ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ

BAKALÁŘSKÁ PRÁCE

SAMOUČÍCÍ ALGORITMY STROJOVÉHO UČENÍ PRO ANALÝZU PRŮMYSLOVÝCH DAT

2019

JIŘÍ JURÍK

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jurík** Jméno: **Jiří** Osobní číslo: **409085**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Samoučící algoritmy strojového učení pro analýzu průmyslových dat

Název bakalářské práce anglicky:

Unsupervised machine learning algorithms for industrial data analysis

Pokyny pro vypracování:

Proveďte rešerši úloh analýzy průmyslových dat (z výroby, z linek, ne obraz, prediktivní údržba strojů, linek, detekce poruchových stavů, klasifikace stavů obecně), tj. co se dnes tím vlastně vše řeší, kam směřuje trend...)
Proveďte rešerši publikovaných metod strojového učení se samoučícími algoritmy používaných pro analýzu průmyslových dat (např. klasifikace a detekce stavů (běžných, poruchových)…)
Popište koncept Big Data a specifické vlastnosti samoučících algoritmů pro Big Data a na základě rešerše vyhodnoťte stav konceptu Big Data v průmyslových aplikacích.
Na simulovaných datech (a případně dodaných reálných) otestujte a porovnejte vybrané metody (k-means, SOM, autoenkodér,…, použití jednoduché konvoluční vrsty + samoučící třídění dat,…?)
Rozsah práce min. 35 stran + přílohy

Seznam doporučené literatury:

[1] Berry, Michael W, and Murray Browne. Lecture Notes in Data Mining, World Scientific Publishing Co Pte Ltd, 2006. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/cvut/detail.action?docID=1681620.
[2] CADY, Field. The data science handbook. Hoboken, NJ: Wiley, 2017. ISBN 978-1-119-09292-6.
[3] Kaufman, Marcia, et al. Big Data for Dummies, John Wiley & Sons, Incorporated, 2013. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/cvut/detail.action?docID=1160914.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Ivo Bukovský, Ph.D., U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **29.04.2019** Termín odevzdání bakalářské práce: **16.08.2019**

Platnost zadání bakalářské práce: _____

doc. Ing. Ivo Bukovský, Ph.D.	prof. Ing. Milan Růžička, CSc.	prof. Ing. Michael Valášek, DrSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího bakalářské práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků bakalářské práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

V Praze, dne _____ *Podpis* _____

Poděkování

Rád bych tímto poděkoval vedoucímu této práce doc. Ing. Ivu Bukovskému, Ph. D. za jeho čas, ochotu a trpělivost a pomoc, kterou mi poskytl při vypracování této bakalářské práce a za vše, co jsem se díky němu naučil. Dále děkuji společnostem CompoTech PLUS s.r.o. za poskytnutí dat použitých v této práci.

Rovněž děkuji své rodině za trpělivost a nekonečnou podporu v průběhu mých studií a vypracovávání této práce. Děkuji Janě za radu, že se vyplácí dát se tou těžší cestou a její podporu. A v neposlední řadě děkuji svým přátelům, bez kterých by toto nebylo možné.

Souhrn: Cílem této bakalářské práce je rešerše problematiky analýzy průmyslových dat za pomocí výpočetní inteligence, představení několika metodik strojového učení bez učitele pro předzpracování, zpracování a vyhodnocování dat a aplikace vybraných metod na reálná a umělá data. Práce je věnována metodám redukce dimenzí vysoko rozměrných datasetů a metodám shlukování dat na základě jejich podobnosti.

Klíčová slova: Neuronové sítě, učící algoritmy bez učitele, klasifikace, redukce dimenzí, zpracování dat, předzpracování dat, PCA, LDA, Autoenkodéry, t-SNE, K-Means, K-Medoids, SOM, DBSCAN

Summary: The aim of this thesis is a review of current problems of industrial data analysis with application of computational intelligence. Several methods are introduced for data preprocessing, processing, analyzing and applications of these methods on real and artificial data. Thesis is devoted to dimensionality reduction of high-dimensional data methods and to clustering methods.

Key words: Neural networks, unsupervised learning, clasification, dimensionality reduction, data processing, data preprocessing, PCA, LDA, Autoenkoders, t-SNE, K-Means, K-Medoids, SOM, DBSCAN

Obsah

Úvod

Cíle této bakalářské práce je rešerše problematiky zabývající se samoučícími algoritmy strojového učení pro analýzu průmyslových dat a aplikace několika známých metodik pro jejich předzpracování a zpracování v odvětví výpočetní inteligence, je tedy tak rozdělena na dvě části, kde v první části jsou teoreticky popsány metody PCA (Principal Component Analysis), LDA (Linear Discriminant Analysis), tSNE (t-Distributed Stochastic Neighbor Embedding), Autoenkodéry, K-Means, K-Medoid, DBSCAN (Density Based Spatial Clustering of Applications with Noise) a SOM (Self-organizing Maps). V druhé části jsou pak některé z těchto metod aplikovány na reálná data, která byla změřena pomocí laserového vibrometru během poškozování kompozitních struktur ve tvaru trubek. Programovacím jazykem byl zvolen jazyk Python 3.7, jako vývojové prostředí bylo zvoleno Spyder 3.3.3.

Strojové učení je aplikace umělé inteligence, která umožňuje systémům se automaticky učit a zlepšovat na základě zkušeností, aniž by byly ručně programovány. Je zde snaha o vývoj softwarových programů, které se dokáží učením na daných datech zdokonalovat. Učící proces začíná předložením algoritmů strojového učení výsledky z pozorování (data) a instrukce, jak se má daný algoritmus na datech učit. Cílem zde je dosažení stavu, kdy se program učí sám bez lidského zásahu [1].

Strojové učení se obecně dělí dle [2] do tří oblastí, Obr. 1:

i. Učení s učitelem

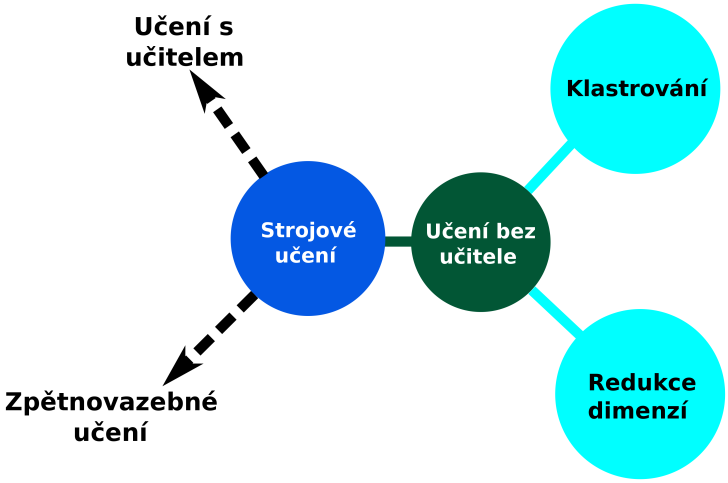
Je prvním typem strojového učení, kde jsou pro natrénování neuronových sítí použita označená data, kde jsou vstupní a výstupní hodnoty známy. Algoritmy zde pracují tak, že porovnávají vstupní a výstupní data a na základě jejich rozdílů upravují váhy u jednotlivých neuronových jednotek . Data se zde rozdělují do dvou částí: První, na kterých se algoritmus učí a druhá, na kterých se pak algoritmus testuje.

ii. Zpětnovazebné učení

V případě tohoto přístupu nejsou algoritmu poskytnuta žádná data. Nasazená metoda je učena na základě akcí, reakcí a případných odměn. Nalézá široké uplatnění v robotice, navigaci a herním průmyslu.

iii. Učení bez učitele

Je třetím typem oblasti strojového učení, kde jsou pro trénování neuronových sítí použita neoznačená data. Cílem je zde nalézat pomocí různých metod struktury v datech. Tyto metody nalézají aplikace tam, kde je potřeba neuspořádaná data klasifikovat a vizualizovat do smysluplné podoby. Tato třetí oblast je i tématem této bakalářské práce.



Obr. 1: Schéma rozdělení směrů výpočetní inteligence, v této práci je věnována pozornost odvětví Učení bez učitele

Algoritmy strojového učení bez učitele jsou široce používány pro klasifikaci, shlukování dat, redukci dimenzí datasetů, datovou analýzu a vizualizaci v oblasti těžby znalostí z dat, které jsou se stále rostoucím trendem používány na čím dál více rostoucí objemy ukládaných dat. S velkými objemy ovšem roste i požadavek na jejich co nejrychlejší zpracování, čehož ale není možné dosáhnout v dostatečně krátkém čase za pomocí běžných výpočetních jednotek. Vedle možnosti sestavení specifických výpočetních jednotek určeným svou architekturou (CPU, GPU) na práci s velkými objemy dat jsou zde i samoučící algoritmy pro jejich rychlé zpracování [3].

1 Neuronové sítě

Neuronové sítě jsou složeny z algoritmů, které jsou designovány k rozpoznávání skrytých souvislostí a struktur v daných datasetech. Myšlenka za vytvořením tohoto přístupu leží v napodobení funkce získávání lidské zkušenosti procesem učení se. Neuronové sítě se

dokáží přizpůsobovat změnám na ve vstupních datech a snaží se nalézt nejlepší možné řešení problému bez možnosti upravovat výstupní kritéria [4].

Neuronové sítě sestávají z několika části, zde je uvedeno rozdělení dle [5]:

- **Neuron:** Je základní jednotkou neuronové sítě. Do neuronu vstupuje určitý počet vstupních dat.
- **Neuronové propojení:** Propojuje jednotlivé neurony mezi sebou napříč vrstvami. Toto propojení je svázáno s váhou neuronu. Váhy těchto neuronů jsou měněny a aktualizovány při procesu učení na základě vstupních hodnot (či výstupních hodnot jiných neuronů)
- **Aktivační funkce:** Aktivační funkce v neuronových sítích je podmínka, zda se má daná neuronová jednotka aktivovat, či nikoliv.
- **Vstupní vrstva:** První vrstva neuronové sítě, která přeposílá vstupní data neuronům ve vrstvách následujících. Žádným způsobem signál nemodifikuje a není opatřena váhami
- **Výstupní vrstva:** Poslední vrstva neuronové sítě
- **Skrytá vrstva:** Vrstva či vrstvy, které obsahují neurony, které již modifikují vstupní data.
- **Váhy neuronů:** Váhy reprezentují sílu propojení mezi jednotlivými neurony.

2 Průmyslová data

S rozmachem 4. průmyslové revoluce označené jako Průmysl 4.0, kde je snaha vládních a soukromých společností o urychlení robotizace a nasazení moderních plně automatických řídících systémů nezávislých na lidské obsluze [6], je nedílnou součástí toho nového směru i sběr, ukládání a vyhodnocování celého spektra informací, nazvaného průmyslová data. Je zde kladen důraz na nasazení dostatečného množství měřících přístrojů a senzorů pro sběr dat jako takových, dále mít k dispozici dostatečně velká a bezpečná datová úložiště a následně nasazení vhodných metod pro jejich zpracování a analýzu.

Detailní monitoring výroby je jedním z hlavních nástrojů konkurenceschopnosti výrobců. Velká řada výrobců již poptává pro své výrobní závody senzoriky založené na

technologii IoT (Internet of Things) společně s monitorovacími řešeními typu SCADA (Supervisory Control And Data Aquisition), řešení uskladnění dat například pomocí konceptu Big data a jejich následné předzpracování, zpracování a analýzu. Tato bakalářská práce shrnuje některé nástroje právě pro předzpracování a zpracování takovýchto dat.

2.1 Big Data

Koncept Big data není označení pro jedinou technologii, ale pro kombinaci starších a novějších technologií, které napomáhají společnostem získat z dat požadované informace. Big data je schopnost pracovat s velkými množstvími různorodých dat s požadovanou rychlostí zpracování aby bylo možné na základě analýzy pružně a včasně reagovat na různorodé situace (např. úprava parametrů výrobní linky a podobné). Tento koncept by se dal dle [7] popsat třemi charakteristikami (v anglické literatuře označené three Vs):

- **Objem (Volume):** Celková velikost dat
- **Rychlost (Velocity):** Jak rychle jsou data zpracovávána
- **Různorodost (Variety):** Počet různých datových tříd

Autor [8] uvádí navíc další dvě charakteristiky:

- **Pravdivost (Veracity):** Udává mírů neuspořádanosti dat a jejich pravdivostní hodnotu
- **Hodnota (Value):** Vypovídá o tom, jakým způsobem mohou být data převedena na hodnoty

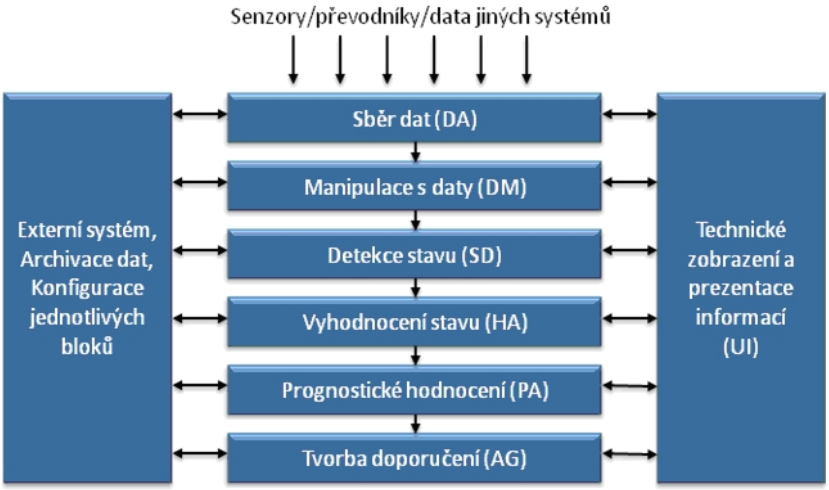
S daty jako takovými se v dnešní době pracuje v téměř každém odvětví průmyslu a jejich management je klíčovým faktorem pro produktivitu dané společnosti. Je zde hned několik důvodů, proč má smysl uvažovat použití konceptu Big data. Za prvé pomáhá nahlédnout na informace v datech obsažených s vysokou rychlostí (v porovnání se staršími metodami). Za druhé mohou společnosti sbírat mnohem větší množství přesnějších dat v digitální podobě a na základě jejich analýz a kontrolovaných experimentů optimalizovat svou produkci. Za třetí Big data umožňují stále užší zaměření se na konkrétního zákazníka a tak stále jemu více přesně přizpůsobené produkty a služby. Za čtvrté Big data může být použit pro podporu výzkumu dalších generací produktů a služeb. Například výrobci využívají data ze senzorů umístěných v produktech k nalezení poprodejních vylepšení a zároveň tak mohou obstarávat jejich aktivní údržbu.

Jelikož se jedná o poměrně mladý koncept, je potřeba k němu přistupovat s opatrností a vyřešit některé problémy v oblasti soukromí, bezpečnosti a řešení odpovědnosti, zejména pak, jedná-li se o komerční sféru [9].

Zde je uvedeno několik společností, které již úspěšně implementovaly koncept Big Data: IBM, Hewlett Packard Enterprise, Google, Amazon, Oracle a další [10].

