FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

| | |
|---|---|
| **Název:** | Spolupráce studentů různé úrovně znalostí |
| **Student:** | Radomír Žemlička |
| **Vedoucí:** | Ing. Magda Friedjungová |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Znalostní inženýrství |
| **Katedra:** | Katedra teoretické informatiky |
| **Platnost zadání:** | Do konce letního semestru 2018/19 |

## Pokyny pro vypracování

Student má k dispozici data ohledně úspešnosti jednotlivých studentů středních škol při řešení matematických úloh. Tyto studenty je nutné rozřadit do skupin tak, aby v každé skupině byli žáci různé úrovně znalostí a skupina zároveň dosahovala co nejlepších výsledků při řešení skupinových úloh.

1) Proveďte rešerši dostupných metod v oblasti data miningu a strojového učení pro tvorbu skupin studentů.
2) Analyzujte dostupná data.
3) Zvolte alespoň 3 metody, které implementujete a otestujete na dostupných datech.
4) Vyhodnoťte úspěšnost jednotlivých metod.

## Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 6. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Spolupráce studentů různé úrovně znalostí

***Radomír Žemlička***

Katedra teoretické informatiky
Vedoucí práce: Ing. Magda Friedjungová

14. května 2018

# Poděkování

Rád bych tímto poděkoval Ing. Magdě Friedjungové za čas, který strávila vedením mé práce, a za její cenné rady. Dále bych chtěl poděkovat společnosti TECHAMBITION LTD za poskytnutá data a za možnost testovat mé řešení v jejich systému. A na závěr bych chtěl poděkovat své přítelkyni, která mi byla a stále je neskutečnou oporou.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 14. května 2018 ……

### Odkaz na tuto práci

Žemlička, Radomír. *Spolupráce studentů různé úrovně znalostí*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

# Abstrakt

V bakalářské práci se zabývám vytvářením vyvážených skupin studentů pro účely skupinové výuky na základě jejich výsledků v individuálních lekcích. Práce obsahuje přehled současných řešení pro vytváření skupin studentů společně a zároveň popis různých metod pro shlukování dat. Součástí práce je také analýza dostupných dat, implementace některých metod a jejich porovnání. Zvolil jsem konkrétně metody *K-means*, *DBSCAN*, *Hopfield network* a kombinaci *Self-organizing map* a aglomerativního hierarchického shlukování. V rámci práce jsem vytvořil knihovnu pro automatickou tvorbu skupin studentů, kterou je možné zabudovat do e-learningových systémů, které mají požadovanou strukturu dat.

**Klíčová slova** systém pro vytváření skupin studentů, educational data mining, skupinová výuka, analýza chování, různé úrovně znalostí, vyvážené skupiny, shluková analýza, e-learning, Node.js

# Abstract

In this thesis I deal with the creation of balanced groups of students for collaborative learning based on their results in individual lessons. The thesis also consists of an analysis of available data, implementation of several methods and their comparsion. I have chosen four specific methods: K-means, DBSCAN, Hopfield network and a combination of Self-organizing map and agglomerative hierarchical clustering. One of the results of this thesis is a library for the automatic creation of student groups, which can be embedded into e-learning systems that have the required data structure.

**Keywords** system for creating student groups, educational data mining, collaborative learning, behaviour analysis, different levels of knowledge, balanced groups, cluster analysis, e-learning, Node.js

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

Skupinová výuka je v moderním školství obrovským trendem a začíná se úspěšně prosazovat i v České republice, a to převážně díky rozmachu internetu a všemožných e-learningových programů. Učitel je v tu chvíli postaven do role moderátora a pouze vede studenty k tomu, aby si navzájem předávali znalosti a společnými silami řešili zadané problémy.

Toto téma jsem si vybral, protože v automatizaci vytváření skupin studentů vidím obří výzvu. Je potřeba spojovat studenty se stejnými učebními styly, přičemž jsou dostupné pouze omezené informace. A právě na jejich základě je nutné odvodit vzorce chování jednotlivých studentů. Jedná se navíc o velice měkké téma, kde velikou roli hraje lidský faktor.

Výsledek této práce je určen pro tvůrce elektronických výukových nástrojů. Zároveň je určen i pro studenty a učitele na školách, kde se tyto nástroje využívají a kde chtějí zautomatizovat tvorbu skupin na základě chování a úspěšnosti studentů.

V práci se zabývám analýzou dostupných metod a řešení pro tvorbu skupin studentů. Následně analyzuji dostupná data a vyvozuji z nich užitečné atributy. Poté popisuji implementaci některých ze zmíněných metod a vyhodnocuji jejich úspěšnost.

Kapitola 1

# Cíl práce

Cílem rešeršní části práce je zmapovat dostupné metody v oboru data miningu a strojového učení pro tvorbu skupin studentů, dále je popsat a zhodnotit jejich klady a zápory.

Cílem praktické části je zanalyzovat dostupná data a rozhodnout, jakým způsobem je zpracovat. Poté implementovat alespoň tři metody pro vytváření skupin uživatelů, vyzkoušet je na dostupných datech a zároveň vyhodnotit jejich úspěšnost. Fáze zpracování dat a implementace jednotlivých metod jsou realizovány ve formě knihovny, kterou lze zabudovat do existujícího e-learningového systému a která zajistí spolehlivé vytváření vyvážených skupin uživatelů, které budou dosahovat dobrých výsledků.

Kapitola 2

# Rešerše

Pro pochopení dané problematiky je nejprve nutné popsat základní principy práce s daty. V této kapitole se zaobírám různými metodami pro shlukování dat, dále možnými postupy pro detekci outlierů (neboli odlehlých hodnot), poté způsoby, jakými lze zjistit kvalitu jednotlivých shluků, a na závěr popisuji současná řešení pro tvorbu skupin studentů včetně toho, jaké shlukovací metody využívají.

## 2.1 Shlukování dat

Shlukování je proces, který vytváří skupiny dat podle zadaného kritéria. Ve většině případů je toto kritérium stanoveno jako vzájemná podobnost či nepodobnost jednotlivých záznamů. Cílem je tedy vytvořit homogenní shluky, což znamená, že jsou si data v nich co nejvíce podobná. [1, s. 322]

Obecně je jednodušší použití nepodobnosti, která je často definována jako vzdálenost (značena $\delta$) mezi dvěma prvky $a$ a $b$. [1, s. 322] Ta musí splňovat následující podmínky:

**identita** $(\forall a)(\forall b)(\delta(a,b) = 0 \iff a = b)$

**nezápornost** $(\forall a)(\forall b)(\delta(a,b) \geqslant 0)$

**symetrie** $(\forall a)(\forall b)(\delta(a,b) = \delta(b,a))$

**trojúhelníková nerovnost** $(\forall a)(\forall b)(\forall c)(\delta(a,b) \leqslant \delta(a,c) + \delta(c,b))$

Nechť $p$ je počet datových atributů a $x_{ai}$ je hodnotou atributu $i$ záznamu $a$. Potom lze vzdálenost definovat různými způsoby.

Pro numerická data se používá např. euklidovská vzdálenost [1, s. 323], která je definována jako druhá odmocnina součtu čtverců odchylek a jedná se v principu o zobecnění tzv. Minkowského vzdálenosti [1, s. 322]. Pro záznamy o $p$ atributech je tedy definována následovně:

$$\delta(a,b) = \sqrt{\sum_{i}^{p}(x_{ai} - x_{bi})^2} \tag{2.1}$$

Dále stojí za zmínku Hammingova (manhattanská) vzdálenost a Čebyševova vzdálenost, nicméně existuje ještě řada jiných. [1, s. 323]

Takto určená vzdálenost má však tu nevýhodu, že může být velice jednoduše ovlivněna různými rozsahy a jednotkami atributů. Mohla by jim totiž přikládat menší, či větší váhu. Máme-li normálně rozdělená data, je možné využít standardizace pomocí tzv. z-skóre [2], [3]. Pro prvek $a$ je hodnota jeho $i$-tého atributu ($x_{ai}$) standardizována pomocí následujícího vzorce:

$$z_{ai} = \frac{x_{ai} - \bar{x}_i}{s_i} \tag{2.2}$$

$$\bar{x}_i = \frac{1}{n}\sum_{q}^{n} x_{qi} \tag{2.3}$$

$$s_i = \sqrt{\frac{1}{n}\sum_{q}^{n}(x_{qi} - \bar{x}_i)^2} \tag{2.4}$$

Takto standardizované hodnoty budou mít střední hodnotu 0 a rozptyl 1.

Jinou možností, jak tento problém řešit, je tzv. min-max normalizace [2], [3]. Ta se běžné používá pro atributy s různými rozsahy, které ale nemají normální rozdělení nebo obsahují outliery. Min-max normalizace převede všechny hodnoty do pevně daného intervalu, většinou $[0, 1]$ či $[-1, 1]$. Hodnoty jsou normalizovány dle následujícího vzorce:

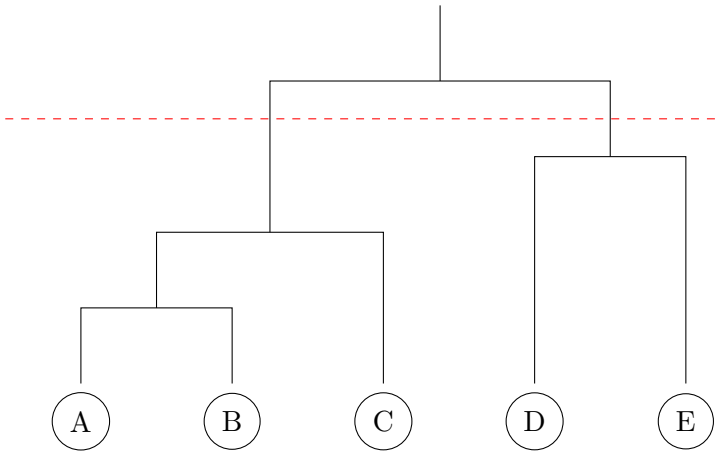
$$z_{ai} = \frac{x_{ai} - \min\{x_i\}}{\max\{x_i\} - \min\{x_i\}} \tag{2.5}$$

V dalších podkapitolách jsou popsány jednotlivé kategorie shlukovacích metod, které využívají zde popsané způsoby určování vzdálenosti.

### 2.1.1 Hierarchické shlukování

První kategorií je hierarchické shlukování, u kterého dle [1, s. 330–331] rozlišujeme dva základní přístupy:

1. *Agglomerative* (česky aglomerativní či seskupující), jinak také *bottom up* (česky zdola nahoru) — začíná s jednotlivými prvky, které jsou označeny jako prvotní shluky. Následně jsou v každé iteraci právě dva nejbližší shluky spojeny do jednoho.




Obrázek 2.1: Ukázka dendrogramu s řezem

2. *Divisive* (česky divizní či rozdělovací), jinak také *top down* (česky shora dolů) — začíná s jedním velkým shlukem, který je postupně v jednotlivých iteracích rozdělován na menší a menší.

U obou typů hierarchického shlukování vzniká tzv. dendrogram (viz obrázek 2.1), ze kterého je vidět, v jakém pořadí byly jednotlivé prvky pospojovány (v případě aglomerativního přístupu) nebo odděleny (v případě divizního přístupu). Zvolená hladina řezu (vyznačená čárkovaně) potom určuje výsledný počet shluků. [1, s. 331]

Při spojování jednotlivých shluků se pro určení vzdálenosti mezi dvěma shluky používá tzv. *linkage criterion* (česky propojovací kritérium). To využívá metriku pro určení vzdálenosti mezi dvěma prvky ($\delta$), např. euklidovskou. Vždy jsou spojeny takové dva shluky, pro které je hodnota použitého kritéria nejnižší. Mezi běžně používaná kritéria dle [1, s. 331] patří:

***complete-linkage*** $\max\{\delta(a,b) : \forall a \in A, \forall b \in B\}$

***single-linkage*** $\min\{\delta(a,b) : \forall a \in A, \forall b \in B\}$

***average-linkage*** $\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B}\delta(a,b)$

Aglomerativní přístup využívající popsané postupy a kritéria je často označován jako *AGNES (Agglomerative Nesting)*. Na druhé straně divizní přístup, který oproti *AGNES* postupuje reverzně, je běžně označován jako *DIANA (Divisive Analysis Clustering)*. [4]

### 2.1.2 Shlukování založené na centroidech

Další kategorií je tzv. *Centroid-based clustering* neboli shlukování založené na centroidech. V tomto případě jsou shluky reprezentovány pomocí tzv. cen-

troidů, které nutně nemusí být součástí shlukovaných dat. Příslušnost jednotlivých záznamů do konkrétního shluku je potom určena podle nejbližšího centroidu.

