje uživatele o chybě a ukončí proces.

A.2.1 Výstupní soubory

Výstupem programu je zapsání výsledků do již pravděpodobně existujícího souboru s dodatkem „_labels“. Je tomu z toho důvodu, aby nevznikaly komplikace při zobrazování výsledku pomocí Vizualizeru, protože budou ve stejném formátu i souboru, jako kdyby proběhla regulerní klasifikace pomocí programu Classifier.

A.3 Generator

Program je schopný vygenerovat až třináct různých datasetů pro hlavní program Classifier.jar. Je spouštěn se dvěma až čtyřmi argumenty podle toho, jaká data chceme generovat. Výstupní soubor tohoto programu je možné využít při klasifikaci programem Classifier.jar.

A.3.1 Parametry spuštění

Prvním parametrem spuštění je cesta k souboru, který má být vytvořen k zápisu výsledku generování. Druhým parametrem je číslo udávající, jaká data se budou generovat. Podle toho, jaká metoda je na základě toho volána, jsou potřeba další (žádný až dva) parametry.

Na obr. A.3 jsou zobrazeny všechny tvary, kterých lze díky tomuto nástroji dosáhnout. V levém dolním rohu každého datasetu je uvedeno číslo. Když je zadáno jako druhý argument programu, tak jsou vygenerována data toho určitého typu. Např. při zadání druhého parametru „1“ jsou vygenerována data do čtverce.

Pro tvary 1 - 10 je potřeba zadat jeden (třetí) celočíselný parametr určující většinou velikost podstavy (v bodech) nebo velikost hrany opsané krychle. Výjimkou je tvar 6 (Válec), který přijímá dva parametry, a sice průměr podstavy a výšku. Tvary 11 - 13 nevyžadují žádný parametr. Pokud parametry nejsou zadány, je vybrána výchozí hodnota (většinou 100).

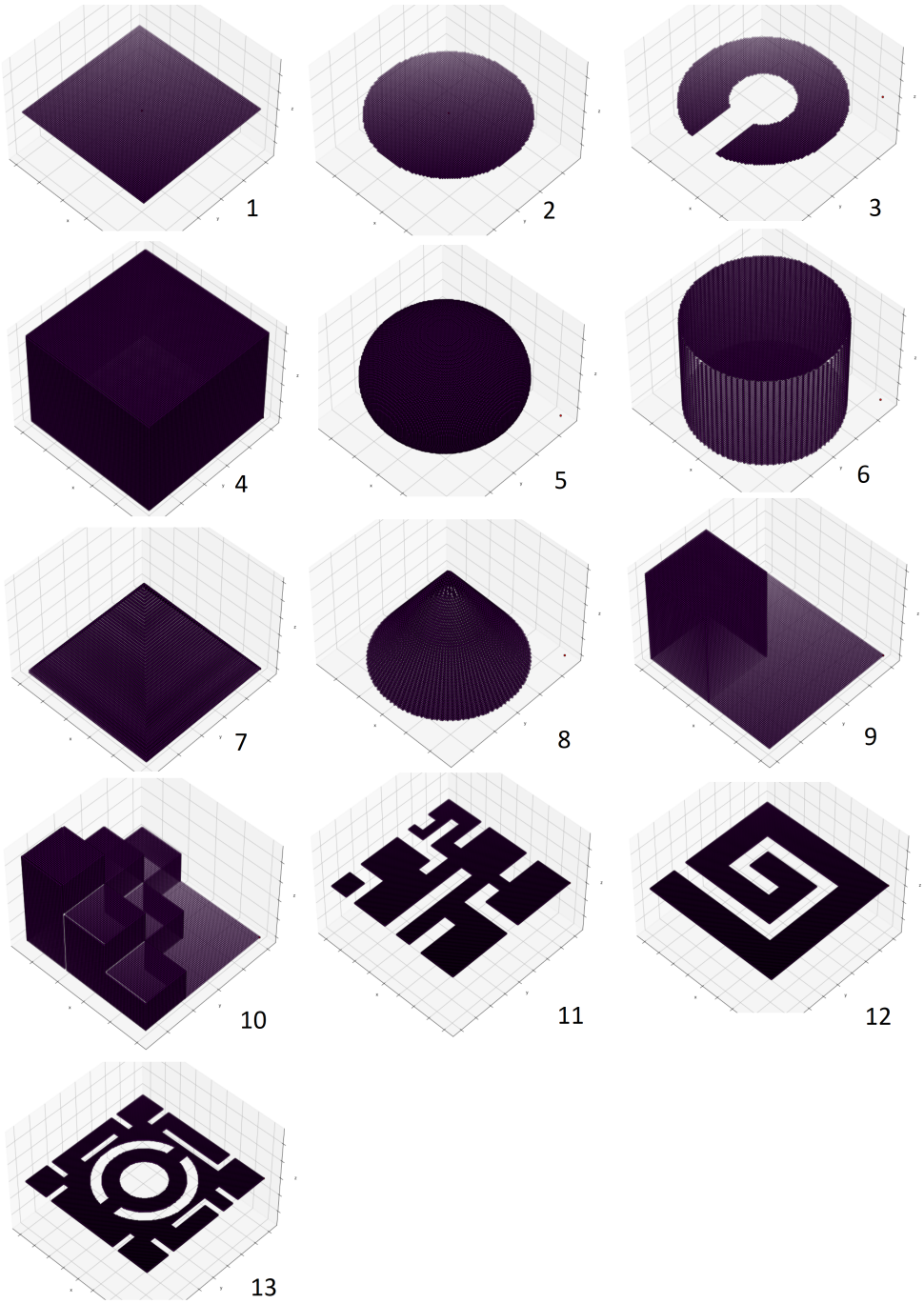
Příkladem spuštění může být například „java -jar Generator.jar Ctverec.txt 1 100“. Výstupem je pak dataset, který je na obr. A.3 označen číslem 1.