2.2 Norma ISO 13374

V současné době existuje celá řada možností, jak si může zákazník sestavit vlastní systém pro sběr a vyhodnocování dat a vytváření následných prognostických hodnocení. Problematika údržby podle technického stavu a kompatibilita komponent tohoto systému je cílem normy ISO 13374, která udává požadavky na specifikaci softwarových nástrojů, použitých pro efektivní údržbu. Je zde snaha tedy tyto systémy obecně unifikovat, pro snadnější předávání informací a dat z monitorovacích systému, jejich následného zpracování, vyhodnocení a doporučení. Na Obr. 2 je zobrazeno schéma normy ISO 13374,



Obr. 2: Zpracování informací a tok dat dle normy ISO 13374. Převzato z [11].

Hlavními bloky z Obr. 2 jsou pro technickou diagnostiku blok DA (Data Aquisition), DM (Data Manipulation), SD (State Detection), HA (Health Assesment) a pro technickou prognostiku blok PA (Prognostic Assesment), které slouží pro zlepšování bezpečnosti, plánování provozu, údržby a minimalizaci odstavení zařízení. [11] Součástí normy je i blok AG (Advisory Generation), který se spoléhá na expertní systém, či Neuro-Fuzzy, což je

spojení Fuzzy logiky a strojového učení. Dále norma udává, jak jsou spolu jednotlivé bloky propojeny s grafickým uživatelským rozhraním a jejich propojení na externí archivační systémy [12].

3 Principy metod pro data

Obecně je v oblasti strojového učení možné použít mnoho metodik a přístupů, jak průmyslová data zpracovávat a analyzovat. Tématem této bakalářské práce jsou samoučící algoritmy pro analýzu dat a jako takové se v oblasti strojového učení vyskytují dva hlavní směry či způsoby předzpracování a zpracování dat:

- **Redukce dimenzí**
- **Klastrování**

3.1 Redukce dimenzí

Redukce dimenzí je velmi často považována jako metoda předzpracování dat a je často používána pro následné klastrování, klasifikaci a mnoho dalších aplikací strojového učení a těžby dat. Metoda hraje významnou roli například v oblastech jako je zpracování obrazu, analýza průmyslových dat, biologie, výzkum klimatických změn a mnoho dalších, jelikož velikost objemu dat v nich bývá značná. [13] V daném datasetu je možné převést data z vysoko dimenzionálního prostoru do prostoru s nižším počtem dimenzí se zachováním dostatečné vypovídající informace daných dat, v mnoha případech tak lze značně zkrátit výpočetní čas následující analýzy.

Existuje celé spektrum přístupů, kterými se dá redukce dimenzí datasetu realizovat, v této práci jsou představeny následující metody:

- **PCA (Principal Component Analysis)**
- **LDA (Linear Discriminant Analysis)**
- **Autoenkodéry**
- **t-SNE (t-Distributed Stochastic Neighbor Embedding)**

3.2 Klastrování

Klastrování je metoda zpracování dat, která rozděluje data neuspořádaného datasetu do shluků, podle vlastností a na nich založené podobnosti datových bodů. Jinými slovy body se stejnými nebo podobnými body jsou umístěny a označeny do jednoho shluku, kdežto data s jinými, odlišnými, vlastnostmi do shluku druhého, zpravidla Euklidovsky vzdáleného.

Klastrování hraje významnou roli v datové analýze pro vytváření organizovaných struktur neuspořádaných dat. Vždy záleží na uživateli, které kritérium pro klastrování zvolí, respektive podle které vlastnosti se mají následně data uspořádat.

Tuto metodu lze obecně rozdělit dle [14] na:

- **Založené na hustotě dat:** Přístup je založen na shlukování dat podle hustoty jejich vlastností. Pomocí této metody lze dosáhnout dobré přesnosti. Například DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify Clustering Structure)
- **Založené na hierarchii:** Metoda vytváří shluky stromové struktury založené na hierarchii dat, kde nové jsou tvořeny za pomocí předchozích. Lze rozdělit na dva přístupy: Aglomerativní (zespoda nahoru) a Divisivní (seshora dolů). Například CURE (Clustering Using Representatives), BIRCH (Balanced Iterative Reducing and using Hierarchies)
- **Založené na rozdělení:** Tento přístup rozděluje data do k (zadané na počátku uživatelem) shluků a každé takové rozdělení vytvoří právě jeden shluk. Například K-Means, K-Medoids, CLARANS (Clustering Large Applications based upon randomized Search)
- **Založené na sítí:** Datový prostor je zde formován do konečného počtu bodů, které tvoří síť. Veškeré shlukovací operace jsou na těchto sítích rychlé a nezávislé na počtu datových objektů. Například STING (Statistical Information Grid), CLIQUE (CLustering In QUEst), SOM (Self-Organizing Maps)

4 Použité metody

V této bakalářské práci bylo vybráno několik známých metod pro předzpracování a zpracování dat v oblasti výpočetní inteligence. Některé z těchto metod pak byly vybrány a jejich účinnost byla demonstrována na reálných či umělých datech.

4.1 PCA

Metoda hlavních komponent (PCA – Principal Component Analysis), je metoda založená na redukci dimenzí, která se s výhodou používá pro redukci dimenzí mnohorozměrných dat. Cílem tohoto procesu je zmenšit objem dat takovým způsobem, aby se zachovalo pokud možno co největší množství informace v něm obsažených. Je zde tedy cílem připravit data pro jejich jednoduší a rychlejší zpracování se zachováním dostatečné vypovídající informace [15]. Princip této metody sestává dle autora [16] z pěti kroků:

i. Standardizace

V tomto kroku se standardizuje rozsah počátečních proměnných tak, aby každá z nich přispívala rovnoměrně k analýze. Standardizaci je nutné provést neboť metoda PCA je citlivá na výchylky v počátečních proměnných. Konkrétněji, hodnoty s vyšším rozsahem by měly později vyšší váhu než ty s nižším, což by vedlo ke zkresleným výsledkům. Jako standardizační nástroj se používá metoda Z-Score

$$z_i = \frac{x_i - \overline{x}}{s}, \tag{4.1}$$

kde:

- $\overline{x}$ je střední hodnota všech dat
- s je směrodatná odchylka

Tento proces zaručuje transformaci všech počátečních proměnných do jednotné škály.

ii. Výpočet kovarianční matice

V dalším kroku následuje výpočet kovarianční matice s cílem najít ty hodnoty, které jsou si podobné, co se hodnoty informace týče. Kovarianční symetrická čtvercová matice s rozměrem *p x p*, kde *p* je rozměr udávající počet dimenzí

$$\begin{pmatrix} Cov(a_1, a_1) & \cdots & Cov(a_1, a_p) \\ \vdots & \ddots & \vdots \\ Cov(a_p, a_1) & \cdots & Cov(a_p, a_p) \end{pmatrix}, \tag{4.2}$$

kde kovariance hodnoty se sama sebou je její variance (kvadrát směrodatné odchylky). Kovariance hodnoty s hodnotou různou popisuje jejich vzájemný vztah co se závislosti týče. Mohou nastat celkem 3 případy:

- **Kovariance je kladná** – hodnoty klesají a stoupají společně (korelace)
- **Kovariance je záporná** – hodnoty klesají a stoupají navzájem opačně (inverzní korelace)
- **Kovariance je nulová** – hodnoty spolu nejsou v korelaci

iii. Výpočet vlastních čísel a vektorů kovarianční matice

K určení hlavních komponent je potřeba vypočítat hlavní čísla a příslušné hlavní vektory kovarianční matice. Vlastní vektory dané soustavy slouží jako budoucí hlavní komponenty, u kterých je posuzována velikost vlastních čísel, respektive čím vyšší vlastní číslo, tím více informace nese jemu náležící vlastní vektor

$$|\lambda_1| > |\lambda_2|, \tag{4.3}$$

Dle rovnice (4.3) bude tedy vlastní vlastní vektor $\overrightarrow{v_1}$ nést větší podíl informace než vektor $\overrightarrow{v_2}$. Procentuální míru obsažené informace v k-té hlavní komponentě lze vypočíst dle

$$\frac{\lambda_k}{\sum_{i=1}^{n} \lambda_i}, \tag{4.4}$$

kde n je počet všech vlastních čísel.

Hlavní komponenty jsou pak nové proměnné vytvořené jako lineární kombinace počátečních proměnných takovým způsobem, že v prvních komponentách je uloženo největší množství informace. Geometricky interpretováno tak, že první hlavní komponenta obsahuje nejvyšší možné variance hodnot v počítaném objemu dat. Druhá hlavní komponenta se počítá obdobným způsobem ovšem s podmínkou, že není v korelaci s první hlavní komponentou. Pokud je počáteční dimenze dat *p* bude počet hlavních komponent rovněž *p*.

iv. Vektor příznaků

Pojem vektor příznaků označuje matici, která je složená z hlavních komponent ve sloupcích. Jejich počet se stanovuje dle požadované přesnosti podle vztahu (4.4). Odstraněním těch komponent, které nesou dle zadání zanedbatelné množství informace dochází k redukci redukci dimenzí datasetu.

Pro převedení datasetu zpět na použitelnou formu je třeba provést poslední operaci vynásobení transponovaného vektoru příznaků s maticí standardizovaných dat

$$D = F^T * S, \tag{4.5}$$

kde:

- D je matice redukovaných data
- F^T je transponovaný vektor příznaků
- S je matice standardizovaných dat

Tímto je tedy dosaženo redukce dimenzí datasetu s dostatečnou vypovídající informací pomocí metody PCA.

Algoritmus PCA je mimo jiné schopen potlačit šum v datech, jak bude ukázáno v praktické části této práce.

4.2 LDA

Metoda LDA (Linear Discriminant Analysis), je další metodou pro předzpracování dat, která se hojně využívá v jejich následné analýze. Stejně jako metoda PCA, LDA redukuje dimenze daného datasetu za pomocí vlastních čísel a vektorů, kde při vhodném zvolení zachovává dostatečně velkou informační hodnotu [17].

Na rozdíl od metody PCA, která používá kovarianční matici, metoda LDA pracuje s maticemi rozptylu [18]. Popis metody dle autora [19]: Uvažujme dataset s daty různých tříd (například teploty, tlaky a podobné). Nejprve je nutné napočítat matici rozptylu uvnitř každé datové třídy

$$S_W = \sum_{i=1}^{n} S_i, \tag{4.6}$$

$$S_i = \sum_{k=1}^{n} (x_k - m) * (x_k - m)^T, \tag{4.7}$$

kde m je vektor průměru

$$m = \frac{1}{n} \sum_{k=1}^{n} x_k \tag{4.8}$$

a rovněž matici rozptylu mezi jednotlivými datovými třídami

$$S_B = \sum_{k=1}^{n} N_k (m_k - m) * (m_k - m)^T, \tag{4.9}$$

kde m_k je průměr vzorků dané třídy a N_k je její velikost. Následně se určí celková matice rozptylu S dle

$$S = S_W^{-1} * S_B. \tag{4.10}$$

U té se vypočítají vlastní čísla $\lambda_1 ... \lambda_n$ a k nim příslušné hlavní vektory $v_1 ... v_n$, kde vlastní vektory nazveme lineárními diskriminanty datasetu. K výběru vektorů se nyní přistoupí stejně jako v případě metody PCA, tedy kdy absolutní velikost vlastních čísel vypovídá o „důležitosti“ lineárních diskriminantů.

Následně se z vybraných lineárních diskriminantů utvoří matice W tak, že vybrané vlastní vektory utvářejí sloupce. Poslední operace sestává z vynásobení matice původních dat Xs maticí W

$$\mathrm{Y} = \mathrm{X}^*\mathrm{W}. \tag{4.11}$$

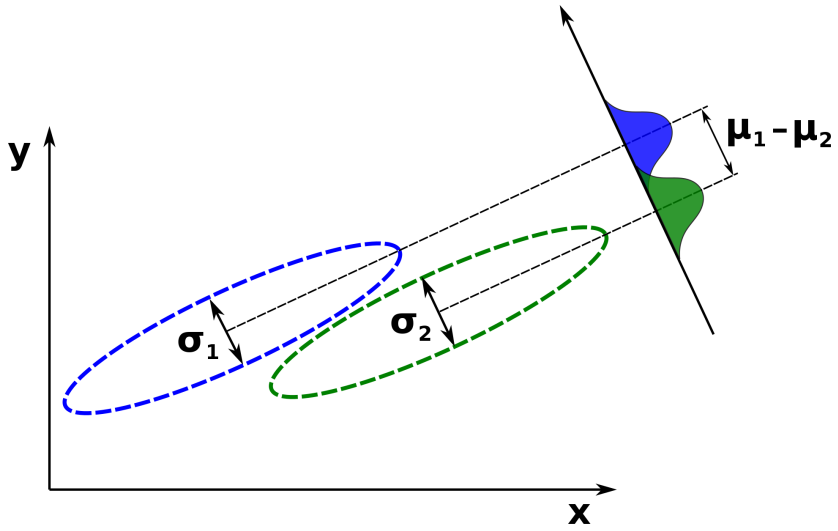
Tím tedy bylo dosaženo redukce dimenzí datasetu pomocí metody LDA. Pro jednodušší pochopení principu metody je zde uvedena demonstrace redukce dimenzí ze dvou do jedné, Obr. 3. Mějme dvě třídy dat, které budou převedeny ze souřadného systému xy do nového, jedno rozměrného systému. V tomto případě lze převod popsat rovnicí

$$max \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}. \tag{4.12}$$

Tato rovnice popisuje metodu LDA tak, že je zde snaha dosáhnout dvou podmínek:

- Co největší rozdíl průměrů projektovaných tříd $max\,(\mu_1 - \mu_2)^2$
- Co nejmenší variance každé třídy $min\left(\sigma_1^2 + \sigma_2^2\right)$

Tímto způsobem je pak možné najít co nejvýhodnější nový prostor, kam lze data projektovat a tím dosáhnout redukce dimenzí.



Obr. 3: Schéma příkladu použití metody LDA pro převod dvou dimenzionálních dat na jedno dimenzionální

4.3 t-SNE

Metoda t-SNE (t-Distributed Stochastic Neighbor Embedding) je poměrně nová, nelineární technika pro redukci dimenzí datasetu a vizualizaci vysoko rozměrných dat. Hojně se využívá v oblastech zpracování obrazu, NLP (Natural Language Processing), v aplikacích genomiky a zpracování řeči [20]. t-SNE zpracovává vysoko rozměrná data do obvykle dvou či tří dimenzionálního prostoru, kde jsou body s podobnými vlastnostmi uskupeny do shluků. Metoda je založena na vytváření a porovnávání pravděpodobnostních distribucí mezi dvěma body vždy v původním, mnoha rozměrném, prostoru a následně v prostoru novém, který je již dimenzionálně redukován. Pokud jsou si distribuce podobné, jsou body přiřazeny do stejného shluku v novém prostoru [21]. t-SNE minimalizuje KL (Kullback-Leibler) divergenci mezi dvěma pravděpodobnostními distribucemi, což je nesymetrická míra rozdílu mezi dvěma pravděpodobnostmi [22].