Běžně používaným algoritmem z této kategorie je *K-means* [5], [6, s. 424–428], [1, s. 333]. Ten vyžaduje předem zadaný počet shluků ($K$), což je z určitého hlediska jeho nevýhoda. Existují však metody, které s výběrem vhodné hodnoty $K$ mohou pomoci. Jednou z nich je např. metoda *Elbow* [7]. Jednotlivé kroky algoritmu *K-means* dle [1, s. 333] jsou popsány zde:

1. Na začátku náhodně vygeneruj souřadnice $K$ centroidů.
2. Rozřaď všechny záznamy do shluků podle toho, jaký centroid je danému záznamu nejbližší.
3. Vypočítej nové souřadnice centroidů jako průměr souřadnic všech záznamů v daném shluku.
4. Kroky 2 a 3 opakuj do té chvíle, kdy není žádný záznam přesunut z jednoho shluku do druhého.

Je garantováno, že algoritmus skončí v nějakém lokálním optimu. Ne vždy se ovšem toto lokální optimum shoduje s optimem globálním. [8] Je taktéž nutné dodat, že způsob, jakým se vypočítávají nové centroidy (průměr souřadnic záznamů ve shluku), může být velice jednoduše ovlivněn outliery v datech, což je asi největší nevýhoda tohoto algoritmu. Před jeho aplikací je tedy nutné data pečlivě prozkoumat a při velkém množství outlierů je pročistit.

Vzhledem k tomu, že jsou prvotní centroidy generovány náhodně, je tento algoritmus nedeterministický a je tak často vhodné ho spustit několikrát a výsledky jednotlivých běhů porovnat. Tento nedostatek částečně řeší metoda *k-means++* [9], která centroidy generuje více deterministicky, avšak stále náhodně. Zde jsou popsány jednotlivé kroky algoritmu dle [9, s. 3]:

1. Náhodně vyber jeden centroid ze záznamů v datech.
2. Pro každý záznam $x$ spočítej jeho vzdálenost $D(x)$ k nejbližšímu již vybranému centroidu.
3. Náhodně vyber další centroid ze záznamů v datech pomocí váženého rozdělení, kde každý záznam je vybrán s pravděpodobností $D(x)^2$.
4. Kroky 2 a 3 opakuj, dokud není vybráno $K$ centroidů.
5. Nyní pokračuj se standardním algoritmem *K-means*.

Z experimentů popsaných v [9] vyplývá, že tento způsob určení prvotních centroidů zlepšuje kvalitu shluků.

Na principu centroidů vznikly i další algoritmy, které z *K-means* přímo či nepřímo vychází. Jedním je např. *Fuzzy C-means* [10], který využívá *fuzzy* hodnoty, které určují příslušnost záznamů do jednotlivých shluků. Dalším algoritmem je *KMOR (K-means with ourlier removal)* [11], který oproti běžnému algoritmu *K-means* umí za běhu detekovat a odstraňovat outliery.

### 2.1.3 Shlukování podle hustoty

Další kategorií je tzv. *Density-based clustering* neboli shlukování podle hustoty. To má oproti předchozím kategoriím tu výhodu, že není potřeba předem zadávat počet shluků (či úroveň řezu) a je navíc možné detekovat shluky různých tvarů.

Jedním z algoritmů z této kategorie je *DBSCAN (Density-based spatial clustering of applications with noise)* [12]. Ten má dva parametry — $\epsilon$, který určuje radius, ve kterém se kolem každého záznamu hledají jeho sousedi, a $minPoints$, který určuje minimální počet sousedů, které záznam musí mít, aby byl považován za tzv. *core point*. Algoritmus při svém běhu rozlišuje tři typy záznamů:

***core point*** Záznam, jehož počet sousedů v okruhu určeném parametrem $\epsilon$ je větší nebo roven hodnotě parametru $minPoints$.

***border point*** Záznam, který má v okruhu určeném parametrem $\epsilon$ aspoň jednoho souseda, který je *core point*.

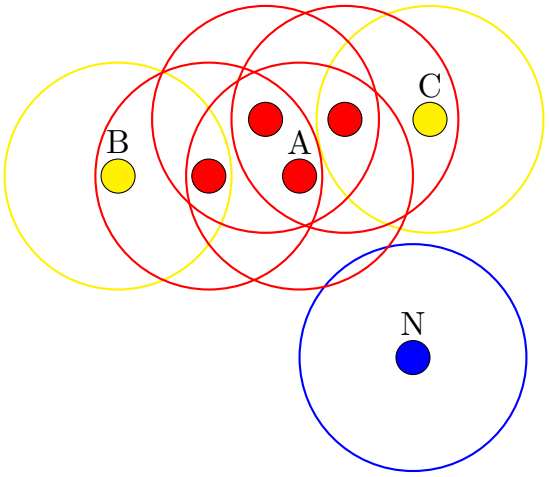
***noise point*** Záznam, jehož počet sousedů v okruhu určeném parametrem $\epsilon$ je nižší než hodnota parametru $minPoints$ a zároveň v tomto okruhu není žádný *core point*. Jinak také označován jako šum či outlier.

Ukázku popsané kategorizace je možno vidět na obrázku 2.2. Červeně jsou vyznačeny záznamy typu *core point* (např. $A$), žlutě jsou vyznačeny záznamy typu *border point* ($B$ a $C$) a modře je vyznačen záznam typu *noise point* ($N$).

Po kategorizace přichází na řadu tvorba jednotlivých shluků. Ta probíhá následujícím způsobem:

1. Vyber *core point*, který dosud nebyl umístěn do žádné shluku.
2. Pomocí algoritmu *DFS (Depth-first search)* [13] prohledej okolní prostor. Všechny nalezené záznamy typu *core point* a k nim příslušející záznamy typu *border point* umísti do stejného shluku.
3. Opakuj kroky 1 a 2, dokud existuje nějaký záznam typu *core point*, který nebyl umístěn do shluku.

Po konci běhu algoritmu tedy vznikne předem nedefinovaný počet shluků a seznam záznamů typu *noise point*. Je nutno podotknout, že volba parametrů




Obrázek 2.2: Ukázka kategorizace záznamů algoritmem *DBSCAN*

$\epsilon$ a *minPoints* musí být velice pečlivá. Např. při příliš vysoké hodnotě $\epsilon$ mohou být malé shluky sloučeny do jednoho velkého, při velmi nízké hodnotě může být naopak mnoho záznamů označeno jako *noise point*.

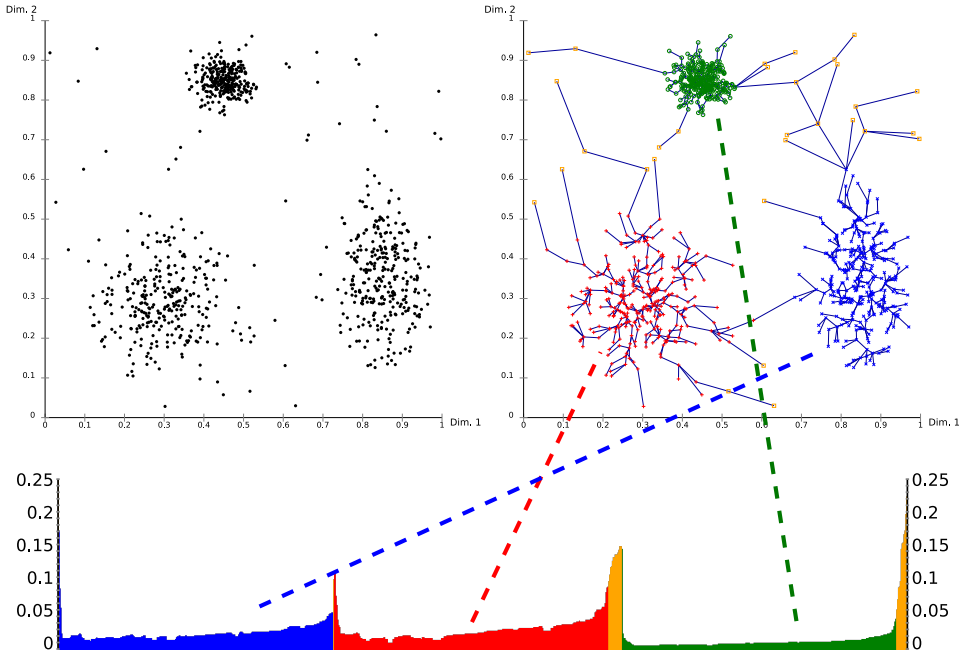
Podobně funguje i algoritmus *OPTICS (Ordering points to identify the clustering structure)* [14]. Vstupní parametry jsou stejné jako u algoritmu *DBSCAN*, postup vytváření jednotlivých shluků se však liší. Jsou zavedeny dvě nové hodnoty — *coreDistance* a *reachabilityDistance*. Ty jsou definovány následovně:

$$coreDistance(a) = \begin{cases} \text{nedefinována}, & \text{když } |N_\epsilon(a)| < minPoints \\ \delta(a, minPoints\text{-tý soused z } N_\epsilon(a)), & \text{jinak} \end{cases} \tag{2.6}$$

$$reachabilityDistance(a,b) = \begin{cases} \text{nedefinována}, & \text{když } |N_\epsilon(b)| < minPoints \\ \max\{coreDistance(b), \delta(a,b)\}, & \text{jinak} \end{cases} \tag{2.7}$$

$$N_\epsilon(a) = \{b : \delta(a,b) \leqslant \epsilon\} \tag{2.8}$$

Algoritmus podle postupu, který je popsán v [14, s. 5], vytváří seřazený seznam všech záznamů a spolu s nimi jejich hodnotu *coreDistance* a hodnotu *reachabilityDistance* od vhodně vybraného souseda (taktéž popsáno v [14, s. 5]). Vynesením získaných hodnot *reachabilityDistance* do grafu získáme tzv. *reachability plot*. Ukázku je možno vidět na obrázku 2.3.



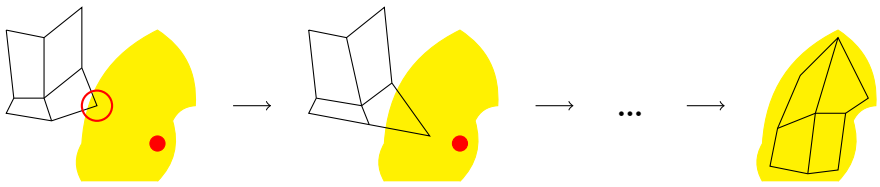

Obrázek 2.3: *Reachability plot* vytvořený algoritmem *OPTICS* a jeho použití pro tvorbu shluků [15]

Pro závěrečné vytváření shluků je ještě nutné vhodně stanovit hodnotu $\epsilon'$, která určuje hladinu řezu v grafu (celý proces je popsán v [14, s. 6]). Výsledné shluky by potom při pohledu na graf měly připomínat jakási „údolí“, přičemž hladina řezu by měla být pod pomyslnými „kopci“. Tento způsob má navíc oproti algoritmu *DBSCAN* tu výhodu, že zvládne vytvářet shluky o různých hustotách.

### 2.1.4 Umělé neuronové sítě

Další kategorií jsou umělé neuronové sítě, přičemž existuje hned několik způsobů, jak s jejich pomocí shlukovat data.

Prvním způsobem je tzv. *SOM (Self-organizing map)* [16] neboli samoorganizující se mapa, která je jinak také známá jako Kohonenova síť. Jedná se o způsob učení bez učitele. Tato síť je často používána pro dvourozměrnou vizualizaci vícerozměrných dat, nicméně díky své povaze může být použita i pro vytváření shluků. Neurony v síti tvoří mřížku, která je nejčastěji čtyřúhelníková nebo šestiúhelníková, přičemž neurony, které spolu v mřížce sousedí, by měly být blízko sebe i v původním datovém prostoru. Jednotlivé neurony (resp. váhy těchto neuronů) tedy reprezentují určitou část dat. Iterativní výpočet jednotlivých vah probíhá následujícím způsobem:



Obrázek 2.4: Průběh algoritmu *SOM* s mřížkou 3×3

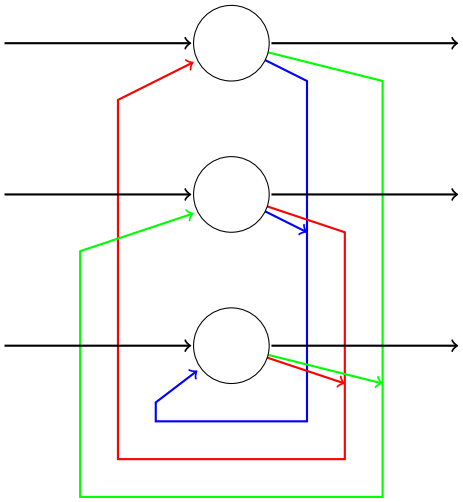
1. Náhodně vygeneruj váhy všech neuronů.
2. Náhodně vyber jeden záznam z dat.
3. Projdi všechny neurony a najdi takový, který je vybranému záznamu nejbližší.
4. Váhy tohoto neuronu a jeho sousedů přibliž hodnotě vybraného záznamu.
5. Kroky 2–4 opakuj, dokud nebyl vyčerpán maximální počet iterací.

Způsob přiblížení vah neuronů z kroku 4 je popsán v [16, s. 66]. Ukázkový běh algoritmu je možné vidět na obrázku 2.4.