A.3.2 Výstupní soubory

Výstupem Generatoru je jeden soubor s příponou .txt, jehož cesta odpovídá tomu, co bylo zadáno jako první parametr. Tento soubor odpovídá formátem tomu, co zpracovává program Classifier.jar.

A.4 SimplifyFile

Tento program pomáhá zjednodušit soubor s body, který je poskytnut Classifieru ke klasifikaci. Je spouštěn se třemi parametry, přičemž poslední parametr je nepovinný. Prvním parametrem je cesta k souboru, který je nutné upravit. Druhým parametrem je způsob, jakým bude soubor upraven.



Obrázek A.3: Datasety, které lze vygenerovat pomocí Generatoru

Hodnotou „0“ spouštíme program ke snížení velikosti dimenze bodů (např. když jsou popsány i jinými hodnotami než pozicí v prostoru) na velikost tři. Třetím (nepovinným) argumentem v tomto případě je oddělovací znak mezi jednotlivými prvky dimenze. Pokud tento parametr není zadán,

je příslušná hodnota nastavena na mezeru. Výstupem tohoto zpracování je soubor s příponou .txt, který má oproti poskytnutému souboru dodatek „_trimmed“.

Hodnotou „1“ je program spuštěn ke snížení počtu bodů v datasetu. Podle hodnoty třetího (nepovinného) argumentu jsou rovnoměrně odebírány body ze souboru. Tento parametr je celočíselný a určuje, kolikátý každý prvek v datasetu zůstane. Tedy velikost výsledného souboru bude původní velikost dělená touto konstantou. Výstupem tohoto zpracování je soubor s příponou .txt, který má oproti poskytnutému souboru dodatek „_small“.

A.5 Vizualizer

Vizualizer slouží k zobrazení výsledků, které jsme obdrželi z programu Classifier.jar, případně Combiner.jar. Program je spouštěn s jedním parametrem, a sice cestou k souboru s body. Po spuštění jsou otevřena dvě okna. V jednom z nich je výsledek zobrazení v rovině a ve druhém je výsledek zobrazení v prostoru. Po zavření těchto oken je výsledek uložen do souborů s koncovkou .png.

Aby program mohl proběhnout správně, je potřeba mít krom souboru s body ve stejné složce také výsledky výpočtu k tomuto souboru, tedy soubory s dodatkem „_labels“ a „_means“.