Metoda konvertuje vysoko dimenzionální Euklidovské vzdálenosti mezi datovými body do podmínkových pravděpodobností, které představují podobnosti mezi body. Podobnost mezi bodem x_j a x_i je podmíněná pravděpodobnost $p_{j|i}$ tak, že bod x_i je vybrán jako soused x_j právě tehdy, když souhlasí poměry hustot pravděpodobností pod Gaussovou křivkou se

středem v bodě x_i. Pro u sebe blízko umístěné body je pravděpodobnost $p_{j|i}$ poměrně vysoká, zatímco pro vzdálené body je téměř nulová. Pravděpodobnost $p_{j|i}$ je dána vztahem

$$p_{j|i} = \frac{exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} exp\left(-\frac{\|x_j - x_k\|^2}{2\sigma_i^2}\right)}, \tag{4.13}$$

kde σ_i je rozptyl Gaussova rozdělení se středem v bodě x_i [23]. V novém, o dimenze redukovaném, prostoru se již nepočítá s pravděpodobnostmi založenými na Gaussově rozdělení, ale na Cauchyho rozdělení (Studentovo t-rozdělení), neboť body by zde měly tendenci se překrývat kvůli redukci dimenzí [24]. Proto je tedy v níže dimenzionálním prostoru vyjádřena podmíněná pravděpodobnost $q_{j|i}$ vztahem

$$q_{j|i} = \frac{exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} exp\left(-\|y_j - y_k\|^2\right)}, \tag{4.14}$$

Optimalizace metody je provedena metodou gradientového spádu na KL divergenci mezi rozděleními $p_{j|i}$ a $q_{j|i}$. Gradient je vyjádřen jako

$$\frac{\delta J}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}. \tag{4.15}$$

Gradient zde vyjadřuje sílu a směr atrakce mezi dvěma body. Kladný gradient reprezentuje atraktivitu mezi body, záporný jejich odpudivost. Takto je dosaženo redukce dimenzí a vytváření bodových shluků pomocí metody t-SNE.

4.4 K-means

Algoritmus K-means je známá metoda pro rozdělení n bodů, které leží v d-dimenzionálním prostoru do k-shluků. V současné době je to jeden z nejpoužívanějších nástrojů pro klasifikaci dat.

Pro zpracování daného datasetu metodou K-means je nejdříve potřeba uživatelem zadat algoritmu počet klastrů k, (existuje řada metod, jak vhodně zvolit počáteční počet shluků k, k nalezení například [25]), jejichž polohy center jsou v prostoru s neuspořádanými daty umístěny náhodně. U každého datového bodu jsou pak určeny Euklidovské vzdálenosti od center všech shluků. Vzdálenost od centra, která je nejmenší ze všech možných, určuje

příslušnost daného bodu k danému shluku. Dále se postupně přepočítávají polohy center na základě průměrů poloh bodů, které náleží určitému shluku, jinými slovy, nalezne se nové „těžiště“ geometrického objektu a proces se opakuje, dokud systém nedosáhne stability, to znamená není již žádný bod, který by byl přiřazen do jednoho ze shluků [26].

4.4.1 Formální postup K-means

Pro každý shluk je definován právě jeden centroid (bod, který reprezentuje těžiště geometrického objektu vytvořeného daty). V prvním kroku jsou data přiřazena k nejbližšímu shluku na základě nejmenší čtvercové Euklidovské vzdálenosti. Pokud c_i je sada centroidů v množině C, pak každý datový bod je přiřazen ke shluku dle následujícího vztahu

$$arg\left(min\left(\|(c_i, x)\|^2\right)\right), \tag{4.16}$$

kde $\|.\|$ je Euklidovská vzdálenost. Nechť pro každé přiřazení bodů pro každý i-tý shluk je jeho centroid označen S_i. Dále jsou aktualizovány polohy center datových shluků, to je dosaženo přepočítáním centra geometrického objektu, vytvořeného daty, které klastr pojal podle vztahu

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i. \tag{4.17}$$

Algoritmus dále iteruje mezi přiřazováním dalších datových bodů ke shlukům a následný změnám jejich center, dokud není systém stabilní, což v tomto případě znamená buďto dosažení stavu, kdy již nelze žádný bod přiřadit k jinému shluku nebo kdy byl dosažen maximální počet iterací zadaný uživatelem [27].

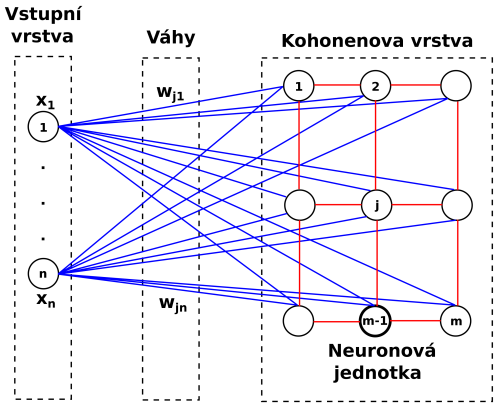
4.5 SOM

Samo-organizační mapa (SOM – Self-Organizing Maps) je samo učící se algoritmus, který představil Teuvo Kohonen. Tento algoritmus je výborným nástrojem pro analýzu dat, díky své schopnosti data projektovat na mapy s dimenzemi nižšími, než byly dimenze původní. SOM sestávají z jedno či dvou dimenzionální sítě neuronů, kde každý z váhových vektorů příslušného neuronu má stejný rozměr jako vektor vstupních dat.

Učící princip metody se zakládá na konkurenčním učení, kdy neuronový aktivátor je funkcí vzdálenosti mezi váhou neuronu a vstupním vektorem. Aktivovaný neuron se tak učí

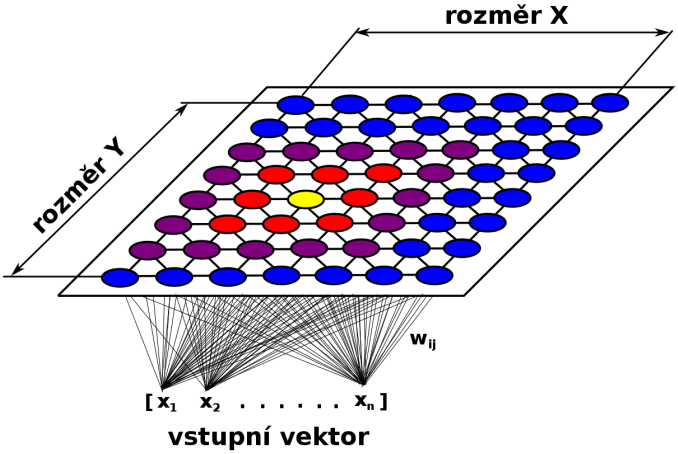
nejrychleji a jeho váha je následně modifikována. Pokud se postupně v datech nalezne podobná závislost, jako v prvním případě, je neuron aktivován znovu, což vede k výhře dané neuronové jednotky a zvětšení míry jejího naučení, oproti neuronům ostatním [28]

Díky tomuto kolektivnímu a spolupracujícímu učení a za předpokladu, že neuronová síť je dvourozměrná, se různé neurony naladí na různé vstupy a vstupní data se tak na základě svých vlastností zobrazí se specifickými souřadnicemi na dvourozměrné mapě. Zejména pokud jsou data seskupena hierarchicky, bude reprezentovaná generovaná struktura velmi explicitní [29]. Teoreticky může být SOM ve výsledku mnoha dimenzionální mapa, avšak pro účely vizualizace je ve většině případů dvou dimenzionální [30].



Obr. 4: Schema Samo-organizační mapy s vstupní vsrtvou, vrstvou vah a Kohonenovou vrstvou

Výhoda samo-organizačních map, jakožto nástroje pro redukci a klastrování dat, spočívá ve schopnosti topologicky uspořádat data na základě vzájemných souvislostí a jejich projekce na dvou-dimenzionální mapu. Data se specifickou vzájemnou informační vazbou jsou topologicky shluknuta do jednoho clusteru, zatímco data s jinou vazbou jsou shluknuta do clusteru druhého, Euklidovsky vzdáleného od předchozího. Takto lze například nahlédnout na to, která data se kterou informací jsou v datasetu dominantní. Statisticky méně vyskytující se body v datasetu jsou zobrazeny jako menší shluky než ty body, které se zde vyskytují četněji [31]. Obr. 5 zobrazuje aktivovaný shluk neuronů, reagující na konkrétní vstupní vektor.



Obr. 5: Nazanačené schéma SOM s vizualizací neuronové vrstvy. Po naučení se reaguje mapa na vstupní vektor aktivováním naučených neuronových jednotek. S výhodou se používá pro vizualizaci vysoko rozměrných dat do 2-D mapy. Překresleno z [47]

Samo-organizační mapa odráží vnitřní strukturu trénovacích dat, nemůže být však jasně ukázáno, který neuron je aktivován kterým vstupním vektorem, navíc neurony natrénovány na jeden konkrétní typ vstupů mohou reagovat na jiný, odlišný, typ vstupních vektorů. Díky tomuto fenoménu musí být SOM kalibrována. Toho lze dosáhnout předkládáním známých příkladů neuronové síti a zaznamenáváním těch neuronů, které jsou aktivovány daným vektorem. Neurony, které nejsou aktivovány v procesu kalibrace, jsou interpretovány interpolací [32].

4.5.1 Algoritmus SOM

Uvažujme Samo-organizační mapu sestavenou z m neuronů umístěnou na běžné dvou dimenzionální mapě. Tyto neurony jsou propojeny se sousedními dle topologických propojení. Existují dva běžné typy propojení a to obdélníkové a hexagonální. Každému neuronu i náleží d-dimenzionální vektor vah $w = (w_{i1}, w_{i2}, ..., w_{id})$ kde $i = 1, 2, ..., m$, který má stejný rozměr, jako vstupní vektor. Konvenční SOM učící algoritmus může být pak vysvětlen v následujících krocích:

i. Inicializace váhových vektorů w_i sítě sestavené z $m \times n$ neuronů

ii. Vybrání náhodného vstupního vektoru $x(t)$ jakožto vstupu pro všechny neurony v síti paralelně ve stejném čase

iii. Nalezení „vítězného“ neuronu BMU (the Best Match Unit) za pomocí následující rovnice

$$c = arg\,(min\,\{\|w_i(t) - x(t)\|\}), \tag{4.18}$$

kde $\|.\|$ je Euklidovská vzdálenost, $x(t)$ a $w(t)$ jsou vstupní a váhový vektor neuronu i v iteraci t.

iv. Váhový vektor neuronu je aktualizován dle

$$w_i(t+1) = w_i(t) + h_{c,i}(t)\,[x(t) - w_i(t)], \tag{4.19}$$

kde $\mathrm{h}_{c,i}(t)$ je Gaussovská funkce pro sousední neurony dána

$$h_{c,i}(t) = \alpha(t) * exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right), \tag{4.20}$$

kde r je souřadnice polohy neuronu na mapě, $\alpha(t)$je míra naučení se, a $\sigma(t)$ je rádiusová vzdálenost od sousedního neuronu. Funkce $\alpha(t)$ a $\sigma(t)$ klesají obě monotónně dle následujících vztahů

$$\alpha(t) = \alpha(0)\left(\frac{\alpha(T)}{\alpha(0)}\right)^{\frac{t}{T}} \text{ a} \tag{4.21}$$

$$\sigma(t) = \sigma(0)\left(\frac{\sigma(T)}{\sigma(0)}\right)^{\frac{t}{T}}, \tag{4.22}$$

kde T je trénovací čas.

v. Pro všechny vstupní vektory se opakují kroky ii. až iv [28].

4.6 Autoenkodéry

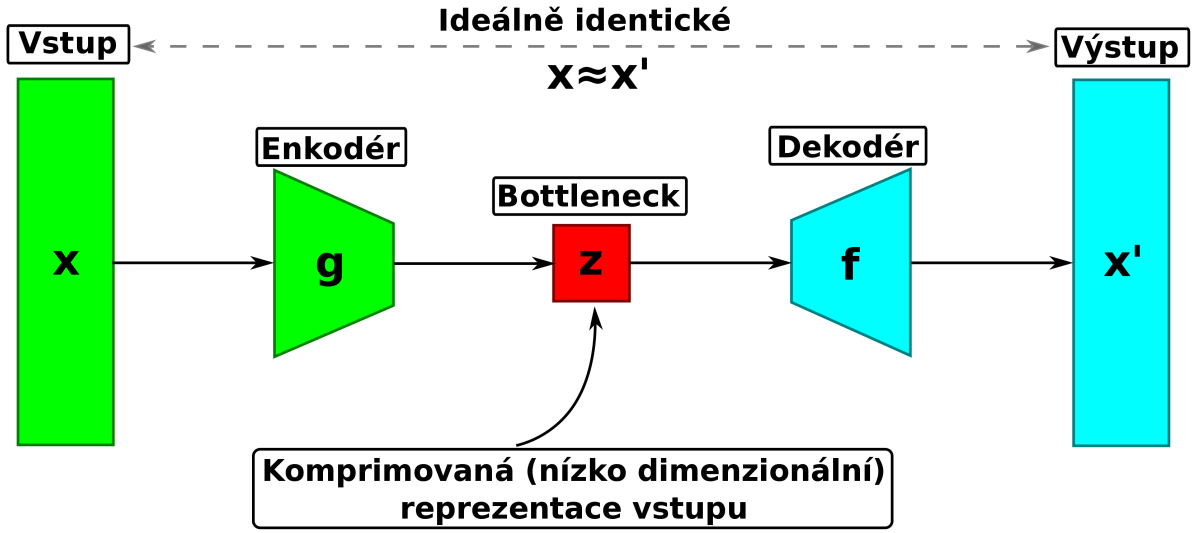
Autoenkodéry jsou typy neuronových sítí, které komprimují a následně dekomprimují vstupní data s pokud možno co nejmenší ztrátou informace v nich obsažených. Odstranění šumu v datech a redukce dimenzí datasetů jsou dnes dvě hlavní využití těchto modelů. Autoenkodéry jsou učeny automaticky z datových vzorků, je poměrně jednoduché naučit tyto neuronové sítě na specifická vstupní data. Algoritmy Autoenkodérů jsou naučený na uchování co největšího množství informace při kompresi a následné dekompresi dat [33]. Dle [34]sestávají ze čtyř částí:

i. **Enkodér:** V této části je model redukuje dimenze daného datasetu a komprimuje vstupní data do kódované podoby.

ii. **Bottleneck:** Je vrstva, která obsahuje komprimovaná data. Data jsou zde redukovaná na co nejmenší možnou hodnotu dimenzí.

iii. **Dekodér:** Zde model rekonstruuje data do původní podoby respektive po možných úpravách je převádí do prostoru se stejným počtem dimenzí, jako byl počet dimenzí prostoru vstupních dat.

iv. **Ztráta při rekonstrukci:** Tato metoda měří, jak efektivně Autoenkodér zpracovává data porovnáváním vstupní vrstvy s výstupní.



Obr. 6: Schéma Autoenkodéru se vstupním vektorem, kódovou vrstvou, bottleneckem, dekódovací vrstvou a výstupním vektorem. Autoenkodér se učí porovnáváním svtupu s výstupem a upravováním vah jednotlivých neuronů enkodéru a dekodéru.