Pro vytvoření jednotlivých shluků lze nyní postupovat podobně jako v případě algoritmu *K-means* (viz podkapitolu 2.1.2). Jednotlivé záznamy jsou rozřazeny do shluků podle toho, k jakému neuronu v mřížce mají nejblíže. Počet výsledných shluků je tedy určen velikostí mřížky.

Podobný princip využívá i algoritmus zvaný *Neural gas* (česky neuronový plyn) [17]. Neurony nicméně nejsou strukturovány do předem dané mřížky a jejich průběh rozmísťování — jak již název napovídá — připomíná pohyb skutečného plynu. Algoritmus se typicky používá pro kvantizaci dat (např. při rozpoznávání řeči [18] nebo obrázků [19]), ale je možné ho (podobným způsobem jako *SOM*) použít pro shlukování dat. Jedním z rozšíření tohoto algoritmu je např. *Growing Neural Gas* (česky rostoucí neuronový plyn) [20], který si počet neuronů v průběhu sám mění a mezi některými neurony udržuje spojení.

Dalším využitím neuronových sítí jsou tzv. rekurentní neuronové sítě. Jednou z nich je Hopfieldova síť [21] (ukázku propojení neuronů je možno vidět na obrázku 2.5). Ta se používá primárně jako *content-addressable memory* (česky obsahem adresovatelná paměť) a umožňuje ukládání binárních vzorů. Navíc je ale možné z paměti zrekonstruovat vzor, který byl poškozen nebo se z něj zachovala pouze část. Jinými slovy, síť je pro určitý vzor na vstupu schopna vrátit buď stejný vzor (pokud byl v síti uložen), nebo vzor velice podobný. Díky popsané funkcionalitě je tedy teoreticky možné natrénovat síť na části dat a následně záznamy rozřazovat do skupin na základě výstupu ze sítě.



Obrázek 2.5: Ukázka Hopfieldovy sítě se 3 neurony

### 2.1.5 Jiné metody

Jinou možností je např. *Distribution-based clustering* neboli shlukování založené na statistickém rozdělení dat. Jednou z používaných metod je metoda *Gaussian mixture models* [6, s. 430–439], která používá algoritmus *EM (Expectation–maximization)* [22].

Dále je možné kombinovat již výše zmíněné metody. Ukázkou takového přístupu je např. použití *SOM* (viz podkapitolu 2.1.4) v kombinaci s aglomerativním hierarchickým shlukováním (viz podkapitolu 2.1.1) nebo s algoritmem *K-means* (viz podkapitolu 2.1.2). [23]

Další ukázkou takového přístupu je použití algoritmu *K-means* (viz podkapitolu 2.1.2) pro účely hierarchického shlukování (viz podkapitolu 2.1.1) a to s použitím aglomerativního i divizního přístupu. [24]

## 2.2 Detekce outlierů

Outlier je záznam, který je od ostatních záznamů značně vzdálený. V principu se může jednat o chybu v měření, šum či o ojedinělé jevy. Běžně je tedy cílem tyto záznamy předem najít a zanalyzovat, neboť (jedná-li se o chyby) mohou značně vychýlit výsledné shluky (např. v případě počítání průměru souřadnic). Není nicméně vhodné tyto záznamy automaticky vyřazovat, neboť právě ony mohou nést nejcennější informace. [25], [26]

Některé shlukovací algoritmy jsou již v průběhu schopny outliery odhalovat a vyřazovat (např. *DBSCAN* [12] či *KMOR* [11]), pro jiné je však potřeba použít dodatečnou metodu, která je aplikována ještě před spuštěním daného algoritmu.

V této práci je konkrétně popsána metoda zvaná *Local outlier factor*. Dalšími metodami jsou např. *Iterative r Algorithm*, *Iterative z Algorithm* [27], použití algoritmu *kNN* [28] či tzv. *replicator neural networks* [29].

### 2.2.1 Local outlier factor

Metoda *LOF (Local outlier factor)* [30] byla představena v roce 2000. Jedná se o algoritmus, který je svým principem velmi podobný již popsanému shlukovacímu algoritmu *DBSCAN* (viz podkapitolu 2.1.3). Odhalování outlierů probíhá na základě hledání lokální odchylky záznamů. To je docíleno díky porovnání hustoty okolí daného záznamu oproti okolí jeho nejbližších sousedů. Jediným parametrem algoritmu je $k$, které určuje počet nejbližších sousedů, na základě kterých se počítá hustota.

Podobně jako v případě algoritmu *DBSCAN* definujeme pro každou dvojici záznamů $a$ a $b$ hodnotu $reachabilityDistance$:

$$reachabilityDistance(a,b) = \max\{\delta_k(b), \delta(a,b)\} \tag{2.9}$$

$$N_k(a) = \{k \text{ nejbližších sousedů } a\} \tag{2.10}$$

$$\delta_k(a) = \text{vzdálenost } k\text{-tého souseda } a \text{ z } N_k(a) \tag{2.11}$$

Ze vzorce je patrné, že hodnota $reachabilityDistance$ záznamů $a$ a $b$ bude $\delta_k(b)$, pokud $a$ patří mezi $k$ nejbližších sousedů $b$. Jinak bude hodnotou skutečná vzdálenost mezi těmito záznamy.

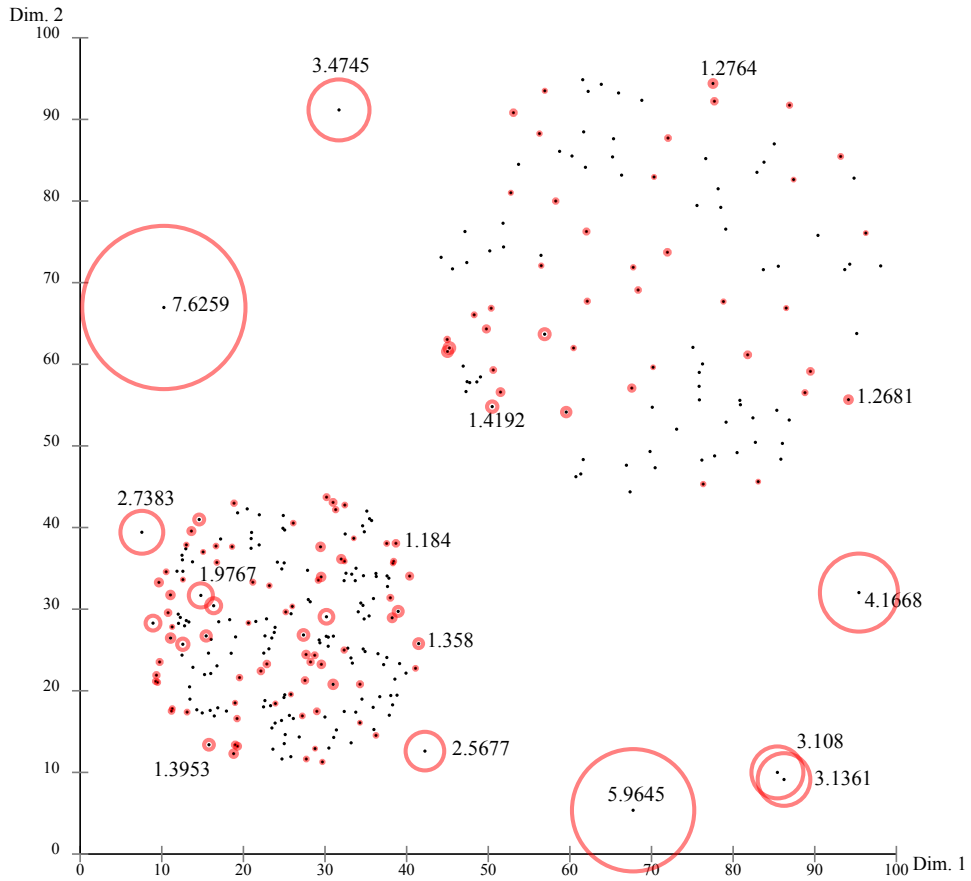
Dále pro každý záznam definujeme hustotu jeho okolí (jinak také *local reachability density*), kterou značíme $lrd$:

$$lrd(a) = \frac{|N_k(a)|}{\sum_{b \in N_k(a)} reachabilityDistance(a,b)} \tag{2.12}$$

Konkrétní hodnotu $LOF$ pro libovolný záznam lze nyní získat pomocí následujícího vzorce:

$$LOF(a) = \frac{\sum_{b \in N_k(a)} lrd(b)}{|N_k(a)| \cdot lrd(a)} \tag{2.13}$$

Jedná se v zásadě o poměr průměrné hustoty sousedů záznamu $a$ oproti hustotě záznamu $a$. Hodnoty nižší než 1 značí, že daný záznam není outlier. Hodnoty, které jsou v úzkém intervalu kolem 1, značí, že je záznam podobný záznamům v jeho okolí a není tedy outlier. Hodnoty značně větší než 1 signalizují, že by daný záznam mohl být outlier, nicméně neexistuje přesná hranice, která by určovala, zda jím záznam skutečně je. Z tohoto důvodu je potřeba takové záznamy dále analyzovat a zhodnotit jejich možný dopad. Ukázku použití popsané metody je možné vidět na obrázku 2.6.



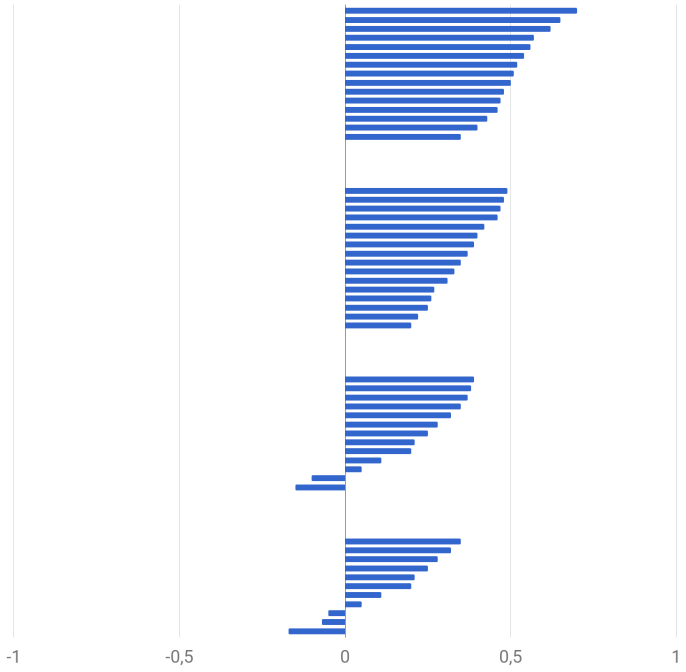

Obrázek 2.6: Ukázka detekce outlierů pomocí metody *LOF* [31]

## 2.3 Zjišťování kvality shluků

Po fázi, kdy jsou shluky vytvářeny, je potřeba určitým způsobem hodnotit jejich kvalitu. Pro to existují v zásadě dva způsoby — externí a interní. [32]

V případě externího způsobu jsou shluky hodnoceny pomocí určité informace, která ve fázi shlukování nebyla použita. Typicky se jedná o známou kategorii či třídu některých záznamů. Tato informace však často nebývá k dispozici.

Na druhé straně interní způsob hodnotí kvalitu shluků pouze na základě již známých informací a podle toho, do jakých shluků byly záznamy zařazeny. V této práci je konkrétně popsána metoda *Silhouettes* [33]. Dalšími přístupy jsou např. *DBI (Davies–-Bouldin index)* [34] či *DI (Dunn index)* [10], [35].




Obrázek 2.7: Ukázka grafu vytvořeného metodou *Silhouettes*

### 2.3.1 Silhouettes

Metoda *Silhouettes* [33] vytváří jednoduchou grafickou reprezentaci toho, jak dobře či špatně jsou jednotlivé záznamy umístěny do shluků. Výsledný graf poté připomíná jakési „siluety“ (odtud onen název).

Nechť $\alpha(a)$ je průměrem vzdáleností mezi záznamem $a$ a všemi ostatními záznamy ve shluku, do kterého byl záznam $a$ zařazen, a $\beta(a)$ je minimem z průměrů vzdáleností mezi záznamem $a$ a záznamy z jiných shluků. Potom je pro libovolný záznam $a$ definováno jeho skóre následujícím způsobem:

$$s(a) = \frac{\beta(a) - \alpha(a)}{\max\{\alpha(a), \beta(a)\}} \tag{2.14}$$

Skóre nabývá hodnot z intervalu $[-1, 1]$, přičemž kladné hodnoty značí dobrou příslušnost a záporné hodnoty značí špatnou příslušnost. Toto skóre je třeba vypočíst pro všechny záznamy. Následně jsou záznamy v jednotlivých shlucích seřazeny sestupně podle hodnoty jejich skóre a spolu s touto hodnotou jsou vyneseny do grafu. Ukázkový graf je možné vidět na obrázku 2.7.

Kvalitu shluků lze nyní určit jak na základě vizuální reprezentace, tak i pomocí průměru skóre v rámci jednotlivých shluků či napříč všemi.

## 2.4 Současná řešení pro tvorbu skupin studentů

V současné době existuje mnoho způsobů, jakými skupiny vytvářet a následně hodnotit.