4.6.1 Formální popis autoenkodéru

Mějme dataset trénovacích dat $\left\{x^{(1)}, x^{(2)}, x^{(3)}, ...\right\}$, kde $x^{(i)} \in R^n$. Pomocí Autoenkodéru je zde snaha dosáhnout $y^{(i)} = x^{(i)}$, tedy že vstup by měl být ideálně roven výstupu. Autoenkodér se snaží naučit právě takovou převodní funkci, aby platilo $x \approx \widehat{x}$. Označme $a_j^{(i)}(x)$ aktivaci neuronu j ve skryté vrstvě Autoenkodéru. Nechť je dále $\widehat{\rho_j}$ průměrná aktivace neuronu j

$$\widehat{\rho_j} = \frac{1}{n} \sum_{i=1}^{n} \left[a_j^{(i)} \left(x^{(i)} \right) \right]. \tag{4.23}$$

Je zde snaha dosáhnout této rovnosti

$$\widehat{\rho_j} = \rho, \tag{4.24}$$

kde ρ je parametr řídkosti, obvykle téměř roven nule. K dosažení rovnosti (4.24) musí být aktivace neuronu ve skryté vrstvě rovněž téměř rovna nule. K dosažení takové rovnosti slouží vztah (4.25), který penalizuje taková $\widehat{\rho_j}$, která se významně liší od parametru ρ

$$\sum_{j=1}^{s} \rho \; log \left(\frac{\rho}{\widehat{\rho_j}} \right) + (1 - \rho) \, log \frac{1 - \rho}{1 - \widehat{\rho_j}}, \tag{4.25}$$

kde s je počet neuronů ve skryté vrstvě. Taková penalizace je založena na KL divergenci, vztah (4.25) lze přepsat na

$$\sum_{j=1}^{s} KL \left(\rho || \widehat{\rho_j} \right), \tag{4.26}$$

kde

$$KL \left(\rho || \widehat{\rho_j} \right) = \rho \; log \left(\frac{\rho}{\widehat{\rho_j}} \right) + (1 - \rho) \, log \frac{1 - \rho}{1 - \widehat{\rho_j}} \tag{4.27}$$

je Kullback-Lieber divergence mezi Bernoulliho náhodou proměnnou [35] s průměrem ρ a Bernoulliho náhodnou proměnou s průměrem $\widehat{\rho_j}$. KL-divergence je funkce, která vyhodnocuje, jak moc rozdílné od sebe jsou dva rozptyly hodnot. Penalizující funkce $KL \left(\rho || \widehat{\rho_j} \right)$ je rovna nule právě tehdy, když $\widehat{\rho_j} = \rho$, jinak monotónně roste podle mezi $\widehat{\rho_j}$ a ρ . V případě Autoenkodéru je ztrátová funkce dána dle [36] jako

$$J_{ztr} \left(W, b \right) = J \left(W, b \right) + \beta \sum_{j=1}^{s} KL \left(\rho || \widehat{\rho_j} \right), \tag{4.28}$$

kde $J \left(W, b \right)$ kvadrát chyby ztrátové funkce a parametr β je váha penalizace $\widehat{\rho_j}$ [37]

4.7 K-Medoids

K-Medoids analýza (nazvaná také jako PAM (Partitioning Around Medoids)) byla představena v roce 1987 dvojicí Kaufman a Rosseeuw. Medoid je zde definován jako bod ve

shluku, jehož rozdílnost je statisticky co nejmenší možná vzhledem ke zbylým bodům v daném shluku [38].

K-Medoids analýza je ze své podstaty podobná K-Menas. Pro nalezení k shluků daného datasetu jsou z něj vybrány reprezentativní hodnoty pro každý budoucí klastr, které se nazvou medoidy [39]. Uživatelsky se pak zvolí počáteční počet k medoidů (shluků), kterým algoritmus náhodně určí souřadnice a do kterých se posléze daná data přiřazují. Data jsou přiřazena do klastru, k jehož medoidu mají co největší informační vazbu, co se týče podobnosti. Na rozdíl od metody K-Means, kde centrum či těžiště shluku nemusí být obsaženo v daných datech, medoid je vždy součástí vstupních dat. V každé iteraci mění medoidy svou polohu, přepočítávají rozdílnosti mezi daty a snaží se nalézt tyto rozdílnosti co nejmenší možné. Iterace pokračují, dokud žádný z medoidů nemění svou polohu. Počet k shluků je pak vytvořen kolem medoidů [40]. Snadné nahlédnutí při použití analogie metod K-Means a K-Medoid: Analýza K-Means počítá vždy s průměrnou hodnotou shluků dat, kdežto K-Medoid s jejich mediánem [41].

4.7.1 Formální popis metody K-Medoids

Autoři v [42] uvádí princip postupu algoritmu K-Medoids. Označme O_j jako nevybraný bod a O_i jako vybraný (medoid), pak O_j náleží do shluku reprezentovaném medoidem O_i právě tehdy, když

$$d\left(O_j, O_i\right) = min(d\left(O_j, O_c\right)), \tag{4.29}$$

kde minimum je hledáno na množině všech medoidů O_c a operace značena $d(O_a, O_b)$ určuje rozdílnost nebo vzdálenost mezi objekty O_a a O_b. Pro nalezení k shluků algoritmus v každém kroku provede výměnou operaci mezi vybraným objektem O_i a nevybraným objektem O_h, do té doby, dokud dané výměny přispívají ke shlukování. Konkrétně pak, pro výpočet efektivity takovýchto výměn mezi objekty O_i a O_h algoritmus počítá ztrátu O_{jih} pro všechny nevybrané objekty O_j. V závislosti na jednom ze 4 možných stavu je počítána ztráta:

i. První případ: Předpokládejme, že O_j již náleží shluku reprezentovaným O_i. Nechť dále O_j je více podobné $O_{j,2}$ než O_h, tedy že

$$d(O_j, O_h) \geq d(O_j, O_{j,2}), \tag{4.30}$$

kde $O_{j,2}$ je druhý nejpodobnější medoid objektu O_j. Takže pokud je O_i nahrazeno O_h, pak O_j bude náležet shluku reprezentovaným $O_{j,2}$. Ztráta při výměnné operaci je zde

$$C_{jih} = d(O_j, O_{j,2}) - d(O_j, O_i) \tag{4.31}$$

ii. **Druhý případ:** O_j náleží do shluku reprezentovaným O_i. V tomto případě je ale O_j méně podobné $O_{j,2}$ než O_h, tedy že

$$d(O_j, O_h) < d(O_j, O_{j,2}), \tag{4.32}$$

pak pokud je O_i nahrazeno O_h, O_j bude náležet shluku reprezentovaným O_h. Ztráta při výměnné operaci je pak

$$C_{jih} = d(O_j, O_h) - d(O_j, O_i) \tag{4.33}$$

iii. **Třetí případ:** O_j náleží do shluku jiného, než toho reprezentovaného objektem O_i. Nechť je $O_{j,2}$ reprezentant shluku, do kterého O_j náleží. Dále nechť je O_j více podobné $O_{j,2}$ než O_h. Pak i pokud je O_i nahrazeno O_h, zůstává O_j ve shluku reprezentovaného objektem $O_{j,2}$ a ztráta tedy

$$C_{jih} = 0. \tag{4.34}$$

iv. **Čtvrtý případ:** O_j náleží do do shluku reprezentovaným $O_{j,2}$, ale O_j je méně podobné $O_{j,2}$ než O_h. Pak vyměněním O_i za O_h způsobí přemístění objektu O_j do shluku s objektem O_h. Ztráta je pak

$$C_{jih} = d(O_j, O_h) - d(O_j, O_{j,2}). \tag{4.35}$$

Celková ztráta výměny O_i a O_h je pak napočítána jako součet nastalých kombinací i-iv

$$TC_{ih} = \sum_j C_{jih}. \tag{4.36}$$

Algoritmus dále pracuje ve 4 krocích:

1) Uživatelsky zadat počet shluků k

2) Výpočet TC_{ih} pro všechny páry objektů O_i, O_h, kde O_i jsou vybrané objekty a O_h nevybrané

3) Vybrání páru objektů O_i, O_h, které odpovídá $min(TC_{ih})$. Pokud je $min(TC_{ih})$ záporné, je O_i vyměněno za O_h a algoritmus se vrací k bodu 2)

4) Pro všechny nevybrané objekty jsou vybrány nejpodobnější reprezentativní objekty

4.8 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applicatons with Noise – Shlukování na základě hustoty s detekcí šumu), je analýza, která rozděluje data do shluků na základě jejich hustoty [43]. Algoritmus seskupuje objekty do shluku v místech, kde jsou body hustě umístěny (znamená body s mnoha blízkými sousedy). Jednou z hlavních výhod algoritmu je, že dokáže najít například shluk uvnitř shluku, který je tvořen prstencem (zde by metody, které byly v této práci zatím představeny, selhaly).

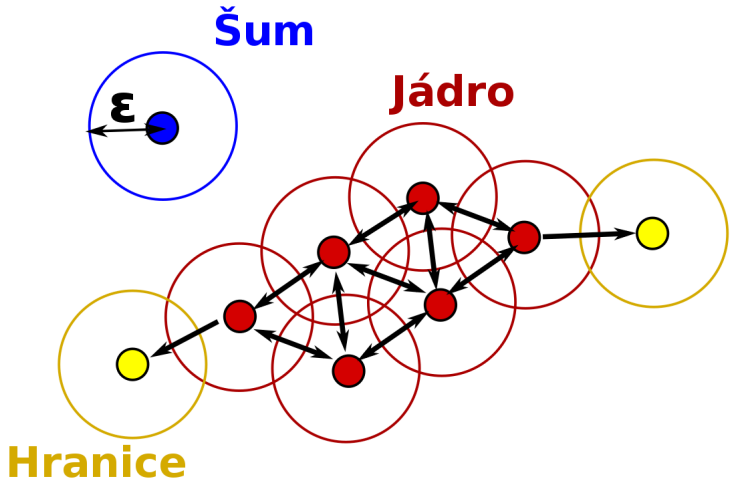
DBSCAN rozděluje body dle [44] na:

i. **Bod jádra *p*:** Bod, který má ve svém okolí dostatečné množství sousedních bodů.

ii. **Hraniční bod *q*:** Bod, který je přímo dosažitelný z bodu *p* (pokud leží ve vzdálenosti ϵ) a bod *p* je bodem jádra. Zároveň ale ve svém ϵ-okolí nemá dostatečný počet sousedních bodů, aby se stal bodem jádra.

iii. **Šum:** Body, které jsou Euklidovsky vzdálené o hodnotu větší, než je ϵ, jsou označeny jako body šumu.

Body jádra jsou základními prvky každého ze shluků. Parametry ϵ a $minPts$ (minimální počet sousedních bodu, pro vyhodnocení bodu jako objektu jádra nebo hranice) se uživatelsky nastavují na začátku výpočtu a jsou stejné v celém jeho průběhu.



Obr. 7: Schéma principu metody DBSCAN. Zpočátku je algoritmem vybrán náhodný bod, u něhož je prověřeno, zda v jeho okolí neleží bod jiný. Pokud ano, algoritmus prověří, počet takových bodů. Pokud je počet dostatečný, je bod označen součástí jádra, pokud není, je označen za bod hraniční. Pokud není v jeho okolí nalazen žádný bod, je označen za šum.

Algoritmus DBSCAN dle [45] pracuje v následujících krocích:

1) Je náhodně vybrán bod, který nebyl přiřazen k žádnému shluku anebo byl vyhodnocen jako šum. Spouští se kontrola, zdali je bod bodem jádra, resp. testuje se existence dalších bodů v jeho ϵ-okolí. Pokud je takový bod nalezen, začíná proces shlukování kolem tohoto bodu. Pokud nalezen není, je prohlášen za bod šumu.

2) Kolem nalezeného bodu jádra se vytváří shluk připojováním bodů, které leží v jeho přímé blízkosti. Pokud v tomto procesu je do shluku přidán bod šumu, je prohlášen bodem hranice.

3) Opakování kroku 1) a 2), dokud nejsou všechny body přiřazené k některému ze shluků nebo označeny jako body šumu.

5 Programovací jazyk Python

Jako programovací jazyk pro následující demonstrace vybraných metody byl zvolen Python. Python je skriptovací, objektově orientovaný, programovací jazyk. Díky své jednoduchosti se s oblibou používá pro Rapid Application Development, stejně jako pro

propojování již existujících systémů a komponent. Jsou zde podporovány knihovny a moduly, což podporuje modularitu programů a opakované použití kódů [46]. Další výhodou je, že se jedná o open source platformu s velmi rozšířenou komunitou. Vedle jazyka MATLAB a R je jazyk Python široce používán pro oblasti Data science a Strojového učení.

V této bakalářské práci byl použit Python ve verzi 3.7 s následujícími knihovnami a moduly:

- **Matplotlib** – knihovna umožňující práci s grafy
- **Pandas** – knihovna pro snadné zpracování dat a jejich analýzu
- **os –** knihovna pro manipulaci se soubory (načítání, ukládání)
- **Numpy –** výpočetní knihovna
- **Scikit-learn –** Knihovna zaměřená na strojové učení
- **SciPy –** výpočetní knihovna
- **TensorFlow –** Knihovna zaměřená na strojové učení

5.1 Vývojové prostředí Spyder

Jako vývojové prostředí byl zvolen Spyder 3.3.3. Toto vývojové prostředí je napsáno v jazyce Python a je s oblibou používáno ve vědeckých sférách, Data science nevyjímaje.

6 Data

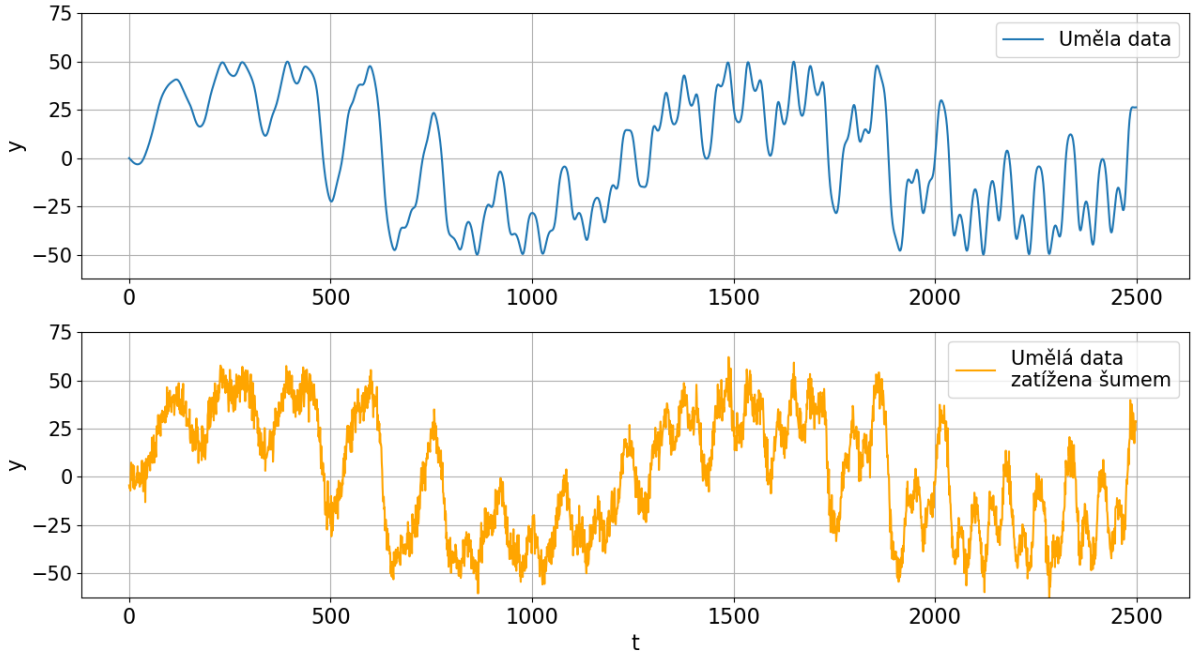
6.1 Umělá data

Pro testování algoritmů byla vytvořena umělá data, cílem zde bylo vytvořit jejich neperiodický průběh, který je podobný reálným vibracím. Daná data byla posléze zatížena umělým šumem přidáním náhodných hodnot k původnímu signálu tak, aby co nejlépe reprezentovala skutečné měřené hodnoty, kdy šum je vytvářen například nepřesností senzoru. Rovnice (6.1) popisuje uměle vytvořená data, jejich vizualizace je pak ukázána na Obr. 8.

$$y(t) = t * sin(0.5 * sin(2 * t) + 1.2 * sin(0.5 * t) - 2 * sin(8 * t)) -$$

$$-(t-50)*sin(0.5*sin(3.5*t)+1.2*sin(0.5*t)-0.7*sin(s*t)) \quad (6.1)$$

Tato umělá data budou použita na testování metod PCA a autoenkoderu. Cílem zde je zbavit je šumu (nebo jej alespoň potlačit na uspokojivou úroveň), aby přefiltrovaný signál svým průběhem co nejvíce odpovídal signálu původnímu.