V [36] je popsán systém, který shlukuje studenty podle informací o předchozích znalostech a zájmech, které byly získány pomocí dotazníků, a informací o jejich aktivitě v online výukovém systému získaných skrze vytěžování dat z *SQL (Structured Query Language)* databáze. Zkoumán je např. počet založených vláken ve fóru daného kurzu, počet vláken, na které studentovi někdo odpověděl, počet uzavřených vláken, aktivita v chatu, to, zda student poslal e-mail celému kurzu či zda publikoval prezentaci, a mnoho dalších atributů. (Kompletní seznam je možno nalézt v [36, s. 3].) V tomto případě je pro shlukování použita metoda maximální věrohodnosti v kombinaci s algoritmem *EM* [22] (viz podkapitolu 2.1.5). Ze získaných dat se podařilo vyvodit několik atributů, které přímo popisují vzorce chování a v určitých případech i přímo ovlivňují výsledky studentů, a na základě těchto atributů vzniklo celkem 6 shluků různě zaměřených studentů, čemuž odpovídalo i výsledné hodnocení.

D. Zakrzewska v [37] představuje způsob, jak vytvářet skupiny uživatelů na základě stylu učení a preferovaných barev uživatelského rozhraní. Styl učení byl získán pomocí dotazníku R. Feldera a B. Soloman [38], jehož výsledky reprezentují model R. Feldera a L. Silverman [39]. Studenti byli následně rozděleni do skupin pomocí dvou různých verzí algoritmu *Two-Phase Hierarchical Clustering*, který je popsán v [37] a [40]. Dále bylo navrženo, aby byl každé skupině předkládán jiný obsah, který vyhovuje potřebám daných studentů.

V [41] je popsán systému, který vytváří skupiny studentů na základě jejich stylu učení. Ten je zjištěn pomocí online dotazníku, ze kterého jsou automaticky vyvozeny určité vlastnosti a jejich míra. Testovaní studenti byli rozřazeni do 13 skupin po 3–5 studentech. Následně skupiny studentů pracovaly na týmovém projektu dle svého výběru. Pro potřeby výzkumu autoři vytvořili vlastní systém s názvem *PEGASUS*, který navíc umožňuje i manuální vytváření skupin. Studenti byli následně ohodnoceni na základě prezentace jejich týmového projektu. Z výsledků výzkumu a z reakcí studentů při rozhovorech vyplývá, že navržení složení skupin ve většině případů velice zjednodušilo týmovou práci i komunikaci.

V [42] je představen systém pro propojování studentů na základě podobnosti jejich aktivity v online výukovém systému, který umožňuje i diskuzi ve fóru. V potaz se bere např. úroveň předchozích znalostí, rychlost čtení, zájem o danou látku, známky z předchozích testů apod. (Všechny atributy jsou detailně popsány v [42, s. 275–276].) Data byla v průběhu dvou měsíců získávána z logu proxy serveru. Po výpočtu popsaných atributů byl spuštěn shlukovací algoritmus *K-means* (viz podkapitolu 2.1.2) a následně bylo pro každý shluk studentů vytvořeno oddělené fórum, skrze které studenti komunikovali a sdíleli své znalosti. Z výsledků závěrečných individuálních testů na konci 3. měsíce kurzu vyplývá, že ono rozdělení studentů mělo pozitivní vliv na jejich výkon,

neboť počet dobrých známek oproti 1. i 2. měsíci skokově vzrostl.

V [43] je popsán způsob, jak vytvářet skupiny studentů pomocí genetického algoritmu na základě několika předem definovaných charakteristik: odhad znalostí studenta vztahujících se k danému předmětu, odhad úrovně komunikačních schopností a odhad schopnosti studenta vést tým. Popsané odhady byly zjištěny pomocí tří dotazníků. Testováno bylo celkem 135 studentů, přičemž část z nich byla rozdělena do skupin za použití popsané metody, část náhodně a část dle vlastního uvážení. Algoritmus vykazoval dobré výsledky již zhruba po 1 000 generacích. Ze závěrů navíc vyplývá, že metoda úspěšně vytvořila homogenní skupiny a že takto vytvořené skupiny dosahovaly lepších výsledků než skupiny vytvořené náhodně či dle vlastního uvážení.

Kapitola 3

# Návrh

Pro potřeby návrhu, implementace a testování mého řešení mi byla poskytnuta anonymizovaná data z e-learningového systému Techambition [44], který se specializuje na interaktivní výuku středoškolské matematiky.

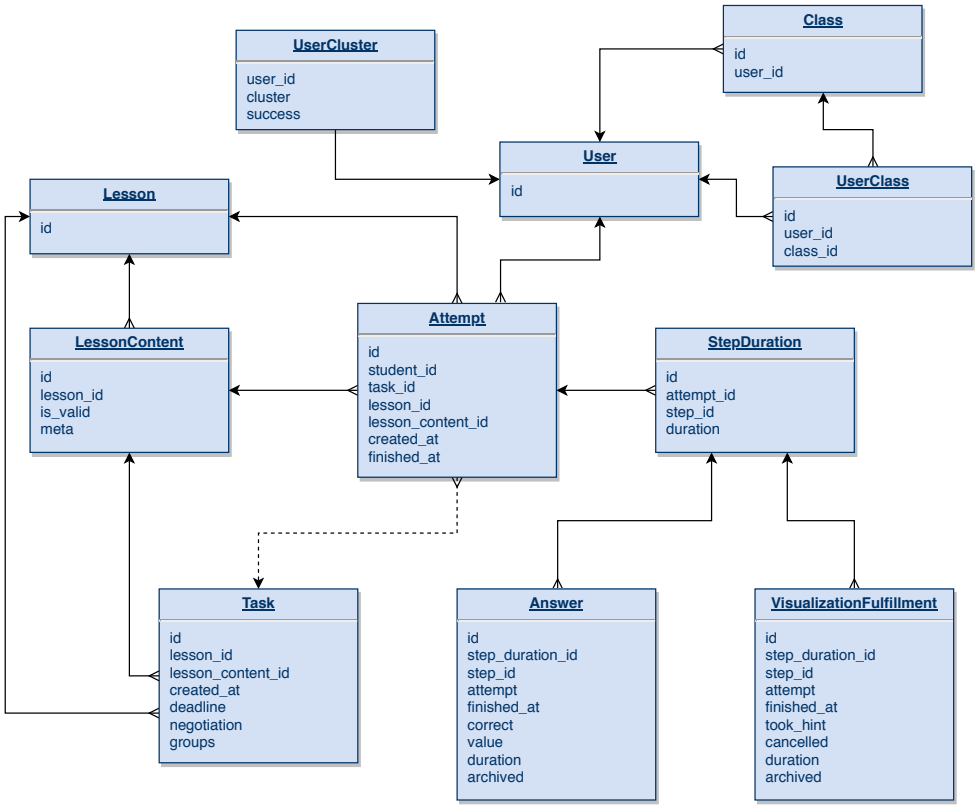
## 3.1 Analýza dat

Datový model použitých entit a atributů je možno vidět na obrázku 3.1

K dispozici jsou údaje o cca 6 300 studentech (entita `User`), kteří byli aktivní v tomto školním roce (2017/2018), a spolu s nimi údaje o cca 76 000 průchodech lekcí (entita `Attempt` a k ní přidružené entity `StepDuration`, `Answer` a `VisualizationFulfillment`). Celkově studenti prošli zhruba 400 různých lekcí (entita `Lesson`) z rozličných kapitol matematiky. Dále jsou dostupné informace o daných lekcích a jejich obsahu (entita `LessonContent` a její atribut `meta`), který se u většiny skládá z kroků obsahujících:

- teorii,
- interaktivní vizualizace pro pochopení určitých principů,
- uzavřené testy, ve kterých student vybírá jednu z daných možností,
- otevřené testy, ve kterých student počítá příklad a následně odešle výsledek,
- interaktivní vizualizace, ve kterých student řeší určitý úkol.

Na testy a úkoly ve vizualizacích v rámci jedné lekce má student neomezený počet pokusů (přičemž s počtem pokusů klesá hodnocení) a po každé špatné odpovědi je studentovi zobrazena nápověda. Otevřené testy a úkoly ve vizualizacích má navíc student možnost vzdát. Lekce studenti vypracovávají buď ve chvíli, kdy jim je učitel zadá jako úkol, nebo kdykoli jindy zcela dobrovolně.



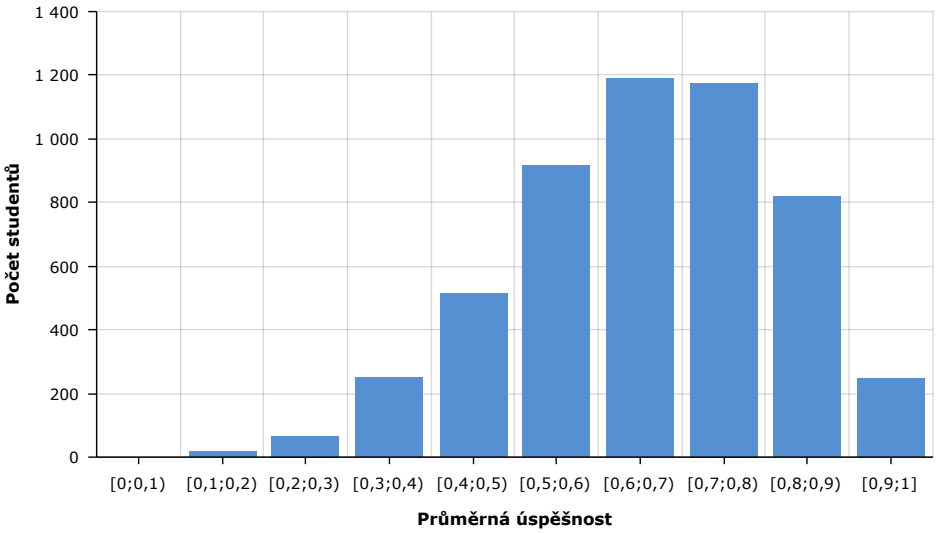

Obrázek 3.1: Datový model

## 3.2 Odvození nových atributů

Cílem je nalézt atribut, který bude vyjadřovat studentovu úroveň znalostí, a dále takové atributy, které budou určitým způsobem vyjadřovat jeho chování a na základě kterých bude možné vytvářet homogenní shluky. Úroveň znalostí lze poměrně dobře určit pomocí studentovy dosavadní úspěšnosti:

`success` průměrná úspěšnost studenta při prvních odpovědích v testech a při prvních odesláních úkolů ve vizualizaci

K tomuto účelu byly využity převážně entity `VisualizationFulfillment` a `Answer`, které obsahují informaci o správnosti řešení dané úlohy. Rozdělení atributu `success` je možno vidět na obrázku 3.2.




Obrázek 3.2: Rozdělení atributu `success`

Jak již bylo popsáno v podkapitole 3.1, obsah lekcí je uložen v atributu `meta` entity `LessonContent`. Oproti existujícím řešením, které jsou popsány v podkapitole 2.4, je tak možné analyzovat chování studentů při práci s různými komponentami v různých částech lekcí.

Z jednotlivých kroků lze vyčíst např. počet slov, či to, zda krok obsahuje vizualizaci. Po propojení entity `StepDuration` a jejího atributu `duration` s těmito informacemi bylo možné odvodit dva atributy:

**`wordTime`** průměrný čas (v sekundách) strávený u jednoho slova v teoretických krocích

**`visualizationTime`** průměrný čas (v sekundách) strávený v krocích, ve kterých je interaktivní vizualizace

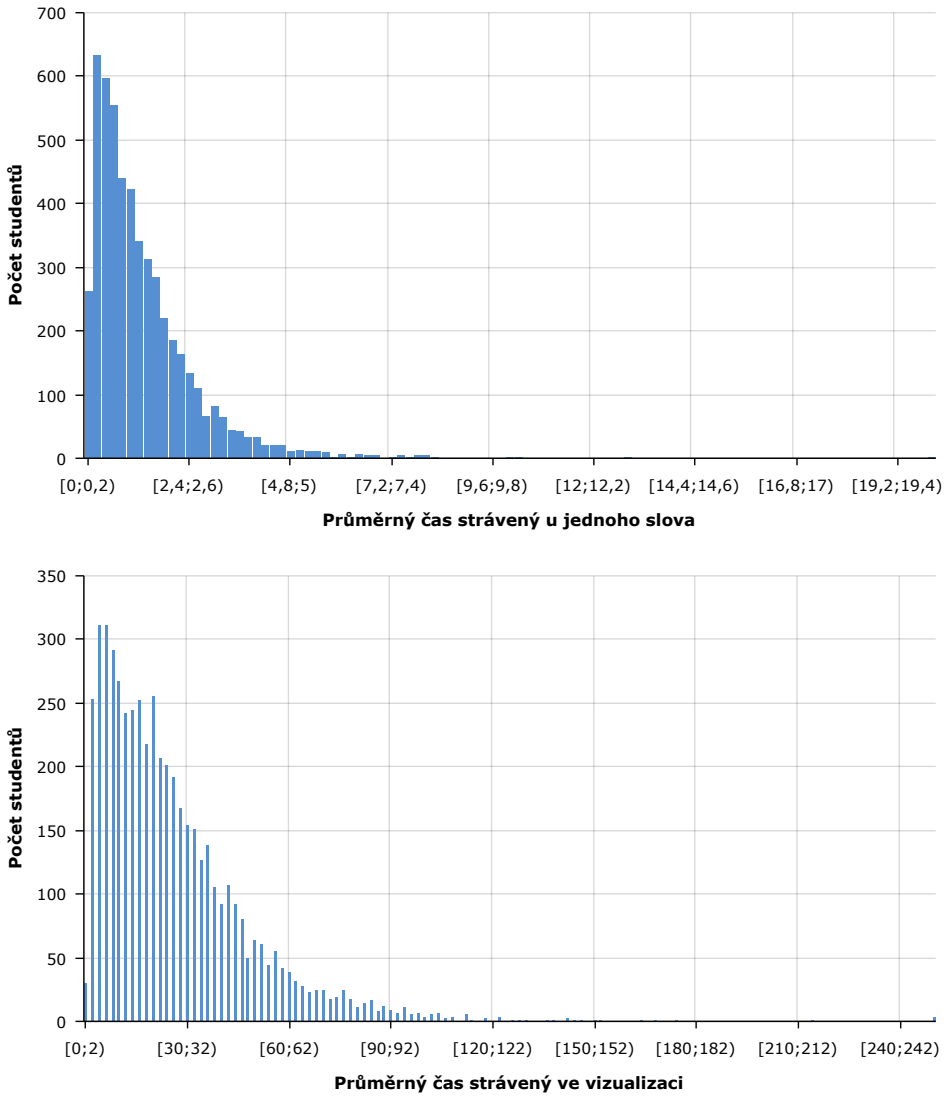
Rozdělení těchto atributů je možno vidět na obrázku 3.3.

Jak bylo popsáno v podkapitole 3.1, po každé špatné odpovědi v testu je studentovi zobrazena nápověda. Atribut `duration` entity `Answer` vyjadřuje dobu, za jakou byla odpověď zaznamenána. Rozdíl hodnot `duration` dvou po sobě jdoucích odpovědí tedy odpovídá době, jakou student strávil u nápovědy, než znovu odpověděl. Díky obsahu lekcí lze navíc rozlišovat typy jednotlivých testů. Po propojení těchto informací je tedy možné vyvodit další dva atributy:

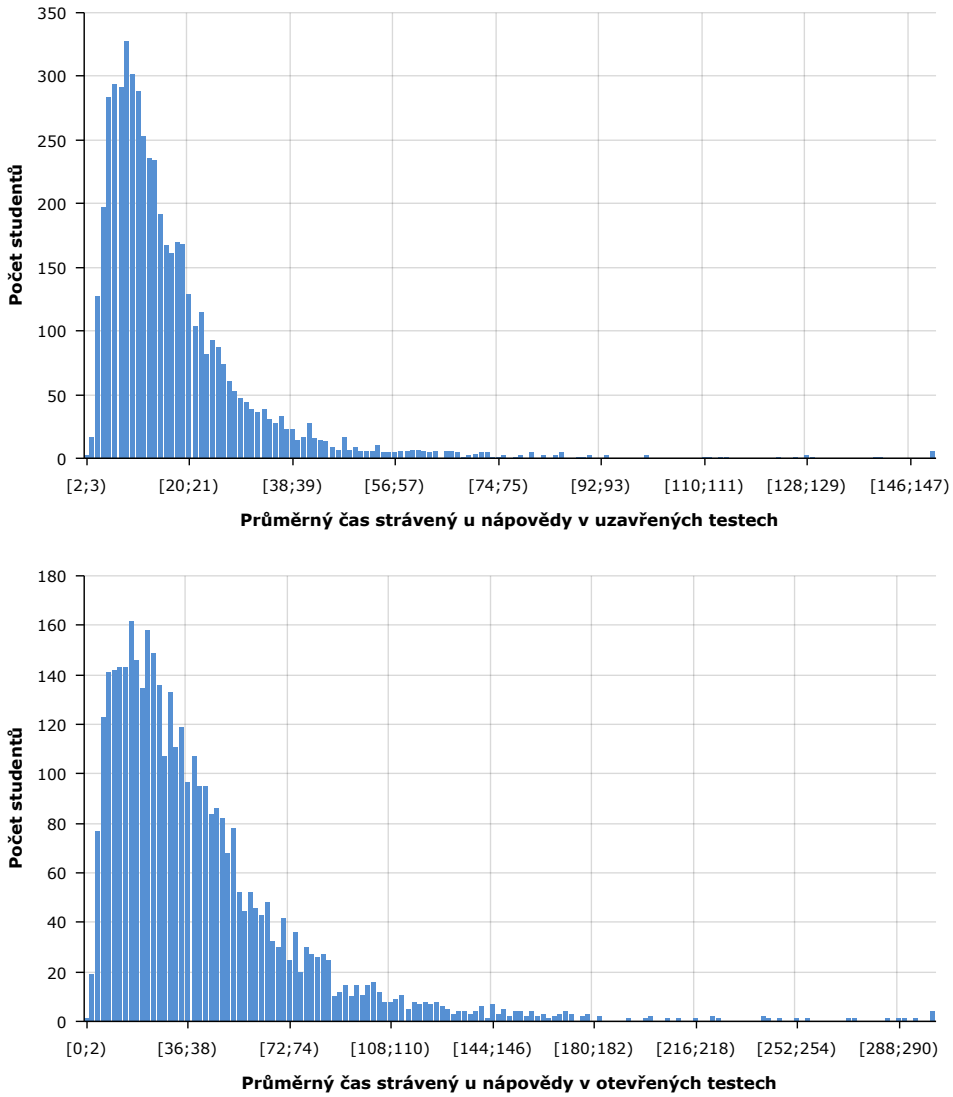
**`optionsHintTime`** průměrný čas strávený u nápovědy v uzavřených testech

**`rangesHintTime`** průměrný čas strávený u nápovědy v otevřených testech

Rozdělení těchto atributů je možno vidět na obrázku 3.4.




Obrázek 3.3: Rozdělení atributu `wordTime` (nahoře) a atributu `visualizationTime` (dole)




Obrázek 3.4: Rozdělení atributu `optionsHintTime` (nahoře) a atributu `rangesHintTime` (dole)

Atribut `took_hint` entity `VisualizationFulfillment` signalizuje, zda student při řešení úkolu ve vizualizaci použil nápovědu. Tato entita má dále atribut `cancelled`, který značí, že student řešení daného úkolu vzdal. Na základě těchto informací je možné vyvodit další dva atributy:

**`visualizationHintRatio`** poměr úkolů ve vizualizaci, u kterých student použil nápovědu, oproti všem těmto úkolům

**`visualizationCancelRatio`** poměr úkolů ve vizualizaci, které student vzdal, oproti všem těmto úkolům

Rozdělení těchto atributů je možno vidět na obrázku 3.5.

U otevřených testů existuje možnost vzdát se, pokud student odpověděl aspoň dvakrát. Tato skutečnost je signalizována tím, že k entitě `StepDuration` není přidružena žádná správná odpověď (entita `Answer` a její atribut `correct`). Podobně jako v případě úkolů ve vizualizaci je tedy možné odvodit další atribut:

**`answerCancelRatio`** poměr otevřených testů, které student vzdal, oproti všem těmto testům

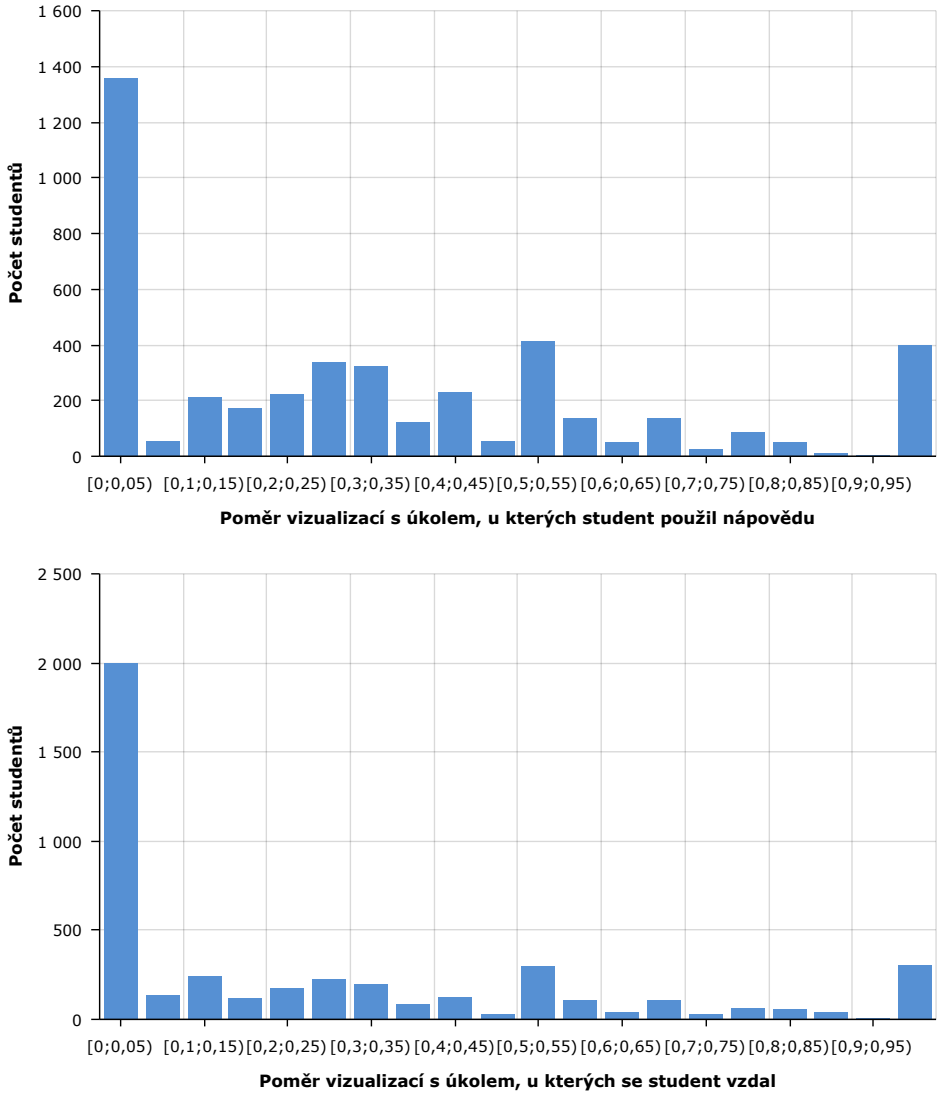
Student má možnost vypracovat lekci, aniž by učitel předem zadal úkol (viz podkapitolu 3.1). K dispozici jsou informace o všech průchodech lekcí daného studenta. Zároveň lze zjistit případná vazba mezi tímto průchodem (entita `Attempt`) a zadaným úkolem (entita `Task`). Na základě těchto informací je tedy možné odvodit následující atribut:

**`voluntaryRatio`** poměr lekcí, které student vypracoval dobrovolně, oproti všem jím vypracovaným lekcím

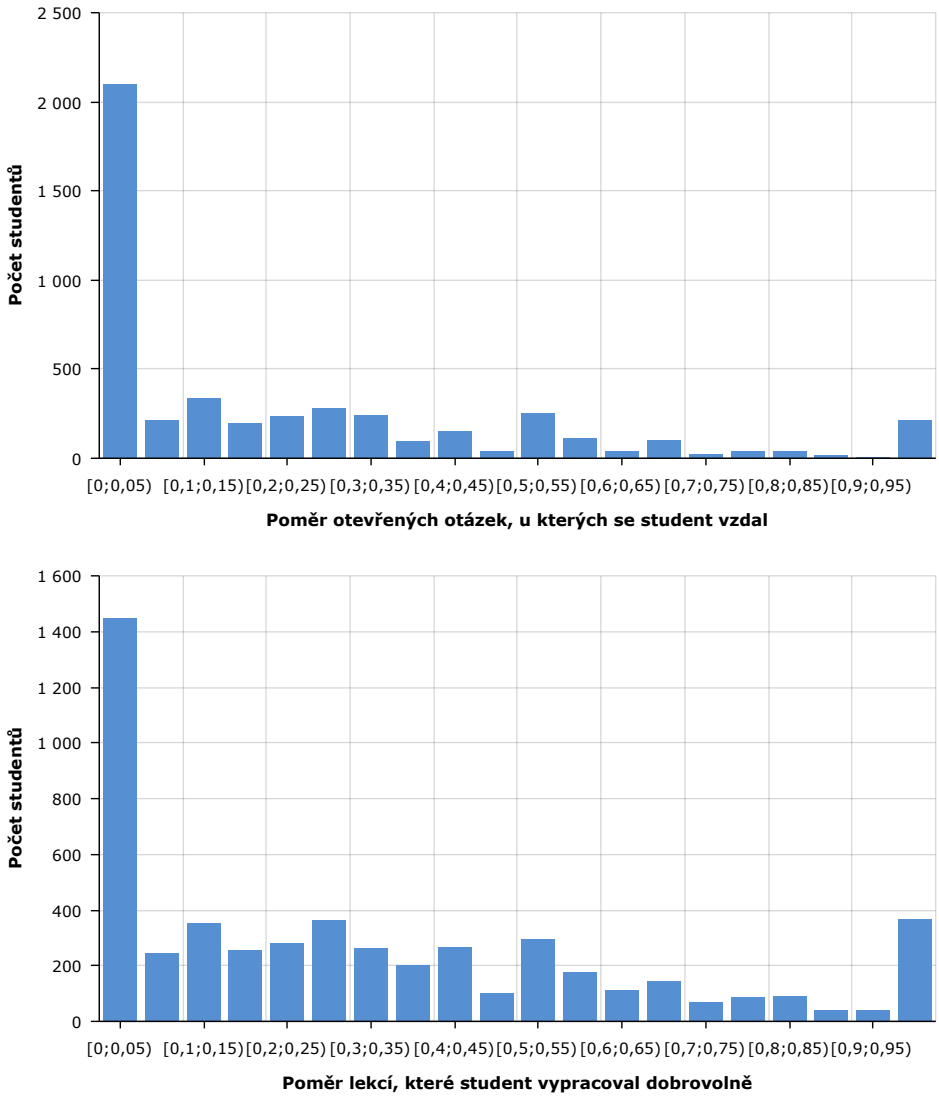
Rozdělení posledních dvou atributů je možno vidět na obrázku 3.6.