Obr. 8: Uměle vytvořená data (graf nahoře) a jejich zašumnění pomocí funkce random (graf dole)

6.2 Reálná data

6.2.1 Kompozitní struktury

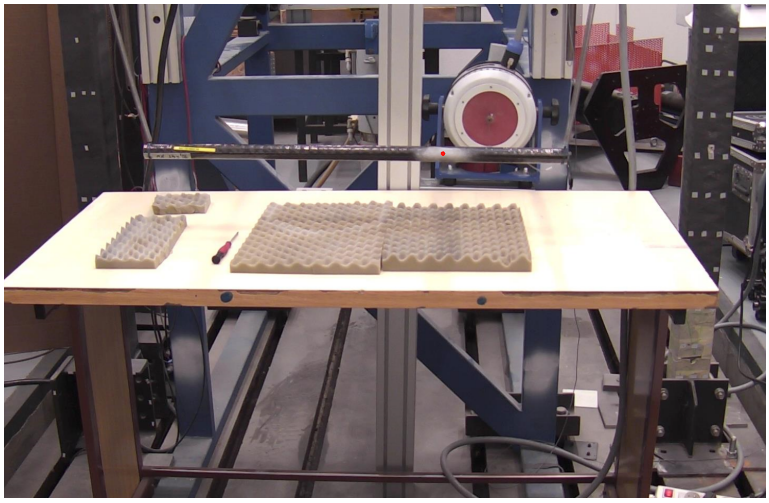
Pro testování shlukovacích algoritmů byla poskytnuta data společnosti CompoTech PLUS, spol. s.r.o., kde se experimentálně měřily vibrace typem a strukturou odlišných kompozitových trubek. Vibrace byly měřeny bezkontaktním laserovým vibrometrem Polytec PSV 400 D4063, mechanické buzení bylo vytvářeno budičem TIRA S51144-M se zesilovačem TIRA BAA 1000, kde jeho skutečná síla byla měřena snímačem Brüel & Kjaer 8230.

Data byla poskytnuta ze 3 druhů kompozitních struktur, kde každý druh má své specifické vnitřní uspořádání, shrnuje Tab. 1.

Tab. 1: Přehled typů kompozitových struktur

Druh	**Vnitřní struktura 1**	**Vnitřní struktura 2**	**Vnitřní struktura 3**	**Vnitřní struktura 4**
T700	N1	N2	P	T
XN80	N1	N2	P	T
XN90	N1	N2	P	T

Každé měření obsahuje 25600 hodnot, kdy budící frekvence byla v rozsahu $f \in \langle 0.125, 3200 \rangle \, [Hz]$.

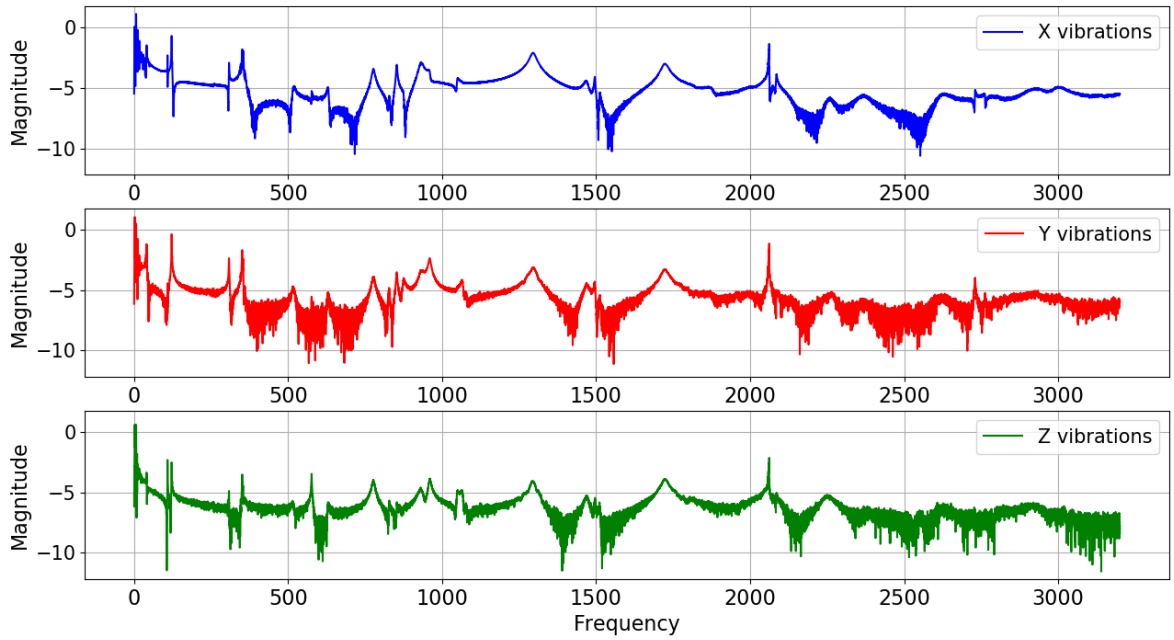


Obr. 9: Ukázka měřené kompozitní struktury ve tvaru trubky

K naměřeným datům se přistupuje přes program Polytec Scan Viewer. V tomto programu lze nastavit, jaká data budou zobrazena a je zde možnost jejich uložení. Program disponuje možnostmi ovládání pořízeného obrázku a animace průběhu experimentu. Je zde zobrazen snímek měřené kompozitní struktury s vyznačenou pozicí měřeného bodu [12].

6.3 Předzpracování dat

Na následujícím obrázku jsou zobrazena data převedena z binárního souboru firmy Polytec do .csv formátu, který je dále zpracován v jazyce Python. Pro lepší přehled byla data převedena do logaritmických souřadnic.

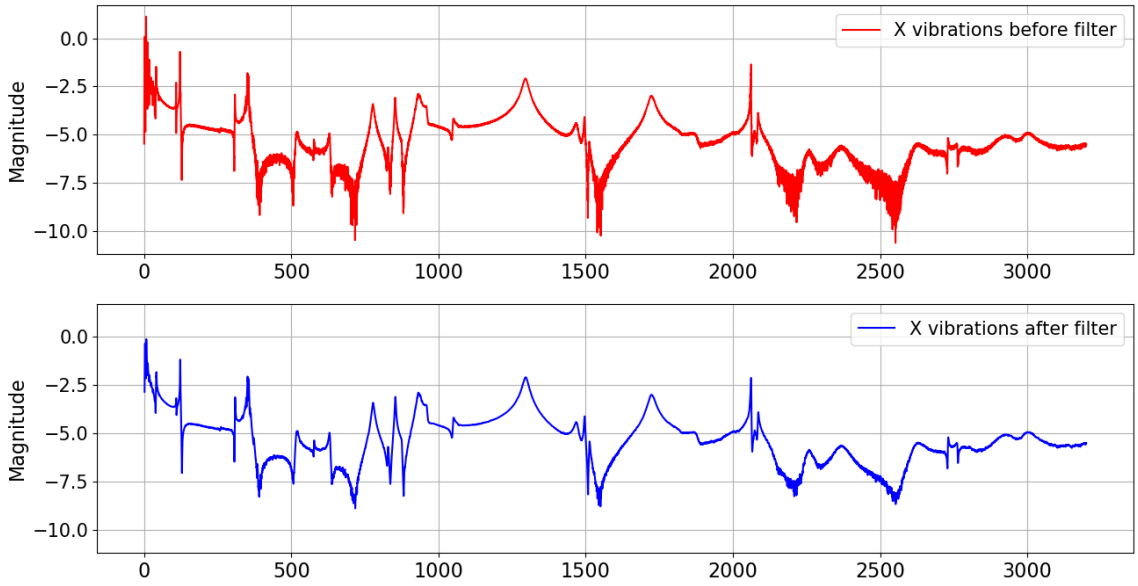


Obr. 10: Zobrazené průběhy vibrací kompozitní struktury T700-P na osách X, Y, Z. Data jsou převedena do logaritmických souřadnic

Jak je z Obr. 10 patrné, data jsou poměrně zašuměná. Pro další zpracování byl tento šum odstraněn metodou plovoucího průměru:

$$y(k) = (0.2 * y(k-5) + 0.3y(k-4) + 0.4 * y(k-3) + 0.5y(k-2) +$$
$$+0.7 * y(k-1) + y()k + 0.7 * y(k+1) + 0.5 * y(k+2) + 0.4 * y(k+3) +$$
$$+0.3 * y(k+4) + 0.2 * y(k+5))/5.2, \tag{6.2}$$

výsledek odfiltrování šumu je znázorněn na Obr. 11.



Obr. 11: Data před a po aplikací metody plovoucího průměru vibrací na ose x kompozitní struktury T700-P

Jelikož byla data poměrně rozdílných rozsahů ($10^{1} - 10^{-8}$), byla na ně dále aplikována metoda Z-Score (4.6), tím byla data unifikována.

7 Testování algoritmů na datech

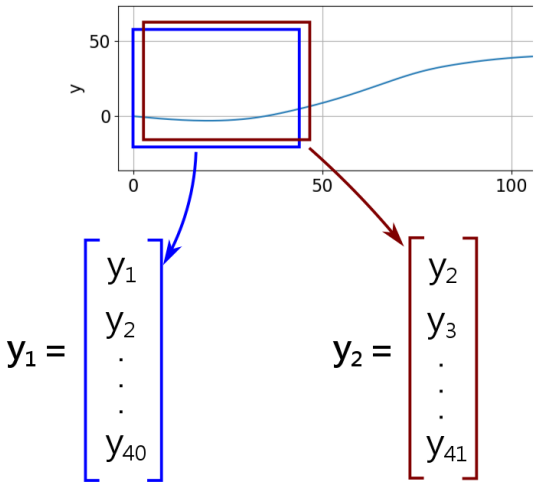
7.1 Odstranění šumu pomocí metody PCA

Metoda PCA byla testována na umělých datech, Obr. 8. Úkolem zde bylo potlačit šum v datech s co možná nejlepším zachováním původního signálu vhodným zvolením počtu hlavních komponent.

Data byla nejdříve rozdělena do vektorů $v_1, v_2, ..., v_{2500}$ se složkami

$$v_1 = [y_1, y_2, ..., y_{40}]^T, \tag{7.1}$$

za použití klouzavého okna s posunutím vždy o jeden vzorek, Obr. 12.



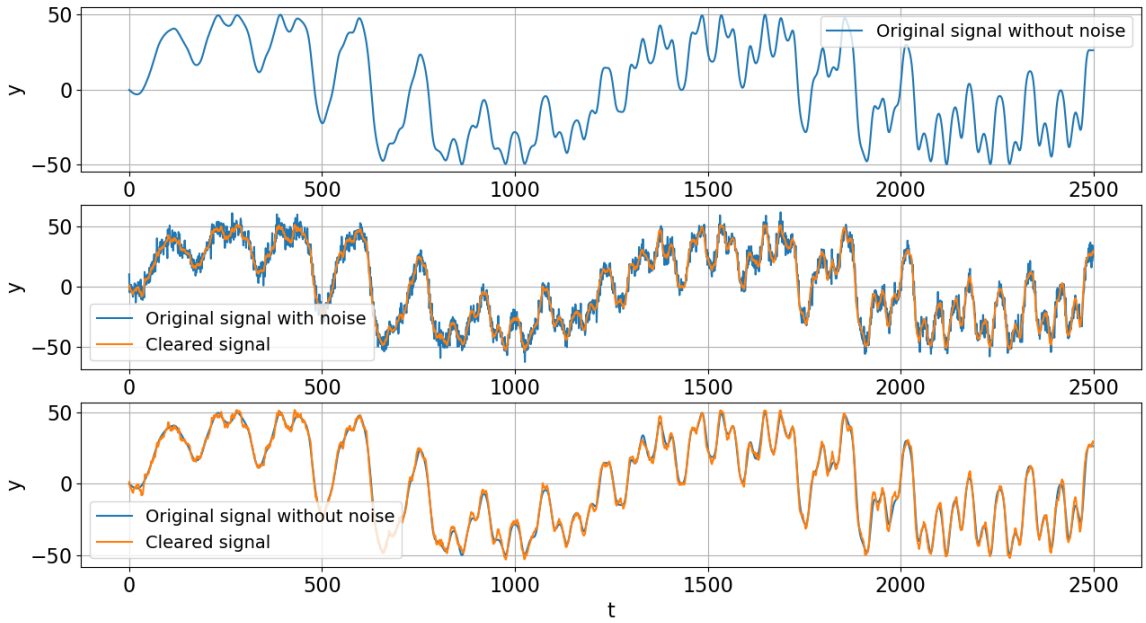
Obr. 12: Ukázka výběru vektorů z umělých dat pro metodu PCA. Okno pro výběr hodnot je široké 40 vzorků, každý další vektor je tvořen posunutím okna o jeden vzorek

Z hodnot vektorů $v_1, v_2, ..., v_{2500}$ byla dále vytvořena matice

$$A = \begin{bmatrix} y_1 & \cdots & y_{2460} \\ \vdots & \ddots & \vdots \\ y_{40} & \cdots & y_{2500} \end{bmatrix}, \tag{7.2}$$

tak, že každý sloupec je tvořen jedním vektorem. Dále byla matice A zpracována metodou PCA, kde byla vytvořena její kovarianční matice. Následně se z kovarianční matice určila hlavní čísla a jejich příslušné hlavní vektory, které jsou dále nazvané hlavní komponenty systému. Pro určení, kolik hlavních komponent je zde nutné použít slouží absolutní velikost hlavních čísel daných komponent, respektive čím vyšší absolutní hodnota vlastního čísla, tím větší důležitost hlavní komponenty z pohledu velikosti přenášené informace. Testováním algoritmu na daných datech bylo zjištěno, že v tomto případě majoritní část informace systému nesou právě první 4 hlavní komponenty.

Tyto 4 hlavní komponenty byly následně použity pro zpětné převedení dat. Jak je vidět na Obr. 13, metoda PCA dokázala potlačit šum na poměrně dobrou úroveň, kdy, až drobné výchylky, odpovídá zpracovaný signál velmi dobře svým průběhem signálu původnímu.