Celkově vzniklo cca 3 200 záznamů, které mají všechny zde popsané atributy. Dostatečný počet atributů (přičemž některé mohou být neznámé) má zhruba 5 200 záznamů.

Hodnoty korelačního koeficientu [45] mezi všemi dvojicemi atributů je možné vidět v tabulce 3.1. Největší korelace je mezi atributy `answerCancelRatio` a `success` $(-0,519)$, druhá největší je mezi atributy `visualizationCancelRatio` a `success` $(-0,483)$. U ostatních dvojic je korelace dostatečně malá.




Obrázek 3.5: Rozdělení atributu `visualizationHintRatio` (nahoře) a atributu `visualizationCancelRatio` (dole)




Obrázek 3.6: Rozdělení atributu `answerCancelRatio` (nahoře) a atributu `voluntaryRatio` (dole)

Tabulka 3.1: Korelace mezi odvozenými atributy

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| `visualizationTime`[1] | | 0,242 | 0,299 | 0,180 | -0,133 | -0,153 | -0,050 | -0,021 | 0,163 |
| `wordTime`[2] | 0,242 | | 0,161 | 0,154 | -0,131 | -0,092 | -0,035 | -0,014 | 0,113 |
| `optionsHintTime`[3] | 0,299 | 0,161 | | 0,289 | -0,241 | -0,240 | -0,117 | -0,059 | 0,375 |
| `rangesHintTime`[4] | 0,180 | 0,154 | 0,289 | | -0,240 | -0,194 | -0,081 | -0,040 | 0,269 |
| `answerCancelRatio`[5] | -0,133 | -0,131 | -0,241 | -0,240 | | 0,340 | 0,145 | 0,087 | -0,519 |
| `visualizationCancelRatio`[6] | -0,153 | -0,092 | -0,240 | -0,194 | 0,340 | | 0,322 | 0,046 | -0,483 |
| `visualizationHintRatio`[7] | -0,050 | -0,035 | -0,117 | -0,081 | 0,145 | 0,322 | | 0,050 | -0,288 |
| `voluntaryRatio`[8] | -0,021 | -0,014 | -0,059 | -0,040 | 0,087 | 0,046 | 0,050 | | -0,071 |
| `success`[9] | 0,163 | 0,113 | 0,375 | 0,269 | -0,519 | -0,483 | -0,288 | -0,071 | |

## 3.3 Zvolené shlukovací metody

Cílem je najít takovou metodu, která bude vytvářet kvalitní shluky. Jak bylo popsáno v podkapitole 2.1, existuje řada různých přístupů. Byly tedy zvoleny celkem 4 metody, které budou otestovány a porovnány:

1. *K-means* s inicializací centroidů pomocí algoritmu *k-means++* (viz podkapitolu 2.1.2),
2. *DBSCAN* (viz podkapitolu 2.1.3),
3. kombinace *SOM* (viz podkapitolu 2.1.4) a aglomerativního shlukování s použitím kritéria *complete-linkage* (viz podkapitolu 2.1.1),
4. *Hopfield network* (viz podkapitolu 2.1.4).

## 3.4 Postup vytváření skupin

Skupiny je potřeba vytvářet pro dva základní typy situací:

1. Studenti ve skupině vypracovávají zadanou lekci podobným způsobem jako při individuálním přístupu.
2. Studenti ve skupině vypracovávají určitý problematický krok lekce. Tu do té chvíle vypracovávali jednotlivě, aniž by měli k dispozici nápovědy a aniž by věděli, zda je jejich odpověď správná, či špatná.

Dále by měly být skupiny vytvářeny ideálně s ohledem na to, do jakých shluků byli studenti zařazeni, a zároveň s ohledem na jejich úspěšnost. Ne vždy se ovšem povede docílit toho, aby byly skupiny naprosto optimální, pro což existují dva hlavní důvody. Prvním je fakt, že počet skupin je stále v režii učitele. Druhým důvodem je, že v určitých třídách nemusí být zastoupení studentů v jednotlivých shlucích rovnoměrné.

Navržený postup vytváří skupiny následujícím způsobem:

1. Seřaď seznam studentů sestupně — primárně podle velikosti shluků a sekundárně podle úspěšnosti.
2. Vytvoř novou prázdnou skupinu.
3. Je-li seznam prázdný, přeskoč na krok 6.
4. Vyjmi první záznam a ten umísti do skupiny. Zároveň si zapamatuj shluk, do kterého patří.
5. Doplňuj skupinu, jsou-li v seznamu ještě nějaké záznamy a neobsahuje-li již skupina maximální počet záznamů:

   a) Vyber záznam ze stejného shluku. Neexistuje-li takový záznam a obsahuje-li skupina již minimální počet záznamů, cyklus ukonči.
   b) Je-li počet záznamů v předchozí skupině větší než minimální a jsou-li záznamy v ní ze stejného shluku, vyber záznam z ní.
   c) Existují-li záznamy, které nebyly zařazeny do shluků, vyber záznam z nich.
   d) Jinak vyber záznam z ostatních shluků.

6. Kroky 2–5 opakuj, dokud nebyl vytvořen požadovaný počet skupin.
7. Jsou-li v seznamu ještě nějaké záznamy, doplň jimi postupně nejmenší skupiny.
8. Existují-li skupiny, jejichž počet záznamů je menší než minimální, doplň je o záznamy z větších skupin.

Při doplňování skupiny, které je popsané v kroku 5, jsou střídavě vybírání studenti s různými úspěšnostmi. Nejprve je vybrán záznam se špatnou úspěšností. Dále (v závislosti na počtu záznamů ve skupině) je vybrán záznam s průměrnou úspěšností a poté je vybrán záznam s dobrou úspěšností. Tento proces se neustále opakuje.

V situaci, kdy skupiny vypracovávají problematický krok lekce (2. typ situace), se navíc zohledňují odpovědi studentů. Při výběrů studentů se špatnou úspěšností jsou prioritizováni ti, kteří měli správnou odpověď. Při výběru studentů s dobrou úspěšností jsou naopak prioritizováni ti, kteří měli špatnou odpověď.

Kapitola 4

# Realizace

V této kapitole je popsána implementace navrženého řešení a taktéž výsledky porovnávání a testování implementovaných metod.

## 4.1 Implementace

Implementace byla realizována ve formě knihovny, která je postavena na platformě Node.js [46] a napsána v programovacím jazyce TypeScript [47].

První částí knihovny je agregátor dat, který pomocí několika *SQL* dotazů zpracovává data popsaná v podkapitole 3.1 a vytváří atributy popsané v podkapitole 3.2. Pro komunikaci s databází a zpracování dat je využita knihovna Sequelize [48]. Jednotlivé dotazy jsou spouštěny sekvenčně po dávkách o 1 000 záznamech.

Druhou částí je poté čištění a normalizace dat. Jsou vyřazeny záznamy s neznámými atributy (např. z důvodu, že studenti neabsolvovali lekce s určitým typem úloh). Následně je na časových atributech spuštěna min-max normalizace (viz podkapitolu 2.1), která všechny tyto atributy přeškáluje do intervalu $[0, 1]$. Hodnoty poměrových atributů v tomto rozsahu již jsou.

Třetí částí je shlukování studentů na základě normalizovaných atributů. Implementovány jsou vybrané metody z podkapitoly 3.3:

- metoda *K-means* za použití knihovny skmeans [49],
- metoda *DBSCAN* za použití knihovny density-clustering [50],
- kombinace *SOM* a *AGNES* za použití knihoven ml-som [51] a hierarchical-clustering [52],
- metoda *Hopfield network* za použití knihovny Synaptic [53].

Čtvrtou a poslední částí knihovny je tvorba skupin. Zde je implementován postup, který je detailně popsaný v podkapitole 3.4. Vstupními parametry je počet skupin a seznam identifikátorů jednotlivých studentů.

Tabulka 4.1: Výsledky detekce potenciálních outlierů pomocí metody *LOF*

| Hodnota *LOF* | $k = 5$ | $k = 10$ | $k = 20$ | $k = 30$ |
|---|---|---|---|---|
| $\geqslant 1,1$ | 29,77 % | 26,13 % | 26,37 % | 26,80 % |
| $\geqslant 1,5$ | 2,41 % | 1,82 % | 1,63 % | 1,39 % |
| $\geqslant 1,75$ | 1,08 % | 0,71 % | 0,59 % | 0,56 % |
| $\geqslant 2$ | 0,46 % | 0,34 % | 0,31 % | 0,28 % |
| $\geqslant 3$ | 0,00 % | 0,00 % | 0,03 % | 0,03 % |

## 4.2 Detekce outlierů

Pro detekci outlierů byla zvolena metoda *LOF* (viz podkapitolu 2.2.1), která byla spuštěna s různými hodnotami parametru $k$. Výsledky je možno vidět v tabulce 4.1. Zastoupení potenciálních outlierů pro hodnoty $\geqslant 1,1$ je poměrně vysoké. Je nicméně velice nepravděpodobné, že by se skutečně jednalo o outliery. Pro hodnoty $\geqslant 1,5$ a $\geqslant 1,75$ je zastoupení zanedbatelné. Z tohoto důvodu jsem se rozhodl tyto záznamy v datech ponechat.

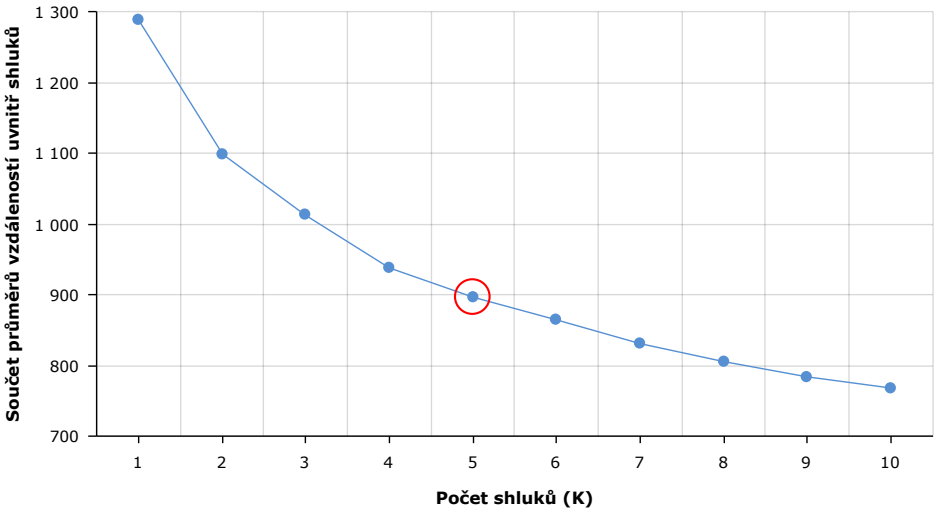
## 4.3 Porovnání metod

Pro porovnání kvality shluků vytvořených různými algoritmy byla zvolena metoda *Silhouettes* (viz podkapitolu 2.3.1).

První testovanou metodou je metoda *K-means* (viz podkapitolu 2.1.2). Ta má jediný parametr $K$, který určuje počet výsledných shluků. Problém obecně nastává ve chvíli, kdy je počet neznámý, což je i tento případ. Dá se očekávat, že se počet bude pohybovat v rozumných mezích. Pro stanovení optimální hodnoty $K$ tedy byla využita metoda *Elbow* [7], která byla spuštěna pro hodnoty v rozsahu 1–10. Jak lze vidět na obrázku 4.1, mezi hodnotami 4 a 5 je ještě znatelný pokles. Dál od hodnoty 5 je pokles téměř lineární. Jako parametr $K$ jsem tedy zvolil hodnotu 5. Graf vytvořený pomocí metody *Silhouettes* pro shluky vytvořené metodou *K-means* je možné vidět na obrázku 4.2.

Druhou testovanou metodou je *DBSCAN* (viz podkapitolu 2.1.3). Parametry této metody jsou $\epsilon$ a $minPoints$. Po vyzkoušení různých hodnot obou parametrů se nakonec jako použitelné jevily hodnoty $\epsilon = 0,1$ a $minPoints = 3$. Graf s těmito parametry je možné vidět na obrázku 4.2. Nicméně i tak byla do shluků rozřazena pouze čtvrtina záznamů. Byly-li navíc ignorovány příliš malé shluky, snížil se počet rozřazených záznamů na pouhou šestinu. Pro nižší hodnoty parametru $\epsilon$ byla převážná většina záznamů označena jako *noise point*. Podobný efekt měly i vyšší hodnoty parametru $minPoints$. Pro vyšší hodnoty parametru $\epsilon$ byla naopak většina záznamů zařazena do prvního shluku.

Třetí testovanou metodou je kombinace *SOM* (viz podkapitolu 2.1.4) a aglomerativního hierarchického shlukování (viz podkapitolu 2.1.1). Pro potřeby metody *SOM* byla vytvořena mřížka neuronů o rozměru 20×20. Ná-




Obrázek 4.1: Nalezení optimální hodnoty $K$ pomocí metody *Elbow*

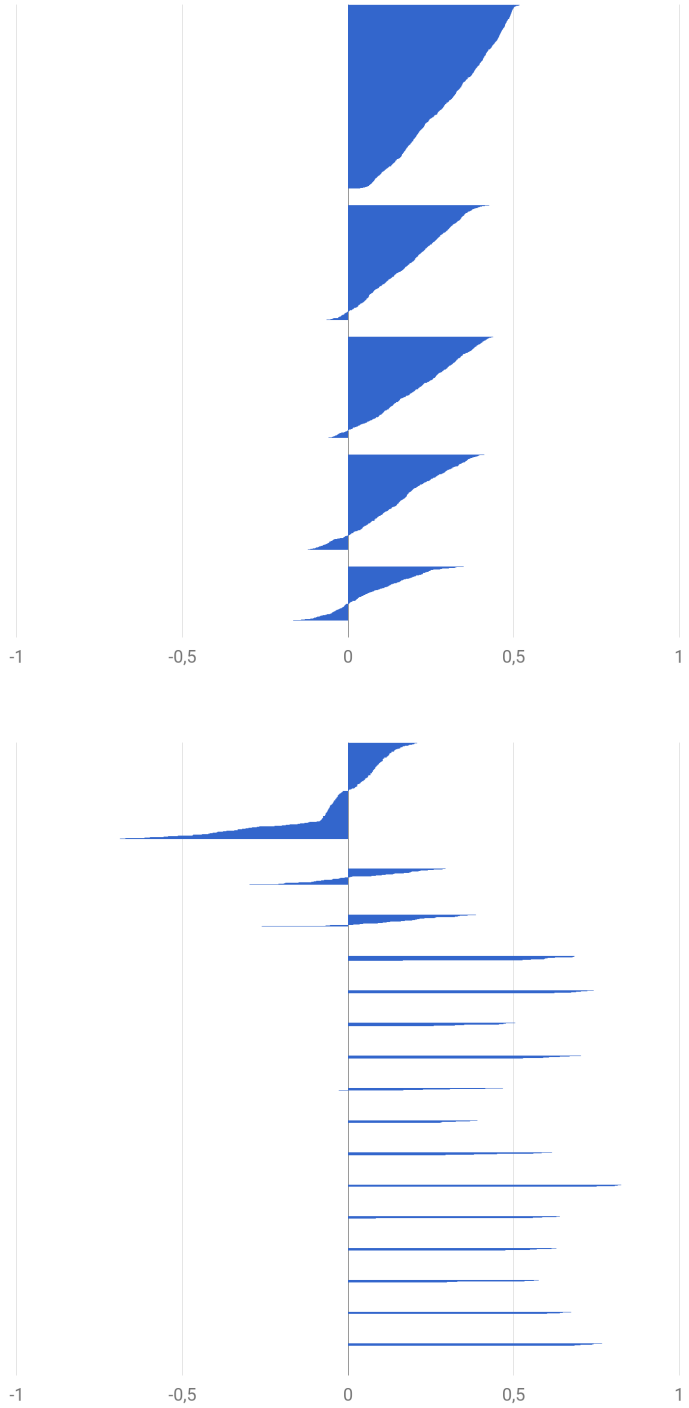
Tabulka 4.2: Průměry skóre jednotlivých metod

| Metoda | Průměr skóre |
|---|---|
| *K-means* | 0,184 |
| *DBSCAN* | 0,447 |
| *SOM* v kombinace s *AGNES* | 0,165 |
| *Hopfield network* | 0,186 |

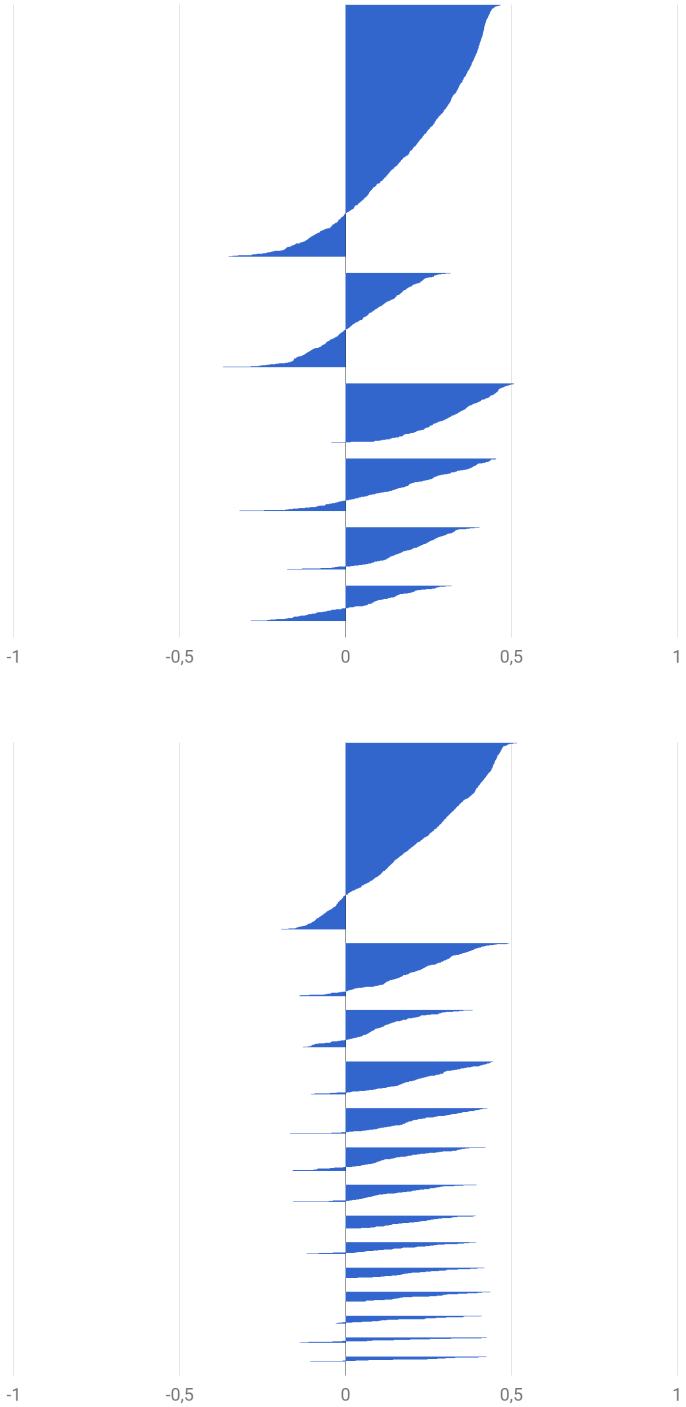
sledně bylo na neurony aplikováno aglomerativní hierarchické shlukování s kritériem *complete-linkage*. Byly vyzkoušeny celkem tři řezy, které vyprodukovaly 4, 5 a 6 shluků. V případě 4 a 5 shluků pojal první (největší) shluk více jak polovinu záznamů. V případě 6 shluků bylo již rozdělení záznamů mezi shluky akceptovatelné. Graf pro 6 shluků vytvořených touto metodou je možné vidět na obrázku 4.3.

Čtvrtou a poslední metodou bylo využití *Hopfield network* (viz podkapitolu 2.1.4). Síť byla trénována na desetině všech záznamů. Záznamy byly následně rozřazeny do shluků na základě výstupů ze sítě (neboli na základě nejpodobnějšího binárního vzoru). Graf pro shluky vytvořené touto metodou je možné vidět na obrázku 4.3. Metoda úspěšně pokryla všechny záznamy, nicméně počet shluků je příliš vysoký a některé shluky jsou velice malé.

Z hlediska počtu shluků a jejich velikostí jsou úspěšnější metody *K-means* a *SOM* v kombinaci s *AGNES*. Po zohlednění průměrů skóre (které je možné vidět v tabulce 4.2) je potom nejúspěšnější metoda *K-means*.




Obrázek 4.2: Graf *Silhouettes* pro metody *K-means* (nahoře) a *DBSCAN* (dole)




Obrázek 4.3: Graf *Silhouettes* pro metody *SOM* v kombinaci s *AGNES* (nahoře) a *Hopfield network* (dole)

Tabulka 4.3: Celková správnost a chyba při doplnění atributů nulami

| Shluk | Celková správnost | Chyba |
|---|---|---|
| 0 | 95,36 % | 4,64 % |
| 1 | 85,55 % | 14,45 % |
| 2 | 82,03 % | 17,97 % |
| 3 | 95,69 % | 4,31 % |
| 4 | 95,15 % | 4,85 % |

## 4.4 Doplnění neznámých atributů

Jak bylo popsáno v podkapitole 3.2, je k dispozici 5 200 záznamů, které mají dostatečný počet atributů (s některými chybějícími). Z toho kompletních záznamů je zhruba 3 200. Dostatečným počtem atributů se rozumí, že má záznam atributy `wordTime`, `visualizationTime` a `voluntaryRatio`, poté alespoň jeden atribut z dvojice `optionsHintTime` a `rangesHintTime` a aspoň jeden atribut z dvojice `answerCancelRatio` a `visualizationCancelRatio`.

Nabízí se tedy otázka, zda by nebylo možné tyto chybějící atributy doplnit nulami, aniž by tato skutečnost nepřiměřeně ovlivnila existující shluky. Pro tyto účely byla metoda *K-means* aplikována nejprve na kompletní záznamy a poté na záznamy s doplněnými nulami. Pro každý shluk s kompletními záznamy byl nalezen nejpodobnější shluk doplněných záznamů. Následně byla za použití matice záměn [54] spočtena celková správnost a chyba pro všechny shluky. Výsledky je možné vidět v tabulce 4.3. Pro většinu shluků je chyba velice nízká. Je tedy možné konstatovat, že doplnění chybějících atributů nulami nemá zásadní vliv na rozdělení kompletních záznamů.

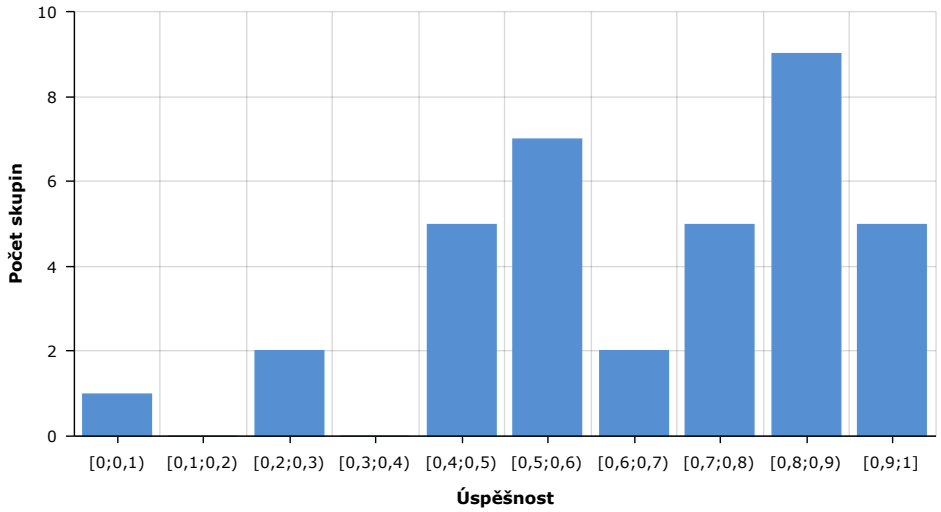
## 4.5 Testování na skutečných studentech

Implementovaná metoda *K-means* s doplňováním neznámých atributů (viz podkapitolu 4.4) byla v průběhu dubna a května 2018 testována v online výukovém systému Techambition [44].