Obr. 13: Výsledky metody PCA, v horním grafu je znázorněn průběh původního, nezašuměného signálu, v prostředním pak porovnání zašuměného signálu se zpracovaným signálem metodou PCA, v dolním grafu srovnání původního signálu se signálem, kde byl šum potlačen.

7.2 Odstranění šumu pomocí Autoenkoderu

Stejně jako u metody PCA i zde byl použit Autoenkodér pro potlačení uměle vytvořeného šumu dat znázorněných na Obr. 8.

Princip Autoenkodéru zde funguje tak, že na vstupu neuronové sítě jsou předložena zašuměná data a na jejím výstupu pak nezašuměná. Autoenkodér je porovnáváním vstupu a výstupu naučen, jak signál zpracovávat. Vloží-li se pak na vstup zašuměný signál, je naučeným Autoenkodérem převeden na signál s šumem značně redukovaným.

Data signálu zde byla pomocí klouzavého okna (s šířkou 40ti vzorků) připravena stejně jako u metody PCA. Autoenkodér se tedy učil porovnáváním jednotlivých vektorů zašuměného a nezašuměného signálu. Algoritmu byla nastavena míra učení na 0.2 s 20000 kroky. Počet perceptorů na kódové a dekódové vrstvě byl zvolen 12.

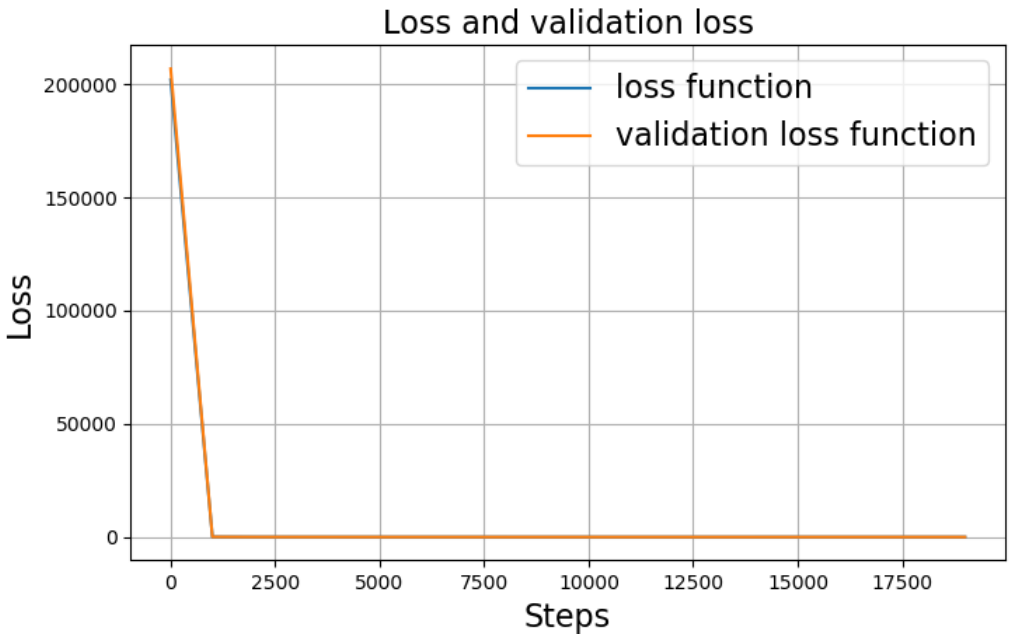
Tab. 2: Hodnoty ztrát při učení a validace modelu Autoenkodéru

Krok	Ztráta při učení	Validační ztráta
1	202033.967	206869.203
1000	55.29	72.62
2000	16.785	29.213
3000	11.981	20.679
4000	9.276	14.457
5000	7.623	11.198
6000	6.637	9.442
7000	5.203	8.274
8000	5.203	7.389
9000	4.89	6.93
10000	4.721	6.581
11000	4.538	6.155
12000	4.967	6.44
13000	4.459	5.953
14000	4.452	5.97
15000	5.682	7.55
16000	4.581	5.831
17000	4.484	5.827
18000	4.463	6.004
19000	4.442	5.913

Průběh učení Autoenkodéru je zaznamenán na Obr. 14, kde je k nahlédnutí průměrný kvadrát chyby - rozdílu mezi modelem a skutečným signálem podle

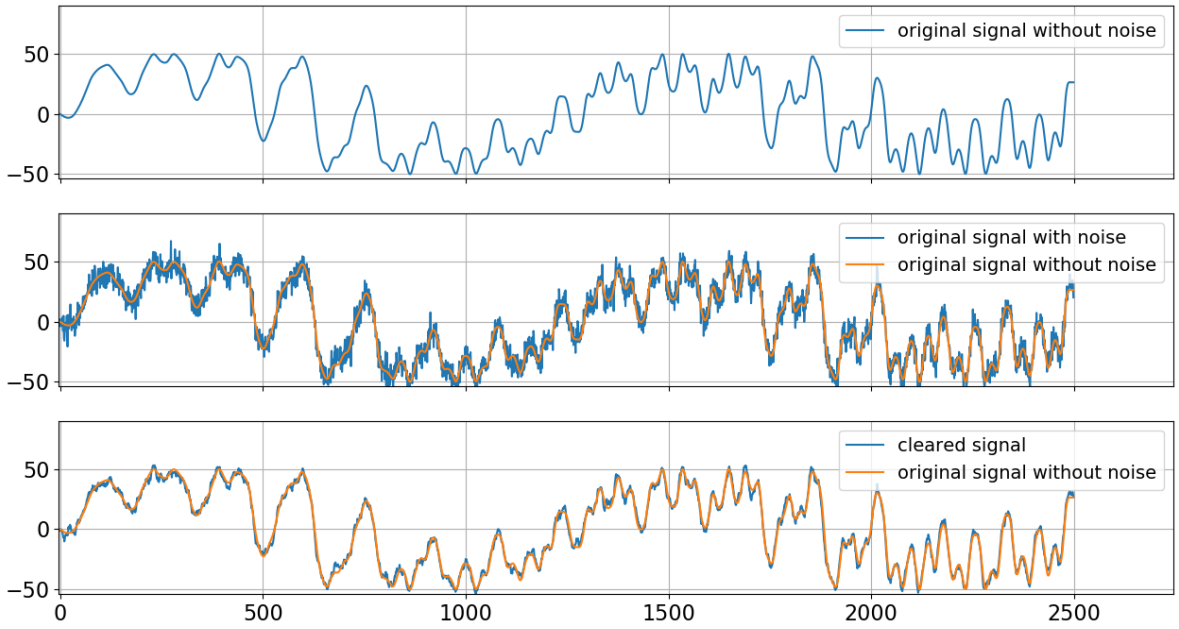
$$J = \frac{1}{M} \sum (y_m - y)^2, \tag{7.3}$$

kde y_m je hodnota modelu v daném bodě a y je hodnota skutečného signálu v daném bodě. Ztráta při učení byla počítána na trénovacích datech, validační ztráta pak na datech, která byla modelu předložena po jeho naučení. Jak lze nahlédnout z Obr. 14, model se velmi dobře naučil v prvních 1000 iteracích a v průběhu snižoval průměrnou chybu na uspokojivou úroveň. Stejně tak obstál i při testování na validačních datech, která nebyla použita při jeho učení.



Obr. 14: Průběh ztrát při učení a ztráty při validaci modelu autoenkodéru

Výsledky použití Autoenkodéru pro potlačení šumu signálu jsou zobrazeny na Obr. 15, kde se porovnává zašuměný signál se signálem původním a ve spodní části pak původní signál se signálem, který byl zpracovaný Autoenkodérem. Jak je k nahlédnutí, váhy neuronů uvnitř kódovací a dekodovací vrstvy byly vypočteny uspokojivě a šum byl na drobné odchylky potlačen na velmi dobrou úroveň.

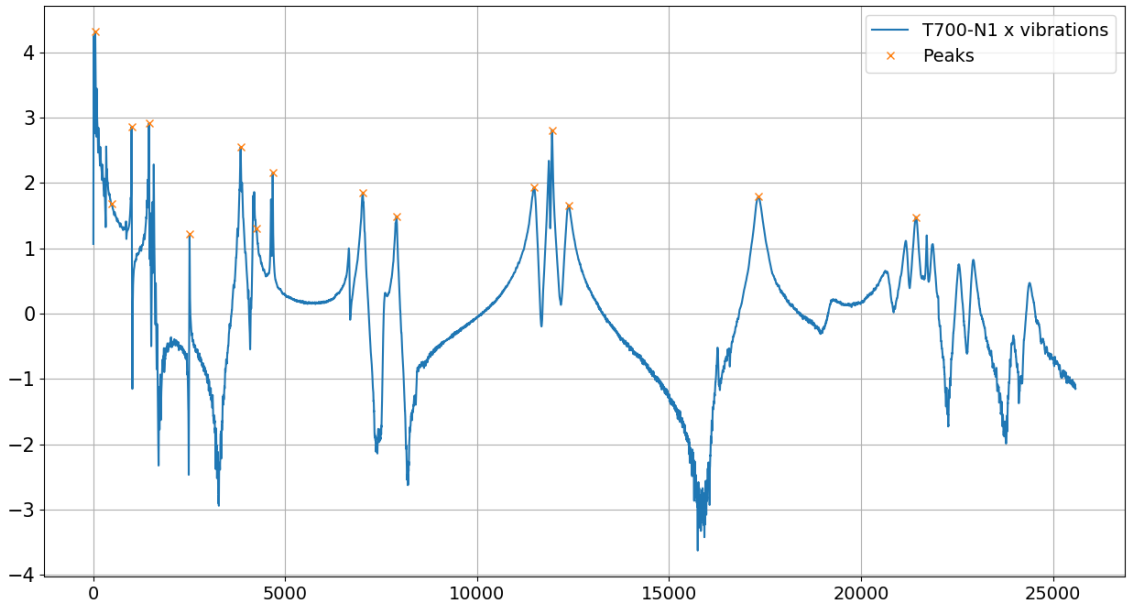


Obr. 15: Průběhy při výpočtu pomocí autoenkodéru. Graf nahoře obsahuje originální, nezašuměný signál. Graf uprostřed pak porovnání nezašuměného signálu se zašuměným. Spodní graf znázorňuje porovnání původního signálu se signálem, který byl zpracován autoenkodérem a u nějž byl značně redukován šum.

7.3 Klasifikace dat z měření vibrací pomocí SOM

Jak již bylo popsáno v rešeršní části, algoritmus SOM slouží pro vytváření shluků daného datasetu. V tomto případě byla metoda aplikována na reálná data z měření vibrací kompozitních struktur.

K problému bylo přistoupeno tak, že z průběhů vibrací na osách x, y, z byly vybrány význačné body (v tomto případě peaky průběhu vibrací, Obr. 16), pomocí knihovny *scipy.signal.find_peaks* a jim příslušné odpovídající frekvence. Takto byly vybrány hodnoty pro všech 12 různých kompozitních struktur a vzaty jako vstupní data pro algoritmus SOM, který měl po svém naučení vytvořit 12 datových shluků. V případě správného naučení by algoritmus pak měl správně určit, o který typ konkrétní struktury se jedná, vloží-li se mu na vstup data konkrétní struktury.

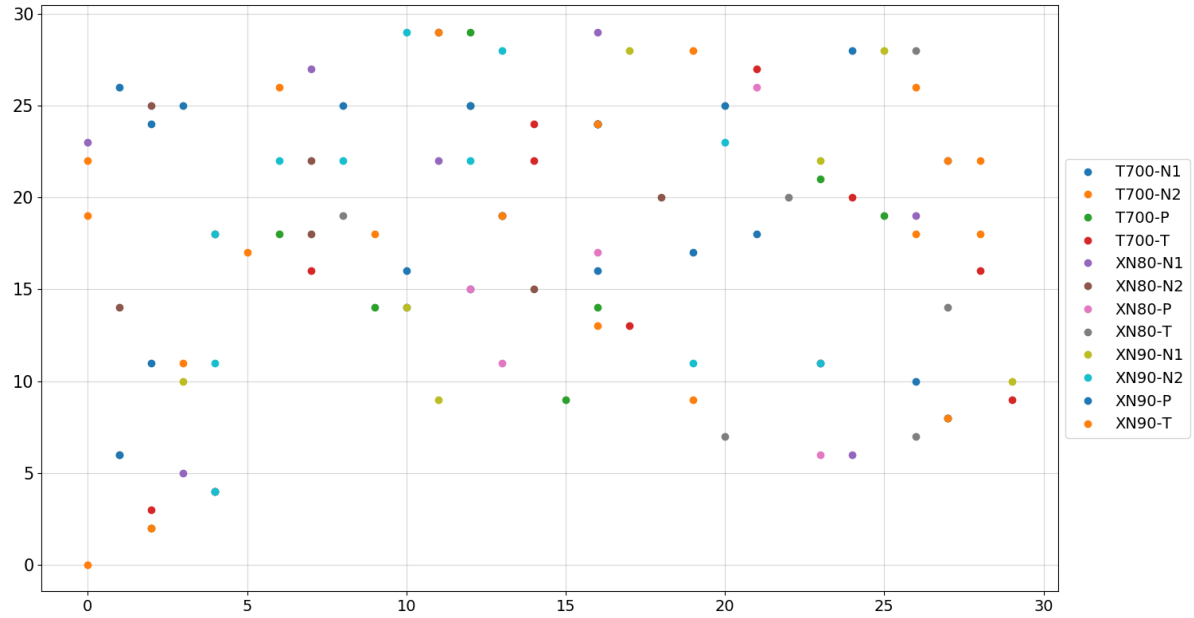


Obr. 16: Průběh vibrací na ose x měřené kompozitní struktury T700-N1 s vyznačenými peaky, které jsou vzaty jako vstup pro algoritmus SOM.

Jak se ukázalo, algoritmus SOM nebyl schopen vybraná data rozklasifikovat do shluků. Bylo vyzkoušeno několik konfigurací Samo-organizační mapy:

- 10×10 neuronů v síti
- 20×20 neuronů v síti
- 30×30 neuronů v síti
- 50×50 neuronů v síti

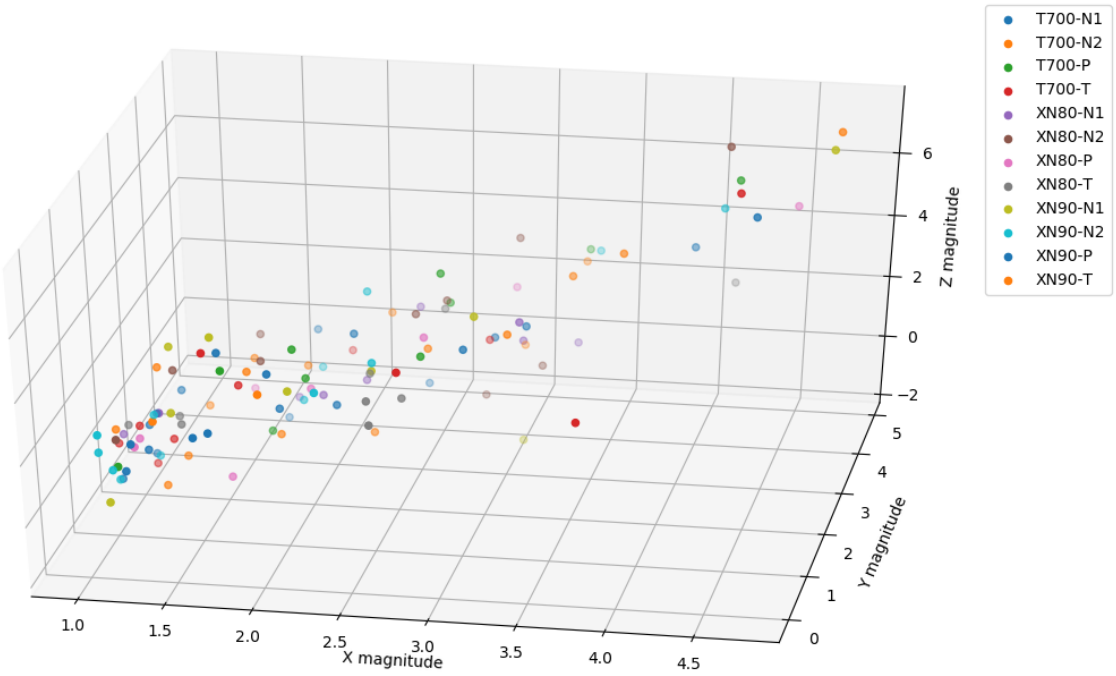
ani v jednom z výše uvedených případů však nedošlo k smysluplnému rozdělení dat. Na Obr. 17 je uveden příklad výsledku naučení neuronové sítě, kde byla o rozměrech 30×30 neuronů učená na vstupní data z průběhu měření vibrací.



Obr. 17: Naučená Samo-organizační mapa s 30 x 30 neurony v síti. Jako vstupní data pro naučení byly použity hodnoty peaků vibrací kompozitních struktur na osách x, y, z

V ideálním případě by měla neuronová sít přeskupit své neurony tak, aby ty aktivované vytvořili shluky (tzn. barevně označené body na Obr. 17). Každé barevné uskupení neuronů zde odpovídá vítězným jednotkám, na základě jejich aktivace v procesu učení při specifickém vstupu jedné dané kompozitní struktury. Zde například neuronová síť aktivuje neurony $a_{4,4}, a_{20,23}, a_{10,24}, a_{23,11}$ pro kompozitní strukturu XN90-N2, které jsou ovšem umístěny vzdálenostně daleko od sebe a neutvářejí tak shluk. Dodají-li se tedy takto naučené Samo-organizační mapě vstupní data od jednotlivých kompozitních struktur, není schopná určit, do jakého klastru daná data náleží.

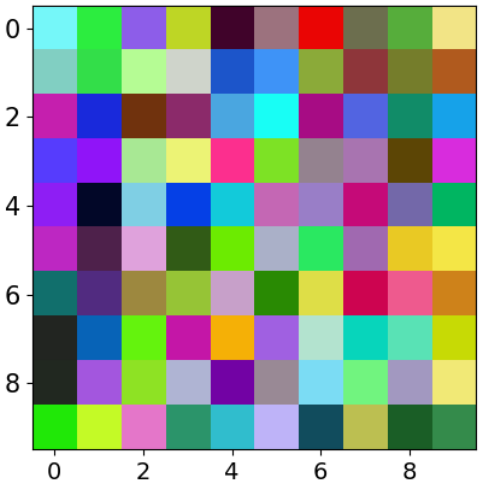
Pro kontrolu byla data peaku vibrací vykreslena v 3-D grafu, Obr. 18. Lze tak nahlédnout, že body nevytvářejí už od počátku žádné struktury a tak není možné naučit Samo-organizační mapu tak, aby správně vytvořila shluky, ze kterých by se následně dalo určit, která data kompozitní struktury jsou určena ke kterému náleží.



Obr. 18: Ukázka 3-D vizaulizace datových bodů peaků průběhů vibrací na osách x, y, z. Jak je k nahlédnutí, data netvoří žádné struktury před klasifikací a proto nebylo možné je rozdělit pomocí algoritmu SOM

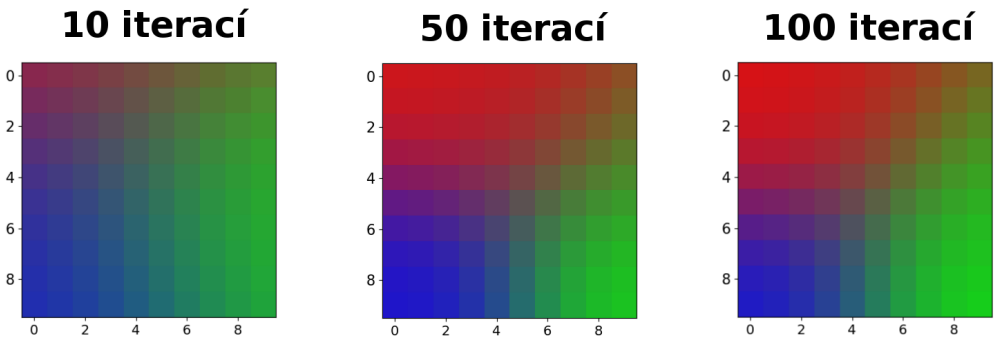
Je pravděpodobné, že nebyl zvolen správný způsob výběru vstupních dat pro klasifikační neuronovou síť a že lze nalézt mezi různými měřeními vibrací kompozitních struktur závislosti, to ovšem přesahuje rámec této práce. Pro funkční demonstraci Samo-organizační mapy byla vybrána jednodušší umělá data.

Mějme vstupní vektory $v_1 = [1, 0, 0]^T$, $v_2 = [0, 1, 0]^T$ a $v_3 = [0, 0, 1]^T$, které reprezentují barvy v RGB a ke kterým se uměle přidá šum. Na Obr. 19 je ukázána neuronová síť před začátkem výpočtu vah jednotlivých neuronových jednotek.



Obr. 19: Samo-organizační mapa před začátkem výpočtu s 10 x 10 neurony v síti. Před výpočtem jsou váhy jednotlivých neuronů rozděleny náhodně - znázorněno různými barvami

Dále nastává proces učení neuronové sítě, kdy jsou vektory v_1, v_2, v_3 zatěžovány šumem a stávají se pak vstupem do Samo-organizační mapy, která na jejich základě určuje vítězné neurony a vytváří tak shluky.



Obr. 20: Výsledek a proces učení se neuronové Samo-organizační mapy. Je zde vidět, jak jsou postupně tvořeny 3 datové shluky při učení SOM vkládáním modifikovaných vektorů v_1, v_2, v_3

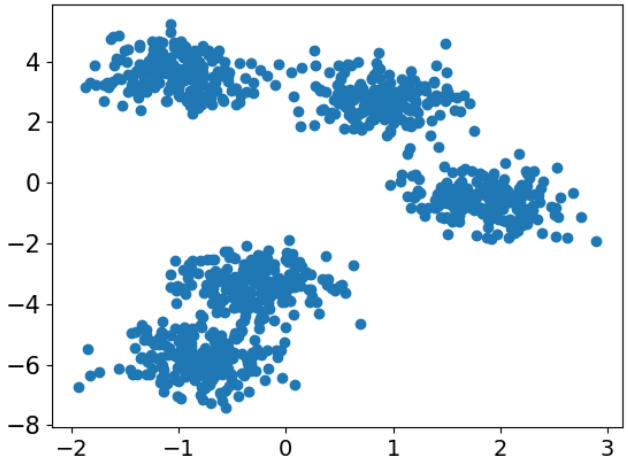
Na Obr. 20 je vidět výsledek výpočtu pomocí Samo-organizační mapy. Algoritmus dokázal správně vytvořit příslušný počet shluků a naučit se tak přiřazovat vstupní data.

Předloží-li se tedy nyní k neuronové síti vektor $v_4 = [0, 1.01, 0]$, bude správně přiřazen do shluku, který je zde reprezentován zelenou barvou.

Jelikož bylo zjištěno, že extrahovaná data z měření vibrací kompozitních struktur nejsou pro klasifikační metody zkoumané v této práci použitelná, jsou pro následující metody vytvořena data umělá, aby na nich bylo možné demonstrovat vybrané typy analýz.

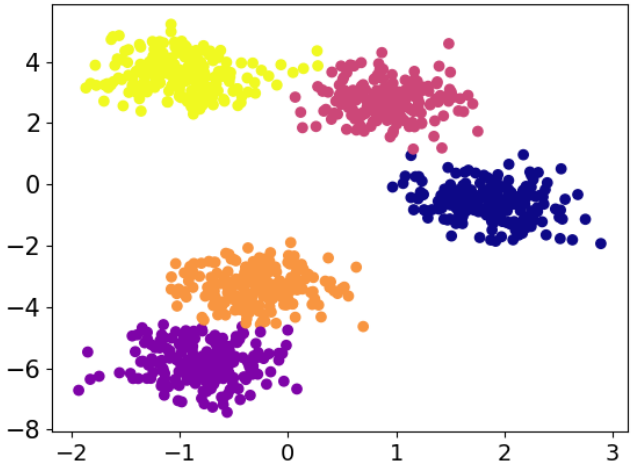
7.4 Klasifikace dat pomocí metody K-Means

Metoda K-Means je nejrozšířenější metodou, co se týče vytváření shluku dat. Její princip je vysvětlen v teoretické části této práce. Algoritmus zde byl testován na uměle vytvořených datech. Tato umělá data byla vytvořena pomocí knihovny *sklearn* a jsou zobrazena na Obr. 21, počet vzorků dat je 1000. Algoritmus K-Means se zde bude testovat, zda bude schopen správně označit z daných dat 5 shluků.



Obr. 21: Uměle vytvořená data pro klasifikaci metodou K-Means. Data jsou předpřipravena tak, aby výsledkem výpočtu bylo vytvoření 5ti datových shluků

Dle předpokladu bylo v daných datech algoritmem správně označeno 5 shluků. Metoda zde náhodně rozmístila zadaný počet ($n = 5$) centroidů, změřila Euklidovské vzdálenosti mezi jednotlivými body a centroidy. K centroidu byli přiřazeny ty body, které k němu měli ze všech 5ti nejkratší vzdálenost. Následně se přepočítala poloha centroidu při výpočtu těžiště nového tělesa a proces se opakoval, dokud nezbyli žádné body, které by byli k přiřazení. Výsledek je k nahlédnutí na Obr. 22.



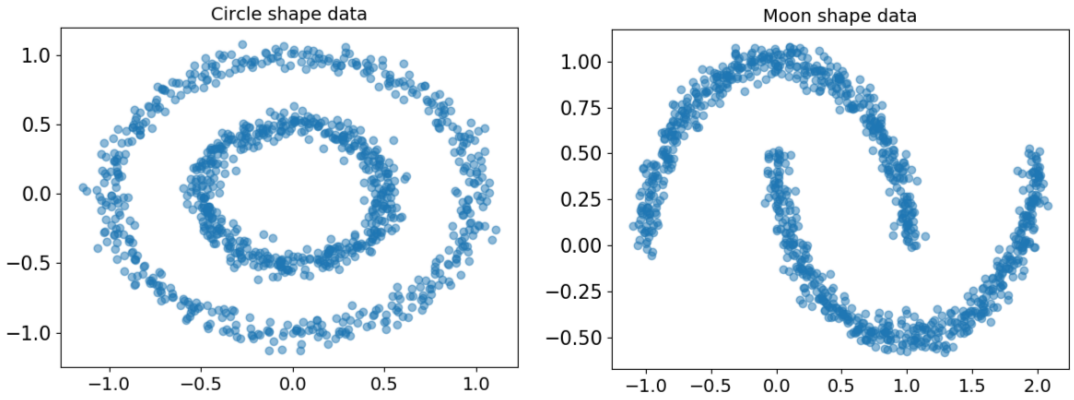
Obr. 22: Zpracovaná data metodou K-Means. Algoritmus správně určil příslušnost datových bodů do klastru, jak bylo zadáno.

7.5 Klasifikace dat pomocí metody DBSCAN

Princip metody DBSCAN, jak bylo vysvětleno v teoretické části, je založen na vyhledávání sousedních bodů a jejich vyhodnocování pomocí dvou parametrů:

- ϵ-okolí daného datového bodu (kde se pozoruje, zda má takový bod ve svém kruhovém okolí sousední body)
- parametr $minPts$, který udává počet sousedů, který musí mít daný bod ve svém okolí, aby mohl být prohlášen jako bod jádra a součástí shluku

Pro testování této metody byly vytvořeny dvě sady umělých dat, na kterých je demonstrováno její efektivita. Každá sada obsahuje 1000 datových bodů a je zatížena drobným šumem. Umělá data byla vytvořena pomocí knihovny *sklearn*.

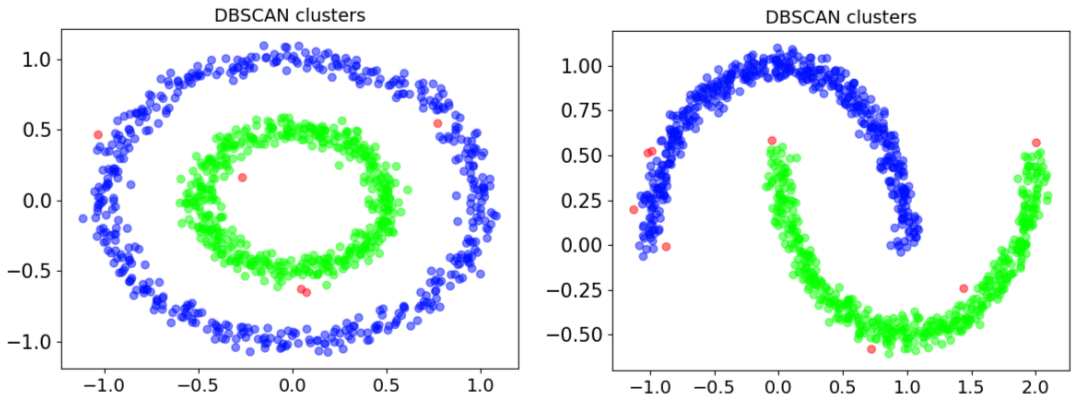


Obr. 23: Umělá data vytvořena pro testování metody DBSCAN. Vlevo data ve tvaru kruhů (zde bude snaha je rozdělit na 2 shluky tak, aby tvořili dvě nezávislé kružnice), vpravo ve tvaru půlměsíců (opět budou vytvořeny dva shluky)

Metoda DBSCAN se těší popularitě právě pro to, že je schopna úspěšně rozdělit data dle topologie jejich rozmístění v souřadném systému. Jinými slovy, tam kde by K-Means selhalo, protože by počítalo s průměry dat a tvořilo shluky proti intuici, analýza DBSCAN uspěje.

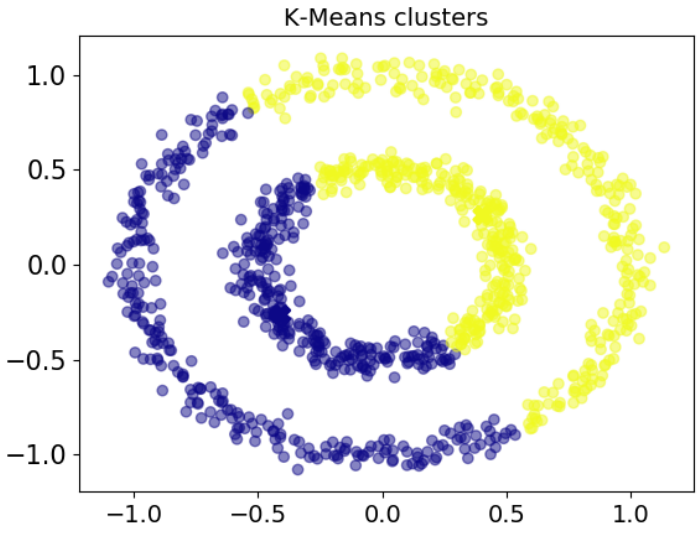
Parametry algoritmu byly nastaveny následovně:

- V prvním případě na Obr. 23 kruhových dat bylo nastaveno okolí $\epsilon = 0.15$ a nutný počet sousedních bodů $minPts = 2$
- V případě druhém u dat ve tvaru půlměsíců pak okolí $\epsilon = 0.1$ a nutný počet sousedních bodů $minPts = 2$



Obr. 24: Výsledek metody DBSCAN. Algoritmus dokázal rozdělit daná data do odpovídajících shluků. Body jádra a hranic jsou označeny modrou respektive zelenou barvou. Body, které algoritmus správně vyhodnotil jako šum jsou označeny červeně.

Výsledky jsou zobrazeny na Obr. 24, kde byla data rozdělena tak, jak bylo zamýšleno. Je ovšem potřeba správně volit parametry ϵ a $minPts$ pro správné vyhodnocení. To bývá na posouzení uživatele. Pro porovnání je zde na Obr. 25 uvedeno rozdělení dat pomocí metody K-Means.



Obr. 25: Příklad nesprávného klastrování zvolením nevhodné metody. Jako algoritmus byl zvolen K-Means.

8 Závěr

V této bakalářské práci byla provedena rešerše metod samoučících algoritmů strojového učení pro analýzu průmyslových dat. Dále zde byl popsán koncept Big data, krátce představen pojem Průmysl 4.0 a popsána norma ISO 13374, která cíli na sjednocení monitorovacích systému mezi podniky a institucemi. Následně byly vybrány metody sloužící pro redukci dimenzí vysoko rozměrných dat a algorimty určené pro klasifikaci. Každá z metod zde byla podrobně představena, obecně vysvětleno její fungování a krátce popsán její matematický princip. Dále byli tyto algoritmy testovány na umělých a reálných datech.

Pro testování algoritmů byla vytvořena umělá data, která byla pro metody PCA a Autoenkodér stejná. Zde se pozorovalo, jak dobře dokáží jednotlivé metody odstranit z dat vytvořený šum se zachováním původního signálu.

Pro testování shlukovacích metod K-Means, DBSCAN a SOM byla původně zamýšleno použití dat z vibrometru, která byl nejdříve předzpracována filtrováním za pomocí metody plovoucího průměru a následně upravena metodou Z-Score pro dosažení jednotné škály. Poté byly z dat vybrány význačné body (peaky), které měly sloužit jako identifikátory pro určení daných konkrétních kompozitních struktur. Bohužel se ukázalo hned na první testované metodě Samo-organizačních map, že takto předzpracovaná data jsou nevhodná pro shlukovací algoritmy. Toto bylo potvrzeno vizualizací daných dat při posouzení, že data už od počátku nevykazují žádné základní struktury vhodné pro klastrovací metody. Následně byla tedy ke každé shlukovací metodě vytvořena umělá data, na kterých se pak testovaly.

Metoda Samo-organizačních map se naučila na několika drobně modifikovaných vektorech, které reprezentovaly základní barvy RGB, rozklastrovávat hodnoty do 3 tříd jak bylo předpokládáno a byla tak verifikována její funkčnost oproti neschopnosti vytvořit shluky z dat reálného měření vibrací kompozitních struktur.

Pro další shlukovací metody byla vytvořena umělá data, aby bylo možné demonstrovat jejich použitelnost. Pro metodu DBSCAN byly naprogramovány dvě datové sady specifických tvarů, na kterých se následně analýza velmi dobře se správným přiřazením bodů do shluků a správným označením bodů, které reprezentovali šum. Pro porovnání metod byla na jednu sadu dat vyzkoušena metoda K-Means, která v tomto případě selhala, neboť není schopna rozlišovat shluky dat dle jejich topologického uspořádání.

Použitelnost metody K-Means byla ukázána rovněž na umělých datech, ale tentokrát vhodných pro její použití. Data byla uskupena do požadovaných shluků.

Zdroje

[1] *What is Machine Learning? A definition - Expert System* [online]. 7. březen 2017 [vid. 2019-08-14]. Dostupné z: https://www.expertsystem.com/machine-learning-definition/

[2] DWIVEDI, Divyansh. Machine Learning For Beginners. *Medium* [online]. 27. březen 2019 [vid. 2019-08-14]. Dostupné z: https://towardsdatascience.com/machine-learning-for-beginners-d247a9420dab

[3] PALADE, Vasile, Cosmin Danut BOCANIALA a L. C. JAIN, ed. *Computational intelligence in fault diagnosis*. London: Springer, 2006. Advanced information and knowledge processing. ISBN 978-1-84628-343-7.

[4] CHEN, James. Neural Network Definition. *Investopedia* [online]. [vid. 2019-08-14]. Dostupné z: https://www.investopedia.com/terms/n/neuralnetwork.asp

[5] *Everything you need to know about Neural Networks* [online]. [vid. 2019-08-14]. Dostupné z: https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491

[6] *Co se skrývá pod výrazy Industry 4.0 / Průmysl 4.0 ? | Automatizace.HW.cz* [online]. [vid. 2019-07-30]. Dostupné z: https://automatizace.hw.cz/mimochodem/co-je-se-skryva-pod-vyrazy-industry-40-prumysl-40.html

[7] *Big data for dummies.pdf* [online]. [vid. 2019-07-29]. Dostupné z: https://eecs.wsu.edu/~yinghui/mat/courses/fall%202015/resources/Big%20data%20for%20dummies.pdf

[8] CHEN, Min, Simone A. LUDWIG a Keqin LI. Clustering in Big Data. In: Kuan-Ching LI, Hai JIANG a Albert Y. ZOMAYA, ed. *Big Data Management and Processing* [online]. 1. vyd. B.m.: Chapman and Hall/CRC, 2017 [vid. 2019-07-30], s. 333–346. ISBN 978-1-315-15400-8. Dostupné z: doi:10.1201/9781315154008-16

[9] *Big data: The next frontier for innovation, competition, and productivity | McKinsey* [online]. [vid. 2019-07-30]. Dostupné z: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation

[10] *Top 13 Best Big Data Companies of 2019* [online]. [vid. 2019-08-14]. Dostupné z: https://www.softwaretestinghelp.com/big-data-companies/

[11] *Prediktivní údržba a metody technické prognostiky – seznámení se s problematikou | BOZPinfo.cz* [online]. [vid. 2019-08-11]. Dostupné z: https://www.bozpinfo.cz/josra/prediktivni-udrzba-metody-technicke-prognostiky-seznameni-se-s-problematikou

[12] *F2-BP-2019-Budik-Ondrej-IAT-BP-2019-Ondrej-Budik-Bakalarska-prace.pdf* [online]. [vid. 2019-07-31]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/83624/F2-

BP-2019-Budik-Ondrej-IAT-BP-2019-Ondrej-Budik-Bakalarska-prace.pdf?sequence=-1&isAllowed=y

[13] DING, Chris, Xiaofeng HE, Hongyuan ZHA a Horst SIMON. Adaptive dimension reduction for clustering high dimensional data. nedatováno, 9.

[14] Clustering in Machine Learning. *GeeksforGeeks* [online]. 15. leden 2018 [vid. 2019-07-31]. Dostupné z: https://www.geeksforgeeks.org/clustering-in-machine-learning/

[15] RAGHAVAN. Principal component analysis (PCA): Explained and implemented. *Medium* [online]. 19. srpen 2018 [vid. 2019-08-14]. Dostupné z: https://medium.com/@raghavan99o/principal-component-analysis-pca-explained-and-implemented-eeab7cb73b72

[16] JAADI, Zakaria. A step by step explanation of Principal Component Analysis. *Medium* [online]. 24. květen 2019 [vid. 2019-07-22]. Dostupné z: https://towardsdatascience.com/a-step-by-step-explanation-of-principal-component-analysis-b836fb9c97e2

[17] *Dimension Reduction Techniques (PCA vs LDA) in Machine Learning – Part 2* [online]. 9. únor 2018 [vid. 2019-08-14]. Dostupné z: http://www.vfirst.com/blog/techfirst/dimension-reduction-techniques-pca-vs-lda-in-machine-learning-part-2/

[18] JIGSAW2212. Comparing PCA and LDA techniques. *a tryst with programming* [online]. 28. říjen 2015 [vid. 2019-08-14]. Dostupné z: https://atrystwithprogramming.wordpress.com/2015/10/28/comparing-pca-and-lda-techniques/

[19] Linear Discriminant Analysis. *Dr. Sebastian Raschka* [online]. 3. srpen 2014 [vid. 2019-08-14]. Dostupné z: https://sebastianraschka.com/Articles/2014_python_lda.html

[20] Introduction to t-SNE. *DataCamp Community* [online]. 13. září 2018 [vid. 2019-07-23]. Dostupné z: https://www.datacamp.com/community/tutorials/introduction-t-sne

[21] PATEL, Ankur A. *Hands-On Unsupervised Learning Using Python: How to Build Applied Machine Learning Solutions from Unlabeled Data*. B.m.: O'Reilly Media, Inc., 2019. ISBN 978-1-4920-3561-9.

[22] MCCULLAGH, P. a John A. NELDER. *Generalized linear models*. 2nd ed. Boca Raton: Chapman & Hall/CRC, 1998. Monographs on statistics and applied probability, 37. ISBN 978-0-412-31760-6.

[23] VAN DER MATTEN, Laurens a Geoffrey HINTON. *Visualising Data using t-SNE* [online]. Dostupné z: http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

[24] KEITAKURITA, Author. Paper Dissected: "Visualizing Data using t-SNE" Explained. *Machine Learning Explained* [online]. 14. září 2018 [vid. 2019-07-23]. Dostupné z: https://mlexplained.com/2018/09/14/paper-dissected-visualizing-data-using-t-sne-explained/

[25] MATT.0. 10 Tips for Choosing the Optimal Number of Clusters. *Medium* [online]. 28. leden 2019 [vid. 2019-07-24]. Dostupné z: https://towardsdatascience.com/10-tips-for-choosing-the-optimal-number-of-clusters-277e93d72d92

[26] VATTANI, Andrea. k-means Requires Exponentially Many Iterations Even in the Plane. *Discrete & Computational Geometry* [online]. 2011, **45**(4), 596–616. ISSN 0179-5376, 1432-0444. Dostupné z: doi:10.1007/s00454-011-9340-1

[27] TREVINO, Andrea. *Introduction to K-means Clustering* [online]. [vid. 2019-07-24]. Dostupné z: https://www.datascience.com/blog/k-means-clustering

[28] CHAUDHARY, Vikas, R. S. BHATIA a Anil K. AHLAWAT. A novel Self-Organizing Map (SOM) learning algorithm with nearest and farthest neurons. *Alexandria Engineering Journal* [online]. 2014, **53**(4), 827–831. ISSN 1110-0168. Dostupné z: doi:10.1016/j.aej.2014.09.007

[29] RITTER, H. a T. KOHONEN. Self-organizing semantic maps. *Biological Cybernetics* [online]. 1989, **61**(4), 241–254. ISSN 0340-1200, 1432-0770. Dostupné z: doi:10.1007/BF00203171

[30] ŷhat | Self-Organising Maps: In Depth. *ŷhat | Blog* [online]. [vid. 2019-07-22]. Dostupné z: http://blog.yhat.com/posts/self-organizing-maps-2.html

[31] *COMPUTATIONAL INTELLIGENCE international joint conference, ijcci 2017.* S.l.: SPRINGER, 2019. ISBN 978-3-030-16469-0.

[32] *Kohonen Network - Background Information* [online]. [vid. 2019-07-22]. Dostupné z: http://www.lohninger.com/helpcsuite/kohonen_network_-_background_information.htm

[33] HUBENS, Nathan. Deep inside: Autoencoders. *Medium* [online]. 10. duben 2018 [vid. 2019-07-27]. Dostupné z: https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

[34] BADR, Will. Auto-Encoder: What Is It? And What Is It Used For? (Part 1). *Medium* [online]. 1. červenec 2019 [vid. 2019-07-27]. Dostupné z: https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726

[35] *070-bernoulli-binomial.pdf* [online]. [vid. 2019-07-29]. Dostupné z: https://web.stanford.edu/class/archive/cs/cs109/cs109.1178/lectureHandouts/070-bernoulli-binomial.pdf

[36] *Unsupervised Feature Learning and Deep Learning Tutorial* [online]. [vid. 2019-07-27]. Dostupné z: http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

[37] *sparseAutoencoder_2011new.pdf* [online]. [vid. 2019-07-29]. Dostupné z: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf

[38] ML | K-Medoids clustering with example. *GeeksforGeeks* [online]. 17. květen 2019 [vid. 2019-08-12]. Dostupné z: https://www.geeksforgeeks.org/ml-k-medoids-clustering-with-example/

[39] YU, Donghua, Guojun LIU, Maozu GUO a Xiaoyan LIU. An improved K-medoids algorithm based on step increasing and optimizing medoids. *Expert Systems with Applications* [online]. 2018, **92**, 464–473. ISSN 0957-4174. Dostupné z: doi:10.1016/j.eswa.2017.09.052

[40] BHAT, Aruna. K-Medoids Clustering Using Partitioning Around Medoids for Performing Face Recognition. *International Journal of Soft Computing, Mathematics and Control* [online]. 2014, **3**(3), 1–12. ISSN 22014160. Dostupné z: doi:10.14810/ijscmc.2014.3301

[41] LI, Xue. K-Means and K-Medoids. *Encyclopedia of Database Systems* [online]. 2009, 1588–1589. Dostupné z: doi:10.1007/978-0-387-39940-9_545

[42] NG, Raymond T a Jiawei HAN. Efficient and Effective Clustering Methods for Spatial Data Mining. nedatováno, 12.

[43] LUTINS, Evan. DBSCAN: What is it? When to Use it? How to use it. *Medium* [online]. 15. září 2017 [vid. 2019-08-12]. Dostupné z: https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818

[44] *BIL0059_FEI_N2647_2612T025_2018.pdf* [online]. [vid. 2019-08-12]. Dostupné z: https://dspace.vsb.cz/bitstream/handle/10084/128336/BIL0059_FEI_N2647_2612T025_2018.pdf?sequence=1&isAllowed=y

[45] Density-Based Clustering. *Data Science Blog by Domino* [online]. [vid. 2019-08-12]. Dostupné z: https://blog.dominodatalab.com/topology-and-density-based-clustering/

[46] What is Python? Executive Summary. *Python.org* [online]. [vid. 2019-08-14]. Dostupné z: https://www.python.org/doc/essays/blurb/

[47] RALHAN, Abhinav. Self Organizing Maps. *Medium* [online]. 17. září 2018 [vid. 2019-08-14]. Dostupné z: https://towardsdatascience.com/self-organizing-maps-ff5853a118d4