Celkově bylo vytvořeno 36 skupin pro skupinové lekce a 23 skupin pro problematické kroky (viz podkapitolu 3.4). Rozdělení úspěšnosti ve skupinových lekcích je možné vidět na obrázku 4.4. Oproti rozdělení úspěšnosti v individuálních lekcích (viz obrázek 3.2) je zde vyšší zastoupení nejlepších úspěšností.

Dále je možné porovnat výsledky daných skupin s průměrem dosavadních výsledků jejich členů:

- 45 % skupin mělo výsledek lepší o 10 % či více.
- 22 % skupin mělo výsledek odpovídající svému průměru (±10 %).
- 33 % skupin mělo výsledek horší o 10 % či více.




Obrázek 4.4: Rozdělení úspěšnosti skupinových lekcí

47 % skupin mělo průměrnou úspěšnost jejich členů menší než 60 %. Po zohlednění této skutečnosti bylo možné získat další informace:

- Tento průměr měly celkem 2/3 skupin, které zaznamenaly zlepšení.
- Zlepšení o 20 % či více zaznamenalo 30 % skupin. Z těchto skupin mělo 5/6 onen horší průměr (pod 60 %).

Celkově se tedy ve skupinových lekcích zlepšily spíše skupiny s horším průměrem.

Skupiny vytvořené pro účely problematických kroků taktéž ve většině případů zaznamenaly úspěch. Celkem 74 % skupin vyřešilo danou úlohu správně.

# Závěr

Cílem rešeršní části práce bylo zmapovat současné metody a řešení pro tvorbu skupin studentů. To se dozajista podařilo, neboť v podkapitole 2.1 byly popsány různé způsoby shlukování dat a v kapitole 2.4 byla představena řada metod, pomocí kterých je možné analyzovat chování studentů a vytvářet skupiny.

Cílem praktické části práce bylo nejprve zanalyzovat dostupná data, rozhodnout, jak je zpracovat, a poté implementovat několik metod a porovnat jejich úspěšnost. Fáze analýzy a zpracování dat byly pečlivě popsány v podkapitolách 3.1 a 3.2. V podkapitole 4.3 byly porovnány celkem 4 různé metody. Nejlepší výsledky měla metoda *K-means*, která byla dále testována na skutečných studentech.

Z výsledků popsaných v podkapitole 4.5 vyplývá, že více jak polovina skupin dosáhla průměrných či nadprůměrných výsledků. Z toho lze vyvodit, že mnou navržené a implementované řešení je úspěšné a vytváří skupiny studentů dosahujících dobrých výsledků.

V budoucnosti by bylo možné práci rozšířit o podrobnější analýzu chování studentů (např. při práci s interaktivními vizualizacemi). Dále by bylo možné vytvářet skupiny studentů i s ohledem na to, zda dané složení již někdy zaznamenalo úspěch, či nikoli. Taktéž připadá v úvahu možnost doporučovat učiteli vhodný počet skupin s ohledem na zastoupení jednotlivých shluků ve třídě.

# Literatura


[1] Rokach, L.; Maimon, O.: Clustering methods. In *Data mining and knowledge discovery handbook*, Springer, 2005, s. 321–352.

[2] Raschka, S.: About Feature Scaling and Normalization [online]. 11. července 2014, [vid. 2018-05-11]. Dostupné z: http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

[3] Institut biostatistiky a analýz Masarykovy univerzity: Standardizace dat [online]. [vid. 2018-05-13]. Dostupné z: http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--vicerozmerna-rozdeleni-pravdepodobnosti--transformace-dat--standardizace-dat

[4] Hierarchical Clustering Essentials - Unsupervised Machine Learning [online]. [vid. 2018-05-08]. Dostupné z: http://www.sthda.com/english/wiki/print.php?id=237

[5] MacQueen, J.; aj.: Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, ročník 1, Oakland, CA, USA, 1967, s. 281–297.

[6] Bishop, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006, ISBN 0387310738.

[7] Bertagnolli, N.: Elbow Method and Finding the Right Number of Clusters [online]. [vid. 2018-04-18]. Dostupné z: http://www.nbertagnolli.com/jekyll/update/2015/12/10/Elbow.html

[8] Selim, S. Z.; Ismail, M. A.: K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on pattern analysis and machine intelligence*, , č. 1, 1984: s. 81–87.

[9] Arthur, D.; Vassilvitskii, S.: k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2007, s. 1027–1035.

[10] Dunn, J. C.: A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. 1973.

[11] Gan, G.; Ng, M. K.-P.: k-means clustering with outlier removal. *Pattern Recognition Letters*, ročník 90, 2017: s. 8–14.

[12] Ester, M.; Kriegel, H.-P.; Sander, J.; aj.: A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, ročník 96, 1996, s. 226–231.

[13] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; aj.: Depth-first search. In *Introduction to Algorithms*, kapitola 22.3, MIT Press and McGraw-Hill, druhé vydání, 2001, ISBN 0-262-03293-7, s. 540–549.

[14] Ankerst, M.; Breunig, M. M.; Kriegel, H.-P.; aj.: OPTICS: ordering points to identify the clustering structure. In *ACM Sigmod record*, ročník 28, ACM, 1999, s. 49–60.

[15] Chire: Result of running the OPTICS on the data set on the upper left [online] [obrázek]. [vid. 2018-05-04]. Dostupné z: `https://commons.wikimedia.org/wiki/File:OPTICS.svg`

[16] Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. *Biological cybernetics*, ročník 43, č. 1, 1982: s. 59–69.

[17] Martinetz, T.; Schulten, K.; aj.: A "Neural-Gas" Network Learns Topologies. 1991.

[18] Curatelli, F.; Mayora-Ibarra, O.: Competitive learning methods for efficient vector quantizations in a speech recognition environment. In *Mexican International Conference on Artificial Intelligence*, Springer, 2000, s. 108–114.

[19] Angelopoulou, A.; Psarrou, A.; Rodríguez, J. G.; aj.: Automatic landmarking of 2D medical shapes using the growing neural gas network. In *International Workshop on Computer Vision for Biomedical Image Applications*, Springer, 2005, s. 210–219.

[20] Fritzke, B.: A growing neural gas network learns topologies. In *Advances in neural information processing systems*, 1995, s. 625–632.

[21] Hopfield, J. J.: Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, ročník 79, č. 8, 1982: s. 2554–2558.

[22] Dempster, A. P.; Laird, N. M.; Rubin, D. B.: Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1977: s. 1–38.

[23] Vesanto, J.; Alhoniemi, E.: Clustering of the self-organizing map. *IEEE Transactions on neural networks*, ročník 11, č. 3, 2000: s. 586–600.

[24] Chen, T.-S.; Tsai, T.-H.; Chen, Y.-T.; aj.: A combined K-means and hierarchical clustering method for improving the clustering efficiency of microarray. In *Intelligent Signal Processing and Communication Systems, 2005. ISPACS 2005. Proceedings of 2005 International Symposium on*, IEEE, 2005, s. 405–408.

[25] Liu, H.; Shah, S.; Jiang, W.: On-line outlier detection and data cleaning. *Computers & chemical engineering*, ročník 28, č. 9, 2004: s. 1635–1647.

[26] Ben-Gal, I.: Outlier detection. In *Data mining and knowledge discovery handbook*, Springer, 2005, s. 131–146.

[27] Lu, C.-T.; Chen, D.; Kou, Y.: Algorithms for spatial outlier detection. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, IEEE, 2003, s. 597–600.

[28] Knorr, E. M.; Ng, R. T.; Tucakov, V.: Distance-based outliers: algorithms and applications. *The VLDB Journal—The International Journal on Very Large Data Bases*, ročník 8, č. 3-4, 2000: s. 237–253.

[29] Hawkins, S.; He, H.; Williams, G.; aj.: Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, Springer, 2002, s. 170–180.

[30] Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; aj.: LOF: identifying density-based local outliers. In *ACM sigmod record*, ročník 29, ACM, 2000, s. 93–104.

[31] Chire: Visualization of LOF outlier scores using ELKI [online] [obrázek]. [vid. 2018-05-09]. Dostupné z: `https://commons.wikimedia.org/wiki/File:LOF.svg`

[32] Rendón, E.; Abundez, I.; Arizmendi, A.; aj.: Internal versus external cluster validation indexes. *International Journal of computers and communications*, ročník 5, č. 1, 2011: s. 27–34.

[33] Rousseeuw, P. J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, ročník 20, 1987: s. 53–65.

[34] Davies, D. L.; Bouldin, D. W.: A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, , č. 2, 1979: s. 224–227.

[35] Dunn, J. C.: Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, ročník 4, č. 1, 1974: s. 95–104.

[36] Talavera, L.; Gaudioso, E.: Mining student data to characterize similar behavior groups in unstructured collaboration spaces. In *Workshop on artificial intelligence in CSCL. 16th European conference on artificial intelligence*, 2004, s. 17–23.

[37] Zakrzewska, D.: Using Clustering Technique for Students' Grouping in Intelligent E-Learning Systems. In *HCI and Usability for Education and Work*, editace A. Holzinger, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, ISBN 978-3-540-89350-9, s. 403–410.

[38] Felder, R. M.; Soloman, B. A.: Index of Learning Styles Questionnaire [online]. [vid. 2018-04-17]. Dostupné z: `https://www.webtools.ncsu.edu/learningstyles/`

[39] Felder, R. M.; Silverman, L. K.; aj.: Learning and teaching styles in engineering education. *Engineering education*, ročník 78, č. 7, 1988: s. 674–681.

[40] Zakrzewska, D.: Cluster analysis for users' modeling in intelligent e-learning systems. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, Springer, 2008, s. 209–214.

[41] Kyprianidou, M.; Demetriadis, S.; Tsiatsos, T.; aj.: Group formation based on learning styles: can it improve students' teamwork? *Educational Technology Research and Development*, ročník 60, č. 1, 2012: s. 83–110.

[42] Manikandan, C.; Sundaram, A. M.; Babu, M. M.: Collaborative E-Learning for Remote Education; An Approach For Realizing Pervasive Learning Environments. In *2006 International Conference on Information and Automation*, prosinec 2006, ISSN 2151-1802, s. 274–278, doi: 10.1109/ICINFA.2006.374128.

[43] Moreno, J.; Ovalle, D. A.; Vicari, R. M.: A genetic algorithm approach for group formation in collaborative learning considering multiple student characteristics. *Computers & Education*, ročník 58, č. 1, 2012: s. 560–569.

[44] TECHAMBITION LTD: Techambition [online]. [vid. 2018-04-18]. Dostupné z: https://techambition.com

[45] Institut biostatistiky a analýz Masarykovy univerzity: Korelační koeficient [online]. [vid. 2018-05-13]. Dostupné z: http://portal.matematickabiologie.cz/index.php?pg=analyza-a-modelovani-dynamickych-biologickych-dat--signaly-a-linearni-systemy--modely-velicin-spojitych-v-case-ii--3-korelace--3-1-korelacni-koeficient

[46] Node.js Foundation: Node.js [software]. [vid. 2018-04-18]. Dostupné z: https://nodejs.org

[47] Microsoft Corporation: TypeScript [software]. [vid. 2018-04-18]. Dostupné z: http://www.typescriptlang.org

[48] Depold, S.; Hansen, M.; Meier, J. A.; aj.: Sequelize [software]. [vid. 2018-04-18]. Dostupné z: https://github.com/sequelize/sequelize

[49] Matarrodona, D. G.: skmeans [software]. [vid. 2018-04-18]. Dostupné z: https://github.com/solzimer/skmeans

[50] Lukasz; Coote, A.; Feuillet, T.: density-clustering [software]. [vid. 2018-04-25]. Dostupné z: https://github.com/uhho/density-clustering

[51] Zasso, M.: ml-som [software]. [vid. 2018-04-25]. Dostupné z: https://github.com/mljs/som

[52] jongleberry; Pauls, F.: hierarchical-clustering [software]. [vid. 2018-04-25]. Dostupné z: https://github.com/math-utils/hierarchical-clustering

[53] Cazala, J.; mkondel; Rodionov, V.: Synaptic [software]. [vid. 2018-04-25]. Dostupné z: https://github.com/cazala/synaptic

[54] Institut biostatistiky a analýz Masarykovy univerzity: Úvod do hodnocení úspěšnosti klasifikace [online]. [vid. 2018-05-13]. Dostupné z: http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--klasifikace--hodnoceni-uspesnosti-klasifikace--uvod-do-hodnoceni-uspesnosti-klasifikace

Dodatek A

# Seznam použitých zkratek

**AGNES** Agglomerative Nesting

**DBI** Davies–-Bouldin index

**DBSCAN** Density-based spatial clustering of applications with noise

**DFS** Depth-first search

**DI** Dunn index

**DIANA** Divisive Analysis Clustering

**EM** Expectation–maximization

**KMOR** K-means with ourlier removal

**LOF** Local outlier factor

**OPTICS** Ordering points to identify the clustering structure

**SOM** Self-organizing map

**SQL** Structured Query Language

Dodatek B

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `edu-collab` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF