**ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE**

**FAKULTA
STROJNÍ**

**DIPLOMOVÁ
PRÁCE**

2020

NÁVRH VYUŽITÍ STROJOVÉHO UČENÍ V MODULU PREDIKTIVNÍ ÚDRŽBY

**KATEŘINA
MÄRZOVÁ**

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Märzová** Jméno: **Kateřina** Osobní číslo: **459973**

Fakulta/ústav: **Fakulta strojní**

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

Studijní program: **Průmysl 4.0**

Studijní obor: **bez oboru**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Návrh využití strojového učení v modulu prediktivní údržby

Název diplomové práce anglicky:

Machine Learning Proposal for Utilization in Predictive Maintenance Module

Pokyny pro vypracování:

Navrhněte využití algoritmů strojového učení pro monitorování a vyhodnocování stavů stroje nebo celé výrobní linky jako základ modulu detekce a predikce poruchových stavů a prediktivní údržby. Monitorovaným systémem (procesem) může být reálné zařízení nebo simulační systém výrobního procesu. Na základě rešerše, analýzy procesu, předzpracování dat (detekce a odstranění odlehlých hodnot, nefunkčních sensorů,...),algoritmů analýzy dat (lineární a nelineární míry korelace, vyhodnocení šumu, statistiky plovoucího okna, rekurentní grafy,...) a metod strojového učení (shluková analýza, samoučící a kompetetivní algoritmy(SOM), autoenkodéry, regresory, supervizorované neuronové sítě pro klasifikaci,...), zvolte vhodnou metodiku a algoritmy pro monitorování a vyhodnocování procesu, tj., pro možnou detekci a predikci poruch pokud možno se širším analytickým potenciálem pro daný proces , např. i pro předem neznáme poruchy/stavy a jejich možné interpretace z dat. Algoritmy můžete realizovat vlastním kódem, nebo můžete využít existující moduly.

Seznam doporučené literatury:

[1] Data Clustering : Algorithms and Applications, edited by Charu C. Aggarwal, and Chandan K. Reddy, CRC Press LLC, 2013. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/cvut/detail.action?docID=1355921.
[2] David B. Fogel, Derong Liu ,. James M. Keller. Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation
[3] Mařík, V. a kol.: Umělá inteligence V, Academia Praha 2008

Jméno a pracoviště vedoucí(ho) diplomové práce:

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

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

Datum zadání diplomové práce: **30.04.2020** Termín odevzdání diplomové práce: **07.08.2020**

Platnost zadání diplomové práce: _____

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

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

29.07.2020
Datum převzetí zadání

Podpis studentky

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracovala samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uvedena jako její spoluautorka.

V Praze dne ..... Podpis .....

Poděkování

Děkuji vedoucímu své diplomové práce doc. Ing. Ivu Bukovskému, Ph.D. za vedení této práce, za možnost zpracovávat toto zajímavé téma, za čas, který mi věnoval, za trpělivost a za všechno, co jsem se od něj naučila. Dále děkuji Ing. Petru Svobodovi, Ph.D. a firmě mySCADA s.r.o. za poskytnutý software, data a knowhow, o které se při konzultacích podělili.

V neposlední řadě děkuji své rodině za veškerou podporu během celého studia.

Abstrakt

Cílem této práce je navržení využití algoritmů strojového učení pro vytvoření modulu prediktivní údržby. Práce se zabývá algoritmy shlukové analýzy za účelem analýzy vstupních dat a jejich možným využitím v prediktivní údržbě. Práce popisuje aplikaci jednotlivých metod na data ze simulátoru výrobní linky (od firmy mySCADA s.r.o.) a na umělá data. Závěrem práce je návrh využití popsaných metod v modulu prediktivní údržby pro detekci anomálií a predikci poruchových stavů.

Abstract

This thesis aims to propose the use of machine learning algorithms to design a predictive maintenance module. Cluster analysis algorithms are examined to analyse input data and to decide about their possible utilization in predictive maintenance. The work describes the application of several methods on the data from the production line simulator (company mySCADA s.r.o.) and artificial data. The conclusion of this thesis is a proposal of usage of described methods in predictive maintenance module to detect anomalies and predict fault states.

Klíčová slova

Prediktivní údržba, strojové učení, shluková analýza, detekce anomálií

Key Words

Predictive maintenance, Machine learning, Cluster analysis, Anomaly detection

Zkratky

PCA - Principal Component Analysis

TPS - Toyota Production System

IMSS - Intelligent Maintenance Support System

IIMSS - An Integrated Intelligent Management Support System

IDSS - Intelligent Decision Support System

CPN - Causal Probability Network, neboli Bayesovské sítě

HUGIN - Handling Uncertainty In General Inference Network

ART - Adaptive Resonance Theory

STR - string, datový typ, typicky sekvence znaků

INT - integer, celočíselný datový typ

BOOL - boolean, logický datový typ

CF - cílová funkce

BMU - best matching unit

Značení

C_X - kovarianční matice stavu X

λ_i - vlastní čísla matice

v_i - vlastní vektory matice

$\overline{X}$ - matice středních hodnot

σ_x - směrodatná odchylka matice X

w_{ij} - váhy

$s(i)$ - silueta

SC - siluetový koeficient

CH - Calinski-Harabaszův koeficient

μ - koeficient učení

μ_j - centroid pro metodu k-Means

$\beta_{ij}(k)$ - topologické okolí BMU (SOM)

$\sigma(k)$ - poloměr topologického okolí BMU (SOM)

$\alpha(k)$ - koeficient učení v SOM

N - počet vzorků

$\overline{QE}$ - střední chyba kvantizace

Obsah

1 Rešerše

1.1 Prediktivní údržba

Téma údržby rezonuje celým výrobním sektorem. Ačkoli se v jednotlivých odvětvích čísla liší, celkové náklady na údržbu se pohybují v rozmezí $15-60\%$ [1]. Údržba se definuje různě, například dle normy ČSN EN 13306 je to kombinace všech technických, administrativních a manažerských opatření během životního cyklu objektu, zaměřených na jeho udržení ve stavu nebo jeho navrácení do stavu, v němž může vykonávat požadovanou funkci [2].

K údržbě lze přistupovat několika způsoby. Tradiční metoda v anglické literatuře označovaná jako $Run-To-Failiure\ Maintenance$, také $Break-Down\ Maintenance$, představuje jakousi první generaci v přístupu k údržbě. Jak vypovídá název, přístup je založen na rychlé opravě po poruše. Hlavní nevýhodou tohoto přístupu jsou neplánované odstávky, což je problém zejména v sériové, nebo hromadné výrobě. Další nevýhodou může být omezená dostupnost náhradních dílů. Při neplánované údržbě nelze držet efektivní množství a druh potřebných náhradních dílů a tím pádem dochází buď k držení zbytečného množství skladových zásob, nebo k prostojům při opravách poruch. [3][1]

Další přístup, který se začal objevovat asi v polovině minulého století, je preventivní údržba (angl. Preventive Maintenance) a představuje druhou generaci v oboru údržby. Tyto metody jsou založeny na pravidelných prohlídkách zařízení v předem stanovených intervalech. Základním kritériem pro plánování je čas, po který je zařízení v provozu. Tento přístup je dodnes rozšířený v provozech s nízkou automatizací. Programy preventivní údržby jsou většinou dány vnitřními předpisy firmy a předepisují zejména čištění, mazání a kontrolu opotřebení dílů. [4][1]

Třetí generaci představuje prediktivní údržba (angl. Predictive Maintenance). Stejně jako v předchozích případech existuje velké množství variant. Obecně lze říci, že je metoda založena na aktivním monitoringu skutečného stavu zařízení a jeho využití k maximalizaci intervalů mezi potřebnými údržbářskými zásahy. Běžně se sledují vibrace, teploty, elektrické veličiny, výkony, apod. Hlavní výhody správně nastavené prediktivní údržba jsou snížení nákladů spojené s nepotřebným servisem a výměnami dílů, redukce prostojů stroje a snížení nákladů spojenými se skladováním přebytečných náhradních dílů. [1][5]

Prediktivní údržba je úzce spojena s komplexním monitorováním procesů, a to nejen ve vztahu k údržbě, ale zejména ve vztahu k řízení kvality. Prediktivní údržba je také součástí konceptu štíhlé výroby (angl. Lean Production). Metody štíhlé výroby se zaměřují na odstranění aktivit, které nepřidávají hodnotu (plýtvání), a to jak ve výrobě, tak v managementu. Plýtvání ve výrobě lze rozdělit do několika skupin [6][1]:

1. velké zásoby,
2. prostoje, čekání,

3. nadbytečná výroba,
4. neefektivní kontrola kvality,
5. opravy a přepracování, zmetky,
6. neefektivní pohyby a manipulace,
7. nevyužitá kreativita pracovníků,
8. neefektivní údržba.

Celý koncept *lean* vychází z Toyota Production System (TPS), který byl po druhé světové válce vyvíjen v Toyota Motor Company. TPS se jako první zabývá identifikací zdrojů plýtvání a jejich efektivním odstraněním [6]. Na TPS dále navazují další metody, jako 5S, Kaizen, Just-In-Time, Kanban, Total Productive Maintenance, Six Sigma, 3M, Jidoka. Každá z těchto metod se zabývá určitou částí výrobního procesu: standardizace, inovace, plánování výroby, řízením kvality a kontrolou, údržbou, atd. [7].

1.2 Metody prediktivní údržby

Zádkladní metody prediktivní údržby lze rozdělit do skupin:

- znalostní - na základě zkušeností a znalostí procesů a strojů,
- analytické metody,
- Petriho sítě,
- Bayesovské metody,
- fuzzy metody,
- neuronové sítě.

Vybrané metody budou představeny v rešerši.

1.2.1 Znalostní metoda - IMSS

Mezi prvními, kteří se zabývali problematikou preventivní údržby, byl například Ming Rao s týmem kolem roku 1990. Zabývali se tvorbou podpůrného systému pro chytrou údržbu systémů kontroly a řízení letového provozu, který nazvali: Intelligent Maintenance Support System (IMSS). Zásadním bodem byla detailní expertní analýza stávajícího stavu, která následně poskytla potřebné „know-how“ pro tvorbu systému, který je založen na znalosti dané problematiky. IMSS lze popsat následujícími body [8]:

- Cíle systému - minimalizace nákladů, maximalizace zisku.

- Stavy systému - počet stavových proměnných, náhradních dílů, jednotek v provozu, ...
- Metodika řešení - diferenciální rovnice, dynamické a lineární programování, teorie řízení, ...
- Systémové znalosti - nejistoty, možnosti investování do vybavení a jeho cena, životnost dílů, cena údržby a provozu, cash-flow, četnost analýzy a kontrol, technologické predikce a možnosti substituce dílů, ...
- Rozhodování - oprava, výměna, kontinuální monitoring, destruktivní/nedestruktivní kontrola, ...
- Modelovací funkce - technologické změny, daně a úrokové sazby, údržba, ...
- Plánování - kontinuální, diskrétní, konečné, nekonečné.

Jádrem IMSS je tzv. „meta-systém“, který na základně vstupních dat rozhoduje o spouštění příslušných výpočetních algoritmů, naznačených v bodě metodika řešení. Jak je patrné z názvu, systém slouží jako podpůrný prostředek pro rozhodování a plánování údržbových činností [8].

1.2.2 Analytická metoda - IIMSS

Kevin Xiaoguo Zhu v roce 1996 navrhl podpůrný systém pro inteligentní integrovaný management údržby (An Integrated Intelligent Management Support System - IIMSS), který je založený na datech získávaných ze senzorů integrovaných v důlním vozíku. Data ze senzorů jsou následně zpracovávána znalostním modelem, který vyhodnocuje stav a optimalizuje interval údržby s ohledem na minimální náklady. Dále také poskytuje včasná varování při neočekávaném stavu a zabraňuje tak závažným poruchám [9]. Jde tedy o kombinaci znalostního datově založeného modelu s vyhodnocovací analýzou.

1.2.3 Prediktivní údržba s využitím Bayesovských sítí

Mezi prvními se využitím Bayesovských sítí pro podporu rozhodování zabývali v roce 1995 Y. L. Tu a E. H. H. Yeung [10], systém nazvali Intelligent Decision Support System (IDSS). Hlavními proměnnými, se kterým systém počítal, byly náklady, kvalita a efektivita výroby.

Svou práci následně rozšířili v roce 1997, kdy představili další IDSS systém, založený na kauzálních pravděpodobnostních sítích (Causal Probability Network - CPN), neboli Bayesovských sítích [11]. Pro modelování systému využili GRAI sítě, které byly v roce 1983 vyvinuty v laboratořích univerzity v Bordeaux (GRAI) [12]. V uzlech sítí, které představují rozhodovací centra, je prováděn výpočet pomocí systému HUGIN (Handling Uncertainty In General Inference Network), vyvíjeného na univezitě Aalborg v Dánsku od 80. let. V tomto systému se Bayesovská pravděpodobnost vyjadřuje jako funkce počtu uzlů a pravděpodobnostních vazeb, ty jsou vyjádřeny podmíněnými pravděpodobnostmi, které jsou pomocí Bayesovy věty určeny z historických dat, označují se jako

experimentální pravděpodobnosti [13] [11]. Rozhodovací centra sítě jsou celkem 4, každé centrum má pak vlastní CPN model, který rozhodne o stavech v hlavních centrech:

1. Rozhodnutí o údržbě - `M _ decision`:
   (a) stav `corrective` - neplánované zásahy omezující výrobu,
   (b) stav `condition` - preventivní zásahy,
   (c) stav `chance` - zásahy prováděné v případě, že stroj není v provozu (mimo směnu, při dovolených, při prostojích stroje, apod.).
2. Kvalita výrobků - `Quality`:
   (a) stav `accepted`,
   (b) stav `rejected`,
   (c) stav `limited`.
3. Efektivita - `Efficiency`:
   - Efektivita výroby byla rozdělena do 4 intervalů: 0%, kdy porucha způsobí zastavení stroje, $(0-80)\%$ značící poruchu, která nevede k přerušení chodu stroje, $\langle 80-90)\%$ indikuje potenciální poruchu a v intervalu $\langle 90-100\rangle\%$ pracuje stroj v normálních podmínkách.
4. Náklady - `Cost`:
   - Nákladový model má 9 uzlů, které jsou přizpůsobeny dané společnosti, vyhodnocují například prostoje, úroveň odbornosti zaměstnanců, náklady na náhradní díly apod.

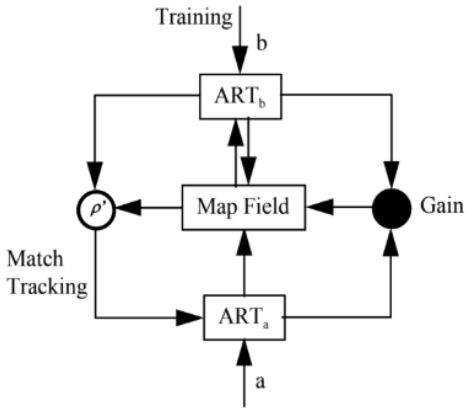
Systém byl vyvíjen a provozován v textilní továrně Central Textile (HK) Ltd. v Hong Kongu. [11]

1.2.4 Prediktivní údržba s využitím neuronových sítí

Společně s rozvojem a intenzivním zkoumáním neuronových sítí, se začínají objevovat první metody prediktivní údržby, které neuronové sítě využívají.

Osvědčily se zejména klasifikátory se zpětnou propagací učení. Mezi prvními je potřeba zmínit ART (Adaptive Resonance Theory), vyvinutou S. Grossbergem a G. Carpenterem. Původní základní nesupervizorovaná síť se skládala ze dvou polí neuronů: rozpoznávací a porovnávací, parametru (angl. vigiliance parameter) a resetovacího modulu. [14]

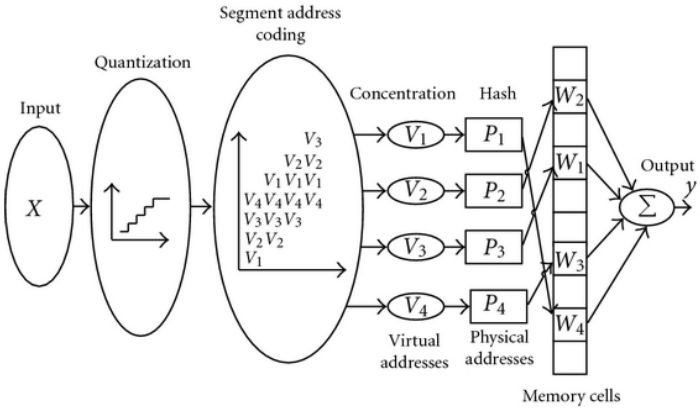
Metodu ART 1 (odvozenou od původní teorie ART) využili například R. J. McDuff et al. [15] při vývoji systému pro diagnózu linie letu letounu F-16 v roce 1989. Zmiňují její výhody oproti klasickým sítím (Backpropagation Neural Networks a Linear Vector Quantization Networks) zejména v online učení, tedy schopnosti rozlišovat mezi novými a známými vzory v datech.



Obrázek 1: Schéma architektury ARTMAP, převzato z [17]

Knapp et al. 1992 [16], který se zabýval klasifikací poruch strojů na základě měření vibrací, vyvinul novou metodu supervizorované klasifikace. Architektura sítě se označuje jako ARTPMAP, jde v podstatě o 2 ART moduly propojené s mapovacím polem a vnitřním kontrolerem. ART_a obdrží matici vzorů $\left[a^{(p)}\right]$ a ART_b obdrží matici vzorů $\left[b^{(p)}\right]$. Ve fázi trénování se výsledky z $a^{(p)}$ porovnávají a opravují správným výsledkem $b^{(p)}$. Ve fázi testování jsou zbylé $a^{(p)}$ prezentovány bez $b^{(p)}$ a jejich predikce v ART_b jsou porovnávány s $b^{(p)}$. Parametr ρ určuje míru shody mezi rozeznanými kategoriemi v ART_a a ART_b. Schéma ARTMAP je na obr. 1.

Lin et al., 1996 [18] poukazuje na nedostatky zpětné propagace pro reálně-časové aplikace, zejména na pomalou rychlost konvergence a s tím související nemožnost přírůstkového učení (tzn. že se koeficient učení mění v průběhu výpočtu a přispívá tak ke snazšímu nalezení řešení). Navrhuje použití CMAC (Cerebellar Model Articulation Controller), který byl původně představen jako matematický model v roce 1975 J. S. Albusem [19]. Albus se nechal inspirovat zpracováváním informace v mozečku a definoval CMAC jako sérii mapování, zobrazena je na obr. 2. Výsledky této



Obrázek 2: Schématické zobrazení CMAC, převzato z [20]

studie zdůrazňují zejména exponenciální rychlost učení a možnost určení závažnosti závady, jelikož

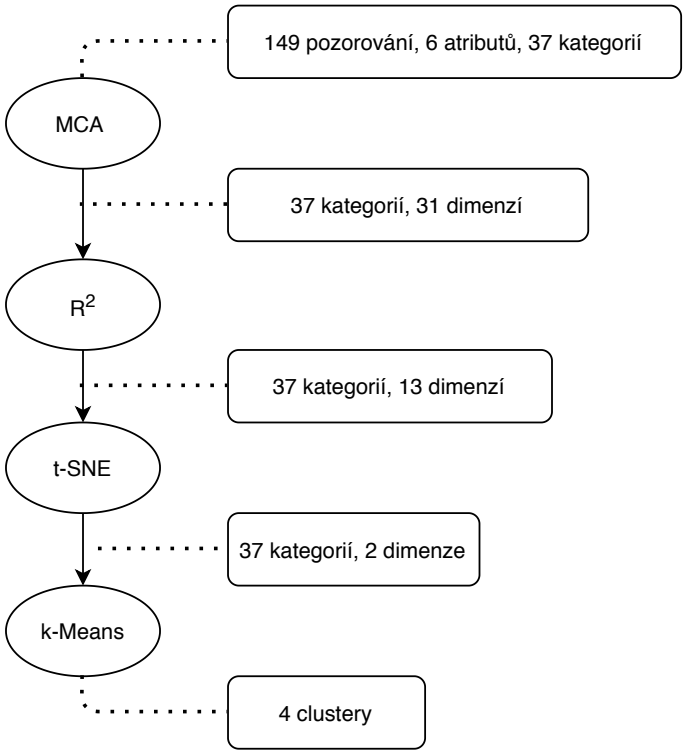
je algoritmus schopen lépe interpolovat funkční závislosti v datech [18].

V současnosti se lze setkat jak se systémy založenými na jednoduchých clusteringových metodách, jako PCA nebo k-Means [21], tak na sofistikovaných metodách hlubokého učení (angl. Deep Learning), jako např. LSTM (angl. Long Short-Term Memory) [22], DBN (angl. Deep Belief Network) [23], atd.

Další oblíbenou metodou je SOM, viz odst. 4.1.5. H. Du et al. 2002 [24] se zabýval predikcí teploty oleje transformátoru ve vnitřním prostředí. Predikci provedl pomocí výpočtu tradiční metodou dle platných Japonských norem, v tomto výpočtu se potýkali se 2 konstantami, které nedokázali přesně určit. V druhém kroku byla provedena predikce pomocí SOM, pomocí které se podařilo přesnost výsledků predikce výrazně zlepšit, průměrná chyba odhadu pomocí SOM se pohybovala mezi $1 - 2,3\%$, oproti tomu průměrná chyba pomocí konvenčního výpočtu dosahovala až $5 - 8\%$.

J. Chen et al. 2007 [25] se zabývali tvorbou modelu na predikci spolehlivosti náhradních dílů v letectví. SOM byla použita ke clusteringu dat selhání (angl. failiure data - zde použit časový interval mezi poruchami) a predikci spolehlivosti. Následně byl vytvořen spolehlivostní model pomocí Weitbullova rozdělení.

Zajímavý přístup k prediktivní údržbě nabízí také např. K. Dhalmahapatra et al. 2019 [26]. Vyvinul DSS (angl. Decision Support System), který hledá závislosti ve vstupních datech a třídí je do 4 kategirií. Vstupní data obsahují aktivitu, její dopad, příčinu a časové údaje. Schéma metody je zobrazeno na obr. 3. Vstupní data $X[149, 6]$ obsahující 37 kategorií jsou zpracovány pomocí multikriteriální analýzy MCA pro identifikaci kategorií, pro redukci dimenze je použita metoda lokte nebo R^2. Připravená data jsou následně metodou t-SNE, zredukována do 2 dimenzí a zobrazena pomocí k-Means. Ve výchozích clusterech jsou následně identifikovány subclustery pomocí χ^2 vzdálenosti od středu clusteru. Tyto subclustery následně poskytly nové informace pro zlepšení plánování, údržby a bezpečnosti.



Obrázek 3: Diagram metodiky zpracování dat systému pro podporu rozhodování - K. Dhalmahapatra et al. 2019, převzato z [26]

2 Vývojové prostředí

Pro tuto práci byl zvolen programovací jazyk Python. Python je vysokoúrovňový skriptovací programovací jazyk, který byl vyvynut v roce 1991 Guido van Rossumem. Python je multiparadigmatický, umožňuje objektové, funkcionální, imperativní i procedurální programování.

Důvody pro výběr Pythonu jako programovacího jazyku pro tuto práci:

- Python je open source, jeho instalační balíčky, včetně velkého počtu knihoven, jsou zdarma a jsou podporovány na všech běžně rozšířených platformách (Unix, MS Windows, Android, macOS).
- Součástí nejvíce rozšířené distribuce Anaconda, je editor Spyder, který je optimalizován pro vědecké výpočty a projekty.
- Ke knihovnám je dostupná detailně zpracovaná dokumentace pro snadnou a jasnou implementaci kódu. Zároveň je k dispozici množství modelových případů použití jednotlivých funkcí.
- Součástí nejběžněji používané knihovny NumPy jsou téměř všechny algebraické operace, lineární algebra a goniometrické funkce. Knihovna Scipy dále obsahuje třídu `stats` pro statistické výpočty a `signal` pro výpočty spojené se zpracováním signálu. Funkce knihoven jsou většinou programovány v jazyce C++, což zajišťuje vyšší rychlosti výpočtu.

- Python umožňuje i tvorbu vlastních knihoven, tříd a funkcí, umožňuje snadnou práci s různými datovými typy.
- Knihovna Matplotlib je výborným nástrojem pro vizualizaci výsledků, obsahuje velké množství různých grafů a umožňuje jejich snadné přizpůsobení.

3 Analýza systému a vstupních dat

Tato diplomová práce vznikla ve spolupráci s firmou mySCADA s.r.o., která dodala software pro simulování dat, která by měla mít vysokou shodu s reálnými daty, běžně získávanými z výrobních linek.

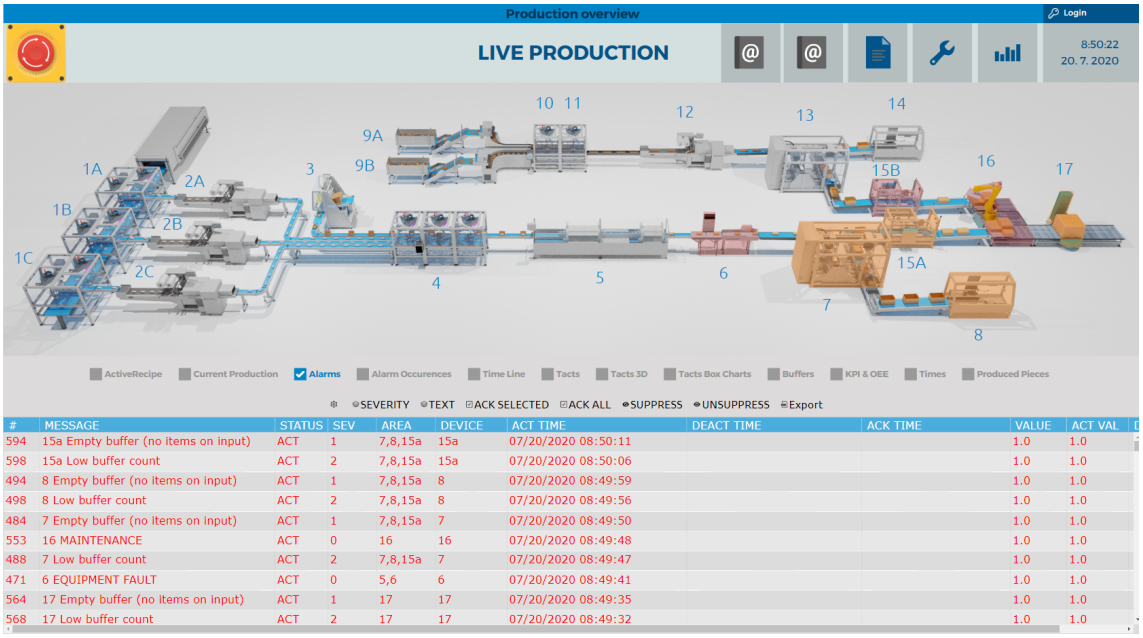
3.1 Popis analyzovaného systému

Analyzovaným systémem je simulovaná automatická výrobní linka pro balení výrobků. Výrobní linka je vytvořena v softwaru myDESIGNER firmy mySCADA s.r.o. Výrobní linka se skládá z 23 stanovišť, viz obr. 4. Stanoviště 1a až 1c jsou zařízení zajišťující sběr výrobků a jejich umístění na pás. Stanoviště 2a až 2c balí výrobky do fólie, stanoviště 3 připravuje krabice a stanoviště 4 umisťuje výrobky omotané fólií do krabice ze stanoviště 3, tyto krabice jsou následně zavřeny na stanovišti 5 a na stanovišti 6 je nalepen štítek. Na stanovišti 8 se připravují větší krabice a na stanovišti 7 dochází k balení menších oštítkovaných krabic do větších krabic ze stanoviště 8. Proces v druhé větvi je podobný, nejprve třídění vstupních výrobků ze zásobníků 9a a 9b na stanovištích 10 a 11, na stanovišti 12 probíhá balení výrobků do fólie, stanoviště 14 připravuje krabice, 13 vkládá výrobky obalené fólií do krabic a stanoviště 15b krabice zavírá. Výrobky ze stanovišť 15a a 15b jsou následně umístěny na palety na stanovištích 16 a 17.

Chod linky je řízen generátorem alarmů z připravené databáze. Každou sekundu se náhodným generátorem vygeneruje alarm pro 1 stanoviště (viz tabulka 1), vygenerovaný alarm změní hodnoty „měřených“ veličin a čeká na svou deaktivaci. Simulace se spouští na lokálním serveru. Data ze simulace lze exportovat v jednotlivých datalozích ve formátu .csv.

Na každém stanovišti jsou v čase zaznamenávány následující hodnoty:

- rychlost,
- časy,
  - stanoviště v chodu,
  - stanoviště čeká,
  - stanoviště stojí,
- status stanoviště.



#	MESSAGE	STATUS	SEV	AREA	DEVICE	ACT TIME	DEACT TIME	ACK TIME	VALUE	ACT VAL
594	15a Empty buffer (no items on input)	ACT	1	7,8,15a	15a	07/20/2020 08:50:11			1.0	1.0
598	15a Low buffer count	ACT	2	7,8,15a	15a	07/20/2020 08:50:06			1.0	1.0
494	8 Empty buffer (no items on input)	ACT	1	7,8,15a	8	07/20/2020 08:49:59			1.0	1.0
498	8 Low buffer count	ACT	2	7,8,15a	8	07/20/2020 08:49:56			1.0	1.0
484	7 Empty buffer (no items on input)	ACT	1	7,8,15a	7	07/20/2020 08:49:50			1.0	1.0
553	16 MAINTENANCE	ACT	0	16	16	07/20/2020 08:49:48			1.0	1.0
488	7 Low buffer count	ACT	2	7,8,15a	7	07/20/2020 08:49:47			1.0	1.0
471	6 EQUIPMENT FAULT	ACT	0	5,6	6	07/20/2020 08:49:41			1.0	1.0
564	17 Empty buffer (no items on input)	ACT	1	17	17	07/20/2020 08:49:35			1.0	1.0
568	17 Low buffer count	ACT	2	17	17	07/20/2020 08:49:32			1.0	1.0

Obrázek 4: Simulátor výrobní linky od firmy mySCADA

Datasety rychlostí a časů obsahují informaci o čase a aktuální hodnotě veličiny, veličiny jsou abstraktní. Současně s vygenerováním alarmu se změní hodnoty jednotlivých veličin dle předpisu simulátoru.

Dataset obsahující alarmy má atributy popsané v tabulce 1.

MESSAGE	str; popsané níže
VALUE	bool; alarm aktivní = 1, alarm neaktivní = 0
STATUS	ACT/OFF; alarm je/není aktivní
SEVERITY	int; závažnost stavu = (0, 1, 2)
AREA	str; dotčené oblasti
DEVICE	str; zařízení, na kterém je alarm aktivní
ACT TIME	time; čas aktivace alarmu
DEACT TIME	time; čas deaktivace alarmu
ACT VALUE	int; konstanta pro aktivaci alarmu = 1
DEACT VALUE	int; konstanta pro deaktivaci alarmu = 0

Tabulka 1: Popis dat obsažených v datasetu alarmy

Nejzajímavější atribut alarmu je `MESSAGE`, který nese vlastní informaci o nastalé události. Vybírá se z 8 kategorií, které jsou popsány níže, kompletní `MESSAGE` se potom skládá z označení stanoviště a kategorie alarmu, např. `15b Casing open` značí, že na stanovišti 15b je otevřený kryt stroje, apod.

- `MAINTENANCE` - probíhá údržba,

- `Low buffer count` - v zásobníku je málo dílů,
- `Empty buffer (no items on input)` - prázdný zásobník,
- `MATERIAL STUCK` - zaseknutý materiál,
- `Casing open` - otevřený kryt stroje,
- `Setup` - seřizování,
- `EQUIPMENT FAULT` - závada na vybavení stanoviště,
- `Sensor error` - porucha senzoru.

Dalším použitým datasetem jsou statusy stanoviště. Status stanoviště je určen v závislosti na aktuálním stavu, počtu a druhu náhodně vygenerovaných alarmů a hodnotách časů a rychlostí. Status 0 indikuje bezproblémový provoz. Kombinace alarmů nebo hodnot veličin, které ukazují na závady nebo zpomalení, ale nezastaví provoz linky, změní status na hodnotu 1. Pokud status zařízení dosáhne hodnoty 2, stanoviště stojí a vyžaduje akutní zásah (údržbu, opravu).

3.2 Předzpracování dat

Předzpracování dat (angl. preprocessing) je stěžejní úkol práce s daty. Zobecněně lze říci, že úkolem předzpracování je transformovat vstupní data do formátu, který může být jednoduše interpretován zvoleným algoritmem pro práci s daty.

Předzpracování dat mívá 3 fáze: integrace, transformace a redukce dat [27].

3.2.1 Integrace vstupních dat

Integrace vstupních dat se provádí tehdy, pochází-li data z různých zdrojů a musí být sjednocena do jednoho datasetu. Vstupní data mohou do procesu vstupovat v mnoha formátech od audio záznamů, obrázků, textových souborů, až po strukturované tabulky dat. Každý algoritmus vyžaduje specifický formát vstupních dat. Taková data mohou být nekonzistentní a obsahovat odlehlé nebo chybějící hodnoty [27].

Chybějící hodnoty mohou vzniknout jak při sběru dat (například výpadkem senzorů), tak i při jejich prvotním zpracování (validační algoritmy apod.) [28]. Chybějící hodnoty je nutné odstranit nebo odhadnout a nahradit. Odstraněním se ztratí část informace a je nutné zvážit, zda je to pro danou problematiku možné, je to ovšem jednodušší a s ohledem na pravdivost dat možná i korektnější řešení. Druhou možností je chybějící data odhadnout, jako podklad můžou sloužit podobné stavy v datasetu, popřípadě lze hodnoty určit pomocí statistických metod jako střední hodnota, medián nebo modus, další možností může být interpolace předchozí a následující hodnoty.

Dalším problémem, který je potřeba řešit, jsou odlehlé hodnoty. Grubbs definoval odlehlé hodnoty (angl. outliers) a rozdělil je podle jejich vzniku a zpracování do 2 skupin:

1. „Odlehlá hodnota (pozorování) může být pouze extrémní projev náhodné variability vlastní datům, pokud je to tak, měly by být i tyto hodnoty zachovány a zpracovány stejným způsobem jako ostatní pozorování v datasetu [29].“

2. „Odlehlá hodnota může být výsledným projevem hrubého odklonu od předepsaného postupu experimentu nebo chyba ve výpočtu nebo zaznamenání numerické hodnoty [29].“ Dále je třeba rozlišit, zda je známá fyzikální příčina vzniku těchto odlehlých hodnot.

Pokud je fyzikální důvod znám je možné hodnoty odstranit, opravit, nebo nahradit novými. Pokud není fyzikální důvod vzniku odlehlých hodnot znám je potřeba tyto hodnoty analyzovat a v závislosti na výsledku postupovat. K analýze lze použít „standardní“ statistické přístupy, většina z nich je založena na modelu Gaussova rozdělení. Další přístupy mohou být:

- založené na vzdálenosti - počítá se vzdálenost (nejčastěji euklidovská) mezi daným elementem a jeho sousedy v datasetu,

- založené na hustotě - počítá se hustota dat v jednotlivých částech datasetu, normální data mají konstantní hustotu, ovšem hustota odlehlých hodnot je rozdílná (např. algoritmus LOF) [30].

3.2.2 Transformace vstupních dat

Do této skupiny je vhodné zařadit operace jako je převzorkování, normalizace, potlačení šumu, agregace funkcí (v lit. označováno i jako kategorické mapování).

- Převzorkování se používá zejména ke zmenšení paměťové náročnosti výpočtu. K tomu lze použít několik strategií, vzorky se mohou vybírat z původního datasetu pomocí různých klíčů, od náhodně se měnící po konstantní periodu vzorkování. Vzorkovací perioda i strategie musí zajisit výběr konzistentního, reprezentativního vzorku dat [28].

- Normalizace je proces, kterým se mění „amplituda“ dat, obvykle se volí interval $\langle 0, 1\rangle$ nebo $\langle -1, 1\rangle$. Normalizace je nezbytná zejména pro algoritmy provádějící shlukovou analýzu [27].

- Agregace funkcí se provádí v případě, je-li možné sloučit více vzorků do jediného reprezentativního bodu s podobnými vlastnostmi bez ztráty informace o původních bodech [27] [28].

3.2.3 Redukce vstupních dat

Dataset vstupních dat musí být analyzován z hlediska obsahujících atributů, některé mohou být nerelevantní, nebo dokonce přebytečné a tím pádem je možné zvýšit efektivitu učení a zmenšit výpočetní náročnost. Pro výběr potřebných atributů existuje mnoho algoritmů, jako např. PCA, Absolute Minimum Shrinkage a LASSO metody, CFS, nebo metody založené na výběru dat a jejich atributů v závislosti na jejich entropii (Information Gain Methods).

3.2.4 Předzpracování experimentálního datasetu

Data jsou ze simulace exportována ve formátu .csv celkem ve 4 souborech: alarmy, rychlosti, časy, statusy. Prvním úkolem je identifikovat stav popisující analyzovaný systém. Stav X, sestavený ze vstupních veličin, musí určovat jednoznačně hodnotu výstupní veličiny y. Takto vytvořený stav je popsán ve vztazích (1) a (2).

$$X^T = \begin{array}{c} \begin{matrix} \text{k=1} & \text{k=2} & \dots & \text{k=N} \end{matrix} \\ \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,N} \\ x_{2,1} & x_{2,2} & \dots & x_{2,N} \\ x_{3,1} & x_{3,2} & \dots & x_{3,N} \\ x_{4,1} & x_{4,2} & \dots & x_{4,N} \end{bmatrix} \end{array} \begin{matrix} \\ \text{timeRun} \\ \text{timeWait} \\ \text{timeStop} \\ \text{Speed} \end{matrix} \tag{1}$$

$$y^T = \begin{array}{c} \begin{matrix} \text{k=1} & \text{k=2} & \dots & \text{k=N} \end{matrix} \\ \begin{bmatrix} x_{5,1} & x_{5,2} & \dots & x_{5,N} \end{bmatrix} \end{array} \begin{matrix} \\ \text{DeviceStaus} \end{matrix} \tag{2}$$

Soubory exportované ze simulátoru je třeba v souladu s odst. 3.2.1 nejprve integrovat do datesetů pro vstupní a výstupní veličiny.

Kompletní data jsou ve formě 2 slovníků (angl. dictionary), který obsahuje jednotlivá stanoviště. Každé stanoviště má data o svých stavech uložená ve formátu pandas.DataFrame, viz. [31]. Struktura pandas.DataFrame pro jednotlivá stanoviště je naznačena v tab. 2 a 3.

	timeRun1a	timeWait1a	timeStop1a	speed1a
0	60	0	0	32,88
1	47,42	0	12,58	0
⋮	⋮	⋮	⋮	⋮

Tabulka 2: Ukázka struktury pandas.DataFrame stanoviště 1a ze slovníku d1 pro vstupní hodnoty experimentálních dat z procesu

	DeviceStatus	Alarm
0	0	0
1	1	15a EQUIPMENT FAULT
⋮	⋮	⋮

Tabulka 3: Ukázka struktury pandas.DataFrame stanoviště 1a ze slovníku d2 pro vstupní hodnoty experimentálních dat z procesu

Takto vytvořené datasety je potřeba dále kontrolovat z důvodu chybějících dat. Prakticky to lze provést například pomocí funkce `pandas.DataFrame.isnull()`, která vrací booleanskou matici,

kde `False` je pro prvek s existující hodnotou a `True` je pro prvek s chybějící hodnotou. Jelikož data zpracovávaná v této práci jsou simulovaná, nejsou žádné chybějící hodnoty očekávané.

```
from Dictionary import Dictionary    # import

d1, d2 = Dictionary()                # nacteni promennych d1 a d2
                                     # z Dictionary

for i in range(0,23):                # pro i od 0 do 23
    df = d1['X{00}'.format(i)]       # df = dataframe stanoviste i
                                     # z d1
    check = df.isnull().sum().sum()  # celkovy pocet nulovych
                                     # hodnot pro cele stanoviste
    print('Station',i, '␣missing␣values',check)
```

Výsledek:

```
Station 0  missing values 0
Station 1  missing values 0
Station 2  missing values 0
Station 3  missing values 0
...        ...            ...
```

Analogicky lze rozbor provést pro druhý slovník d2. Po provedení rozboru se vyberou řádky obsahující `True` a z obou datasetů v obou slovnících se odstraní, využít lze například funkci `DataFrame.dropna(how = 'any')`, která vypustí všechny řádky, v nichž alespoň jedna hodnota chybí. Dataset lze ponechat s chybějícími řádky, nebo je lze nahradit novými pomocí různých metod, jako interpolace předcházejícího a následujícího, nebo pomocí statistických veličin.

Integrovaná data se dále transformují (viz. odst. 3.2.2). Normalizaci provedeme metodou z-score.

$$X = \frac{X - \overline{X}}{\sigma_X}, \tag{3}$$

kde $\overline{X}$ je matice středních hodnot a σ_X je směrodatná odchylka. Normalizována byla data ze slovníku d1, která budou předmětem zpracování.

Další možností pro transformaci dat je agregace funkcí. Pro časy jednotlivých stanovišť platí:

$$t_{run}(i) + t_{wait}(i) + t_{stop}(i) = 60, \ pro \ \forall i, \tag{4}$$

kde i je počet měření. Jelikož je součet všech 3 časů vždy konstantní, lze jeden z časů teoreticky vypustit a tím zredukovat množství dat. V praxi se ovšem tato úvaha nepotvrdila a tudíž byly zachovány všechny časy.

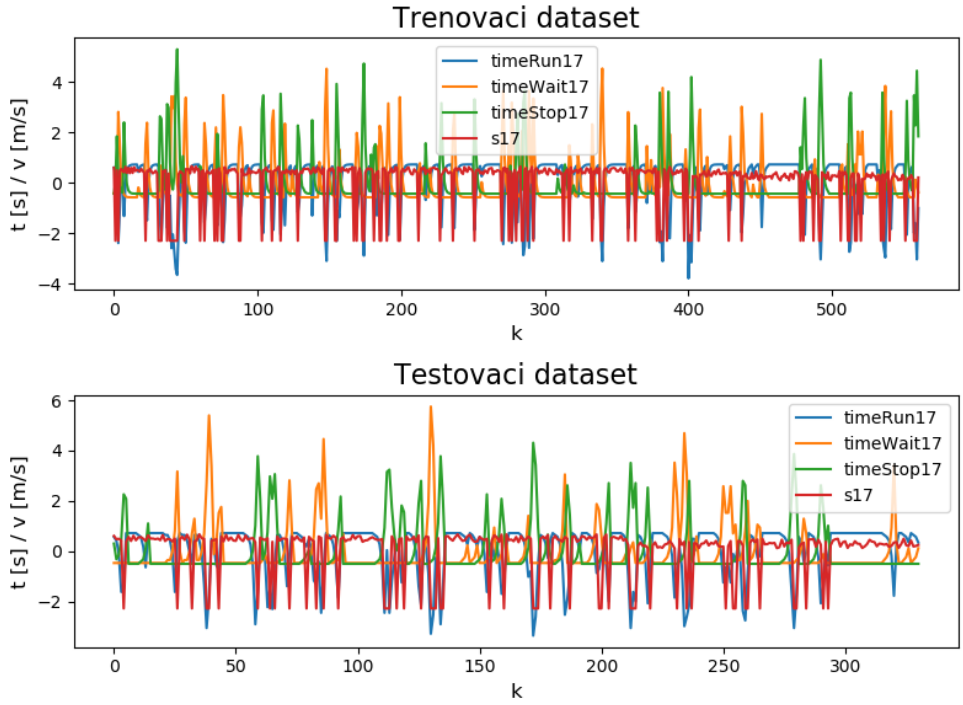
3.3 Předzpracovaný dataset

Na obr. 5 jsou zobrazena data po předzpracování dle sekce 3.2. Experimentální dataset se skládá ze dvou částí: trénovací a testovací. Trénovací dataset slouží k „natrénování“ algoritmu (neuronové sítě), zejména vah, testovací data jsou potom předmětem samotného zkoumání, nebo mohou sloužit jako validační pro kontrolu správnosti. Oba datasety byly připraveny stejným způsobem, trénovací data jsou simulována 21.března 2020 od 14:50 do 22.března 2020 01:08, dále 9:19 až 14:49 a 12.června 15:41 až 17:40, testovací data z 12.června 2020 15:41 až 17:40, 23.června 9:30 až 11:07, 26.června 11:08 až 13:01. Trénovací data obsahují 561 vzorků a testovací 331 vzorků s periodou 1 sekunda.

Vstupní data, například pro stanoviště 17 mají následující podobu:

$$X = \begin{matrix} \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{timeRun17} \\ \begin{bmatrix} timeRun17_{k=1} \\ timeRun17_{k=2} \\ \vdots \\ timeRun17_{k=N} \end{bmatrix} \end{matrix} \begin{matrix} \text{timeWait17} \\ \begin{bmatrix} timeWait17_{k=1} \\ timeWait17_{k=2} \\ \vdots \\ xtimeWait17_{k=N} \end{bmatrix} \end{matrix} \begin{matrix} \text{timeStop17} \\ \begin{bmatrix} timeStop17_{k=1} \\ timeStop17_{k=2} \\ \vdots \\ timeStop17_{k=N} \end{bmatrix} \end{matrix} \begin{matrix} \text{Speed17} \\ \begin{bmatrix} Speed17_{k=1} \\ Speed17_{k=2} \\ \vdots \\ Speed17_{k=N} \end{bmatrix} \end{matrix} \begin{matrix} \\ \text{k=1} \\ \text{k=2} \\ \vdots \\ \text{k=N} \end{matrix} \tag{5}$$

kde N je celkový počet stavů v datasetu.



Obrázek 5: Vstupní data z procesu ze simulátoru výrobní linky po předzpracování

3.4 Umělá data

Mimo dat získaných ze simulátoru výrobní linky budou algoritmy testovány také pomocí umělých dat. Umělá data mají stejný formát jako data simulovaná, skládají se z 3 časů (`timeRun, timeWait, timeStop`), 1 rychlosti (`speed`) a statusu stanoviště (`DeviceStatus`). Pro generování umělých dat byly využity poznatky z dat ze simulátoru, pravidla pro generování jednotlivých hodnot jsou naznačena v tabulce 4. Algoritmus je následující:

1. Generování vektoru DS, který představuje status stanoviště. Generuje se pomocí modulu random knihovny NumPy, konkrétně třída `NumPy.random.randint(a, b, N)` generuje vektor pseudo-náhodných čísel X o velikosti N, kde $a \leq X[i] \geq b$.

2. Určení časů a rychlosti.

   (a) Pro status 0; `timeRun` se generuje jako náhodné číslo mezi 50 a 60, `timeStop` je nula, `timeWait` se dopočítá, jako: `timeWait` $= 60 -$ `timeRun` $-$ `timeStop`. Rychlost `speed` je náhodné číslo mezi 40 a 60.

   (b) Pro status 1; `timeStop` se generuje jako náhodné číslo mezi 10 a 35, `timeWait` je nula, `timeRun` se dopočítá, jako: `timeRun` $= 60 -$ `timeWait` $-$ `timeStop`. Rychlost `speed` je náhodné číslo mezi 0 a 25.

   (c) Pro status 2; `timeWait` se generuje jako náhodné číslo mezi 10,35, `timeStop` je náhodné číslo mezi 0 a 10, `timeRun` se dopočítá, jako: `timeRun` $= 60 -$ `timWait` $-$ `timeStop`. Rychlost `speed` je náhodné číslo mezi 0 a 10.

`DeviceStatus == 0`			
`timeRun`	`timeWait`	`timeStop`	`speed`
$\langle 50, 60 \rangle$	$\langle 0, 10 \rangle$	0	$\langle 40, 60 \rangle$
`DeviceStatus == 1`			
`timeRun`	`timeWait`	`timeStop`	`speed`
$\langle 25, 50 \rangle$	0	$\langle 10, 35 \rangle$	$\langle 0, 25 \rangle$
`DeviceStatus == 2`			
`timeRun`	`timeWait`	`timeStop`	`speed`
$\langle 15, 50 \rangle$	$\langle 10, 35 \rangle$	$\langle 0, 10 \rangle$	$\langle 0, 10 \rangle$

Tabulka 4: Pravidla pro generování umělých hodnot

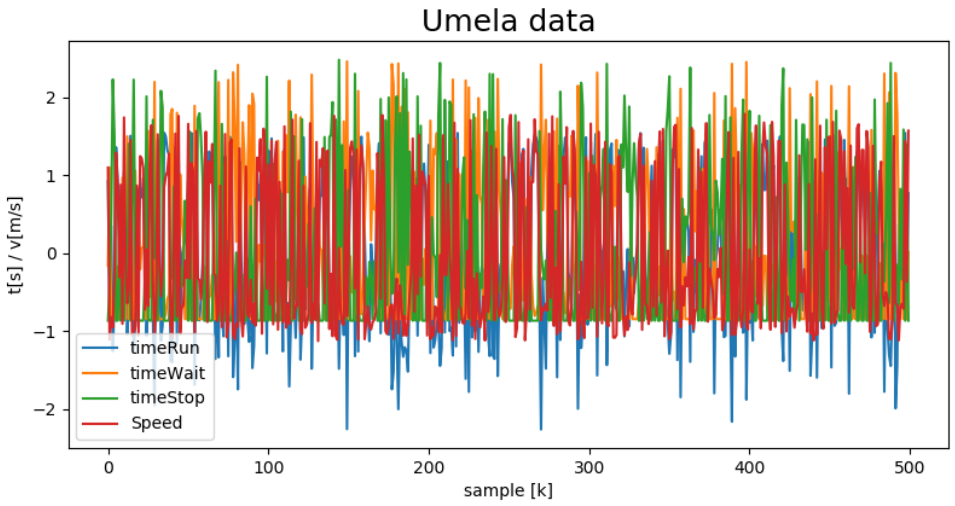
Konkrétní implementace v Pythonu pomocí knihovny NumPy je popsána níže:

```
import numpy as np                      # import knihovny NumPy

N = 500                                 # pocet vzorku
DS = np.random.randint(0, 3, N)         # tvorba vektoru nahodnych celych
                                        # cisel [0,3) o delce N (Device Status)

my_data = np.zeros((N, 5))              # matice nul o velikosti (Nx5)
my_data[:,-1] = DS                      # posledni sloupec je DS

for i in range(0,N):                    # pro i v rozsahu (0,N)
    if my_data[i,-1] == 0:              # pokud je DS pro i rovno 1
        my_data[i,0] = np.random.uniform(50,60)
                                        # prvni sloupec je nahodne cislo
                                        # v rozsahu 50 a  60
        my_data[i,2] = 0                # treti sloupec je 0
        my_data[i,1] = 60-my_data[i,0]
                                        # suma prvnich tri sloupcu
                                        # je 60
        my_data[i,3] = np.random.uniform(40,60)
                                        # ctvrty sloupec je nahodne
                                        # cislo v rozsahu 40 a  60
# analogicky pro DS == 1 a 2
    elif my_data[i,-1] == 1:
        my_data[i,2] = np.random.uniform(10,35)
        my_data[i,1] = 0
        my_data[i,0] = 60-my_data[i,1]-my_data[i,2]
        my_data[i,3] = np.random.uniform(0,25)
    elif my_data[i,-1] == 2:
        my_data[i,1] = np.random.uniform(10,35)
        my_data[i,2] = np.random.uniform(0,10)
        my_data[i,0] = 60-my_data[i,1]-my_data[i,2]
        my_data[i,3] = np.random.uniform(0,10)
```

Data vygenerována dle popsaného algoritmu jsou zobrazena v tabulce 5 a na obr. 6.

index	timeRun	timeWait	timeStop	Speed	DeviceStatus
0	52.817700	7.182300	0.000000	45.669658	0.0
1	35.140028	20.341584	4.518389	0.463482	2.0
2	46.246192	0.000000	13.753808	6.308846	1.0
3	27.796877	0.000000	32.203123	3.694732	1.0
4	35.486765	0.000000	24.513235	17.558161	1.0
⋮	⋮	⋮	⋮	⋮	⋮
495	44.192655	10.310350	5.496996	9.834027	2.0
496	34.493483	0.000000	25.506517	8.581535	1.0
497	59.898243	0.101757	0.000000	52.607969	0.0
498	36.492945	0.000000	23.507055	15.703439	1.0
499	51.018642	8.981358	0.000000	55.374550	0.0
[500 rows x 5 columns]					

Tabulka 5: Umělá data



Obrázek 6: Zobrazení umělých dat po normalizaci

4 Metody shlukové analýzy a jejich aplikace na data z procesu

4.1 Metody shlukové analýzy

Shlukové analýzy, neboli clustering, jsou metody, které se používají ke klasifikaci dat, data se třídí na základě různých metrik do skupin, které mají podobné vlastnosti, znaky, vzory, apod. Matematicky lze shlukovou analýzu popsat dle Kelbela a Šilhána [32]:

Nechť χ značí množinu n objektů. Rozklad $\Omega = C_1, C_2, \ldots, C_m$ množiny χ je množina disjunktních, neprázdných podmnožin χ, které dohromady tvoří χ. Tedy, pro $i \neq j$ platí, že $C_i \cap C_j = \emptyset$, $C_1 \cup C_2 \cup \cdots \cup C_m = \chi$. Každá množina C_i se nazývá komponentou rozkladu (cluster, shluk).

V této sekci budou představeny použité metody shlukové analýzy, popsán postup jejich využití a ten bude dále demonstrován na umělých datech.

4.1.1 PCA - metoda hlavních komponent

Metoda hlavních komponent (angl. Principal Component Analysis) byla poprvé popsána anglickým matematikem a statistikem Karlem Pearsonem na začátku 20. století. Podobné metody se používají v různých odvětvích, v mechanice SVD (angl. Singular Value Decomposition), v lineární algebře EVD (angl. Eigenvalue Decomposition), apod. [33].

PCA je metoda, která umožňuje zredukovat komplexní vícerozměrná data do méně dimenzí při zachování trendů, vzorů a vztahů v datech. Používá se jako metoda předzpracování dat, kde se využívá redukce dimenzí dat pro zmenšení datasetu a zrychlení výpočtu, toho se využívá například v biologii při analýze genomu [34]. Dalším častým využitím metody PCA je zpracování obrazu, používá se například pro kompresi obrázků, odstranění šumu nebo k detekci anomálií. Je to zároveň nejjednodušší nástroj shlukové analýzy.

Postup výpočtu je následující:

1. Normalizace nebo z-score matice vstupních dat, výpočet je uveden ve vztahu (3).
2. Výpočet kovarianční matice C_X:
$$C_X = XX^T \tag{6}$$
3. Výpočet vlastních čísel λ_i a vlastních vektorů v_i matice C_X.
4. Výběr vlastních vektorů, které odpovídají hlavním komponentám matice X.
5. Výpočet X_{pca}.
$$X_{pca} = P_m X, \tag{7}$$
kde P_m je matice skládající se z m vybraných vlastních vektorů.

Z algoritmu výpočtu je zřejmé, že správnost výpočtu ovlivňuje hlavně volba počtu hlavních komponent (vlastních vektorů). Množství informace (vysvětleného rozptylu) obsažené v jednotlivých komponentách lze kvantifikovat pomocí vlastních čísel λ_i.

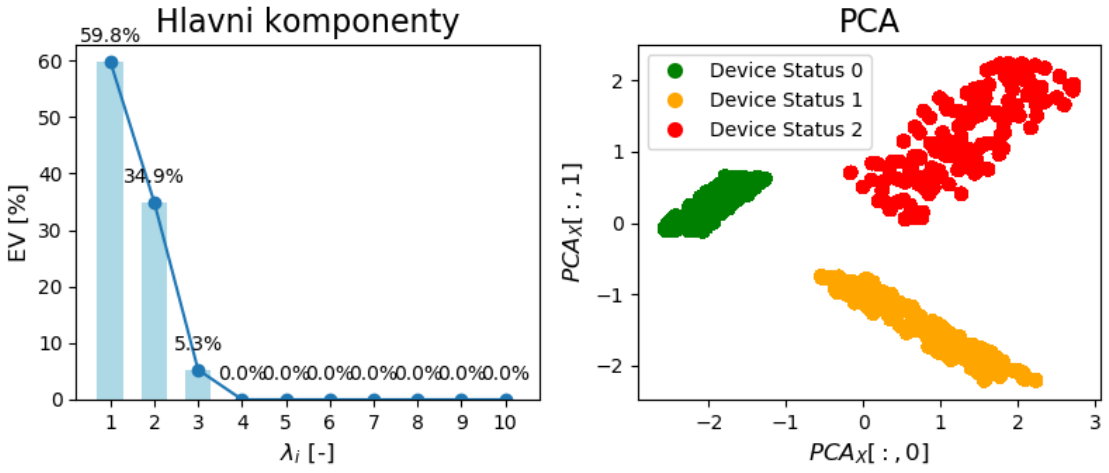
$$EV(i) = \frac{\mid \lambda_i \mid}{\sum \mid \lambda_i \mid} \cdot 100 \tag{8}$$

Metoda byla pro účely shlukové analýzy použita následovně: nejprve bylo dle vztahu (8) vypočítáno procento vysvětleného rozptylu pro jednotlivé komponenty, dále byla provedena komprese

datasetu pomocí metody PCA při použití 2 hlavních komponent a byla získána matice X_{PCA} popsaná v (7). Následně byl vynesen bodový graf závislosti $X_{PCA}[:,0]$ na $X_{PCA}[:,1]$ a jednotlivé body grafu byly zobrazeny v barvě závislé na statusu stanoviště pro jednotlivé stavy původních datasetů (status 0 - zelená, status 1 - oranžová, status 2 - červená).

Pro aplikaci algoritmu byla využita knihovna scikit-learn, modul `decomposition`, třída `PCA`, konkrétně:

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)          # priprava modelu
pcaX2D = pca.fit_transform(data)     # fitovani modelu pro 2 hlavni
                                     # komponenty a aplikace dimenzionalni
                                     # redukce na data
```



Obrázek 7: Grafické znázornění metody hlavních komponent na umělých datech, na levém obrázku je znázorněno množství informace obsažené v jednotlivých komponentách dle vztahu (8), v pravém grafu je metoda hlavních komponent pro 2 hlavní komponenty

Na obr. 7 je demonstrováno použití popsané analýzy pro umělá data, vlevo je zřetelně vidět, že cca 94,7 % informací o datasetu je obsaženo v prvních dvou hlavních komponentách. Na tomtéž obrázku vpravo, kde je zobrazena metoda PCA pro 2 hlavní komponenty, jsou 3 výrazně oddělené shluky.

4.1.2 Metoda k-Means a metody určení optimálního počtu clusterů

Algoritmus k-Means publikoval v roce 1967 James McQueen, který navázal na původní myšlenku Huga Steinhause z roku 1957. Algoritmus předpokládá, že vstupní množinu bodů lze popsat

v euklidovském prostoru a lze je roztřídit do předem definovaného počtu shluků (clusterů), tyto shluky jsou charakterizovány svými centroidy. Algoritmus postupuje následujícím způsobem [32]:

1. Náhodná inicializace centroidů μ_j.
2. Data x_i jsou přiřazena do shluků y_i na základě minimální euklidovské vzdálenosti bodu od centroidu: $y_i = \arg\min_j ||x_i - \mu_j||$.
3. Výpočet nových hodnot centroidů μ_{j+1}, jako těžiště bodů x_i přiřazených do daného shluku y_i, tedy: $\mu_{j+1} = \frac{1}{l_j} \sum_i (x_i)$, kde l_j je počet bodů x_i klasifikovaných do shluku y_i.
4. Kroky 2 a 3 se opakují po zvolený počet iterací, nebo dokud $\mu_{j+1} - \mu_j$ nedosáhne předem zvolené hodnoty.

Hlavní výhodou tohoto algoritmu je jednoduchost a rychlá konvergence k řešení v konečném počtu kroků, toto řešení je ovšem pouze lokálním optimem. Nevýhodou je zejména potřeba definování konkrétního počtu shluků k, do kterých se budou vstupní body přiřazovat. K rozhodnutí o hodnotě k lze použít několik metod, například metodu lokte (angl. Elbow Method), siluety, nebo Calinski-Harabaszovu metodu.

Mezi nejčastěji používané metody pro určení optimálního počtu shluků se řadí jednoznačně metoda lokte. Její název vychází z tvaru, který má její diagram, teorie říká, že v bodě zlomu (lokti) křivky loktového diagramu se nachází optimální počet clusterů. Loketní diagram se sestrojí tak, že se spočítá clustrovací algoritmus (např. k-Means) pro různý počet shluků k a pro každý počet k se spočítá celková suma čtverců vzdálenosti každého bodu x_i k jeho nejbližšímu centroidu μ_j a vynese se v závislosti na celkový počet shluků k.

Další často používanou metodou jsou tzv. siluety (angl. Silhouette Method). Tato metoda odráží míru konzistentnosti bodů uvnitř společného clusteru. Silueta $s(i)$ je vyjádřena jako [32]:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{9}$$

$a(i)$ vyjadřuje míru podobnosti dat x_i uvnitř clusteru y_i:

$$a(i) = \frac{1}{|y_i| - 1} \sum_{j \in y_i, i \neq j} ||x_i - x_j||, \tag{10}$$

kde x_i, x_j jsou body uvnitř clusteru y_i. $b(i)$ určuje míru odlišnosti bodu $x_i \in y_i$ od clusteru y_k, ve kterém bod x_i neleží:

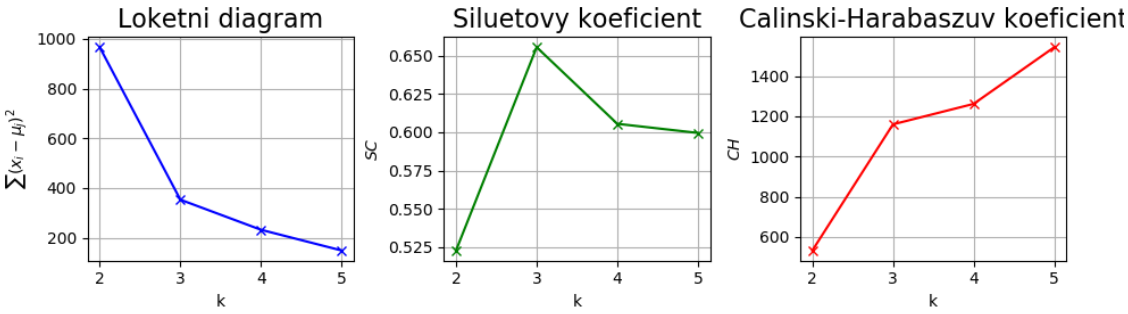
$$b(i) = \min_{k \neq i} \frac{1}{|y_k|} \sum_{j \in y_k} ||x_i - x_j|| \tag{11}$$

Siluety se zobrazují pro každý cluster y_i zvlášť, nebo se určuje siluetový koeficient, $SC = \max_k \widetilde{s}(k)$, kde $\widetilde{s}(k)$ je střední hodnota $s(i)$ pro všechny shluky y_i pro daný počet shluků k.

Třetí použitá metoda Calinski-Harabasz je velice podobná metodě siluet, výpočet indexu je následující [35]:

$$CH = \frac{(N-k) \sum_{j=1}^{k} \left(n_j ||\mu_j - \mu||^2 \right)}{(k-1) \sum_{j=1}^{k} \sum_{x_i \in y_i} \left(||x_i - \mu_j||^2 \right)}, \tag{12}$$

kde k je počet shluků, N je počet pozorování (počet bodů), n_j je počet bodů v daném shluku y_i a μ_j je jeho centroid, μ je střední hodnota dat v daném shluku. Pro určení optimálního počtu clusterů je potřeba vybrat maximální hodnotu Calinski-Harabaszova indexu přes různý počet clusterů.

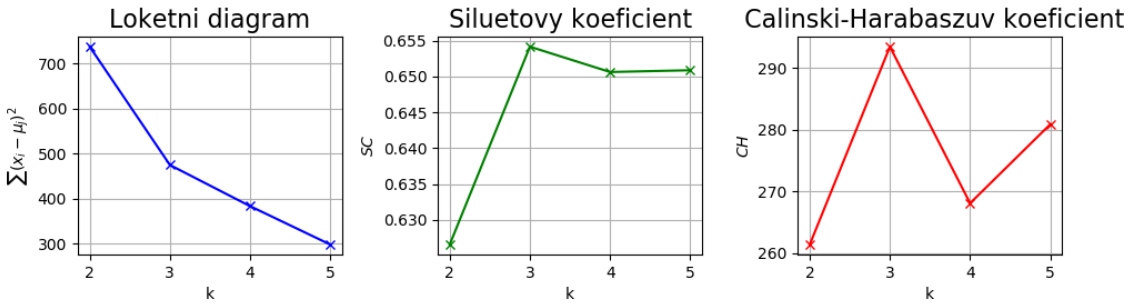


Obrázek 8: Grafické zobrazení tří metod pro určení optimálního počtu shluků pro umělá data, v loketním diagramu je patrný zlom pro $k = 3$, SC je maximální pro $k = 3$ a CH pro $k = 5$

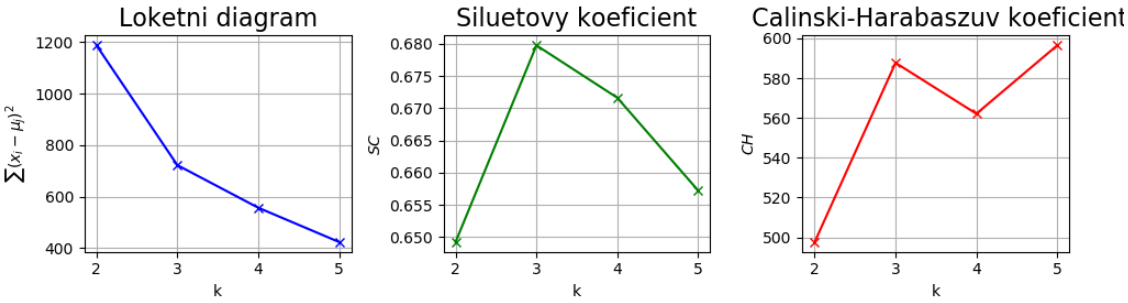
Na obr. 8 jsou zobrazeny výsledky uvedených metod pro umělá data. Na obr. 8 vlevo je metoda lokte, uprostřed siluetový koeficient a vpravo Calinski-Harabaszův index. Pro výpočet těchto diagramů, byla spočítána metoda k-Means pro $k = 2, 3, 4, 5$ shluků a pro každý případ byly spočítány uvedené koeficienty, koeficienty byly následně vyneseny do grafů v závislosti na k. Na prvním grafu vlevo, kde je zobrazen loketní diagram, je zřetelně vidět zlom („loket“) diagramu pro $k = 3$, lze tedy konstatovat, že optimální počet shluků pro tato data je 3. Podobně lze říci, že nejlepší výsledek pro siluetový koeficient je pro $k = 3$. Pro tento koeficient platí, že je-li $SC = 1$, jsou od sebe shluky dobře oddělené (mají dostatečnou euklidovskou vzdálenost) a data uvnitř shluku jsou konzistentní, naopak pro $SC = -1$ vykazují data z různých clusterů vysokou míru shody a zároveň nekonzistentnosti uvnitř jednoho clusteru, pro $SC = 0$ nejsou rozdíly mezi shluky dostatečně signifikantní pro učinění rozhodnutí o správnosti zařazení bodů do clusterů [36]. Pro 3 clustery je $SC(k = 3) = 0,655$, lze tedy usoudit, že data jsou dobře roztřízena do jednotlivých shluků a dané shluky jsou dostatečně rozdílné. Třetí použitá metoda je zobrazena na grafu vpravo, optimální počet shluků pomocí této metody je určen maximální velikostí Calinski-Harabaszova indexu, pro tento případ $CH(k = 5) = 1542,93$.

Dvěma ze tří použitých metod bylo dokázáno, že optimální počet shluků pro umělá data je 3.

Analogicky byla provedena analýza pro data z procesu, výsledky jsou zobrazeny na obr. 9 a 10. Stejně jako v předchozím případě je patrný zlom v loketním diagramu při $k = 3$ (graf vlevo) pro oba datasety. Siluetový koeficient dosahuje svého maxima taktéž pro 3 shluky, $SC(k = 3) = 0,6542$ pro testovací a $SC(k = 3)0,6798$ pro trénovací dataset. Nejvyšší hodnota Calinski-Harabaszova indexu byla v případě testovacích dat dosažena při $k = 3$, $CH(k = 3) = 293,49$ (obr. 9 vpravo), v případě trénovacích dat je to pro $k = 5$, $CH(k = 5) = 596,52$ (obr. 10 vpravo). Pro simulovaná data trénovací bylo obdobně jako pro umělá data dosaženo optimálního počtu shluků $k = 3$ dvěma ze tří testovaných metod, pro testovací data tento výsledek potvrzují všechny tři použité metody.



Obrázek 9: Grafické zobrazení tří metod pro určení optimálního počtu shluků pro testovací data, loketní diagram má zlom v $k = 3$ a koeficienty SC i CH dosahují maxima taktéž v $k = 3$



Obrázek 10: Grafické zobrazení tří metod pro určení optimálního počtu shluků pro trénovací data, loketní diagram má zlom v $k = 3$, SC má maximum v $k = 3$ a CH dosahuje maxima v $k = 5$

4.1.3 Shluková analýza a predikce stavu pomocí k-Means

Metoda k-Means umožňuje jako jedna z mála metod i predikci clusteru. Predikce probíhá tak, že je nejprve identifikován k-Means model, zejména jsou na trénovacích datech určeny polohy centroidů. Následně je počítána vzdálenost centroidů a nových (testovacích) dat. Na základě těchto vzdáleností je následně určena příslušnost bodů k jednotlivým clusterům.

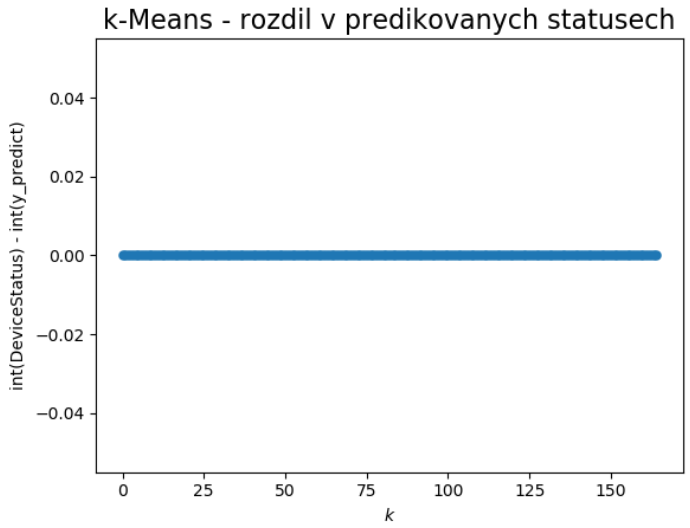
Umělá data byla rozdělena do trénovacího a testovacího datasetu pomocí metody z knihovny scikit learn. Konkrétně:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, target,
                                                    test_size=0.33, random_state=42)
```

Touto metodou je tedy rozdělena vstupní matice X s odpovídajícím vektorem `target`, obsahující status stanoviště příslušných stavů, na 2 datasety, kde 33% náhodně vybraných vzorků bude přiřazeno do datasetu X_{test} a odpovídajícímu vektoru y_{test}, zbytek bude tvořit trénovací dataset. K výpočtu byla použita taktéž knihovna scikt-learn následujícím způsobem:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3,          # priprava modelu
                max_iter=10000,        # maximalni pocet iteraci
                tol=0.000001)          # pozadovana presnost
KM = kmeans.fit_transform(X_train)     # fitovani modelu
                                       # a trenovani
y_pred = kmeans.predict(X_test)        # predikce clusteru
                                       # testovacich dat
```

Jelikož vstupní datasety jsou v obou případech čtyřdimenzionální, není možné zobrazit přímo výsledky. K porovnání výsledků a vytvoření představě o výsledcích predikce byly tedy vytvořeny grafy, kde na ose x je index vzorku a na ose y je rozdíl původního clusteru a predikovaného (clustery jsou v obou případech určeny celými čísly 0, 1, 2), jakákoli hodnota kromě 0 značí nekonzistentnost mezi původním a predikovaným clusterem.



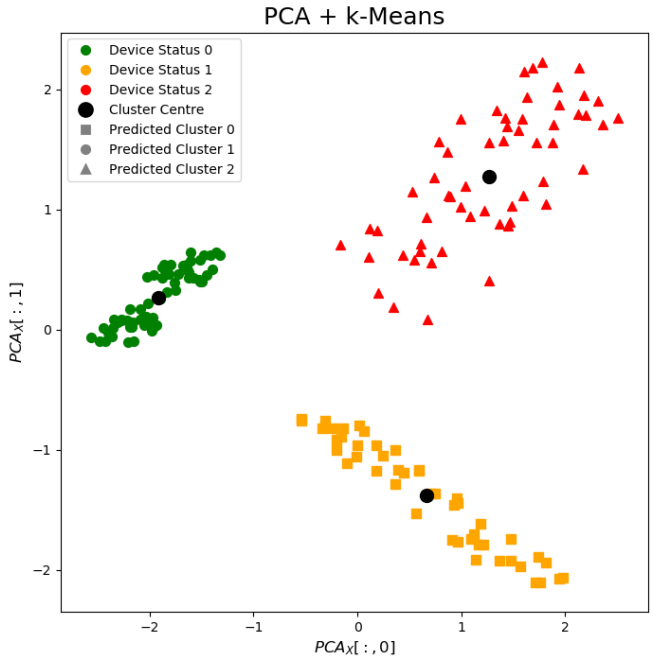
Obrázek 11: Rozdíl v predikovaných a původních clusterech na umělých datech - správná predikce všech stavů

Na obr. 11 jsou zobrazené výsledky predikce pro umělá data. Všechny predikované hodnoty jsou ve shodě s reálným statusem stanoviště.

Metodu k-Means lze ve spojení s metodou PCA využít i pro vizualizaci shlukové analýzy. Analýza byla provedena následujícím způsobem:

1. Komprimace původního datasetu metodou PCA; $X_{orig}(N \times 4) \longrightarrow X_{PCA}(N \times 2)$.
2. Výpočet algoritmu k-Means pro X_{PCA}, pro $k = 3$, natrénování na trénovacím datesetu a následně predikce příslušnosti stavů z testovacího datasetu k jednotlivým shlukům.
3. Grafické znázornění výsledků predikce pomocí bodového diagramu, cluster predikovaný metodou k-Means určuje tvar značky bodu a status stanoviště původního stavu její barvu.

Pro umělá data byl dataset nejprve komprimován do 2D pomocí PCA. Následně byl dataset rozdělen pomocí metody `model_selection.train_test_split` z knihovny scikit learn, jak již bylo uvedeno výše.



Obrázek 12: Analýza k-Means na umělých datech - vytvoření tří oddělených clusterů

Na obr. 12 je provedena analýza k-Means viz popis výše. Kromě zřetelně oddělených 3 shluků je vidět, že se barvy a tvary značek bodů nemíchají, a tudíž algoritmus umístil body z testovacího datesetu do stejných clusterů, které byly viditelné již při pouhé komprimaci datasetu do 2 dimenzí metodou PCA.

4.1.4 Metoda t-SNE

Metoda t-SNE (angl. t-Distributed Stochastic Neighbor Embedding) je nesupervizorovaná nelineární metoda, využívaná podobně jako metoda PCA, pro datovou analýzu a vizualizaci vysokodimenzionálních dat. Tuto metodu vynalezli Laurens van der Maaten a Geoffrey Hinton a představili ji ve svém článku Visualizing Data using t-SNE v roce 2008 [37].

Algoritmus metody t-SNE lze shrnout do tří kroků:

1. Prvním krokem je určení podobností mezi daty ve vysokodimenzionálním prostoru, k tomu je využito Gaussovo rozdělení:

$$p_{i|j} = \frac{exp(\frac{-\|x_i - x_j\|^2}{2\sigma^2})}{\sum_{k \neq 1} exp(\frac{-\|y_k - y_l\|^2}{2\sigma^2})} \tag{13}$$

   Dále se provede normalizace pro všechny body, která zajistí větší robustnost algoritmu vůči odlehlým hodnotám:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} \tag{14}$$

   Gaussovské rozdělení pro jednotlivé body lze ovlivnit perplexitou:

$$Perp(Pi) = 2^{H(P_i)} \tag{15}$$

   $H(P_i)$ je Shannonova entropie bodu P_i měřená v bitech:

$$H(P_i) = -\sum_j p_{j|i} log_2 p_{j|i} \tag{16}$$

   Perplexitu lze interpretovat jako měřítko efektivního počtu sousedů, obvyklá hodnota je mezi 5 a 50.

2. Dále se analogicky jako v prvním kroku počítá párová podobnost v nízkodimenzionálních datech. V tomto případě se využívá Studentova (někdy také Cauchyho) rozdělení pravděpodobnosti s jedním stupněm volnosti.

$$q_{ij} = \frac{exp(-\|y_i - y_j\|^2)}{\sum_{k \neq 1} exp(-\|y_k - y_l\|^2)} \tag{17}$$

3. Posledním krokem je minimalizovat rozdíly v P a Q. Cílová funkce je vyjádřena ve tvaru Kullback-Leiblerovy odchylky (divergence) mezi sdruženou distribuční funkcí P, pro vysokodimenzionální prostor, a sdruženou distribuční funkcí Q, pro nízkodimenzionální prostor.

$$CF = KL(P\|Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{18}$$

   Minimalizace cílové funkce CF se provádí algoritmem gradient descent.

$$\frac{\partial CF}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \tag{19}$$

$$\gamma^{(t)} = \gamma^{(t-1)} + \mu \frac{\partial CF}{\partial \gamma} + \alpha(t) \left(\gamma^{(t-1)} - \gamma^{(t-2)} \right) \tag{20}$$

$\gamma^{(t)}$ je řešení v iteraci t, μ je koeficient učení (angl. learning rate), $\alpha(t)$ reprezentuje spád v iteraci t [37].

Efektivitu a přesnost výpočtu ovlivňuje výrazně předpříprava dat. Ačkoli je algoritmus t-SNE výrazně robustnější vůči odlehlým hodnotám než předchozí metody SNE (popsáno v roce 2002 Hintonem a Roweisem [38]) díky normalizaci hodnot Gaussova rozdělení pro jednotlivé body, je potřeba dbát na normalizaci hodnot a odstranění odlehlých hodnot. Dále je nutné vhodně zvolit následující parametry:

- perplexita
- μ

Uvedené parametry výrazně ovlivňují průběh výpočtu i výsledek, pro jejich správný výběr je potřeba výborná znalost vstupních dat, veličin a procesů, které představují. V tomto ohledu jsou jednodušší metody, jako například PCA popsaná v odstavci 4.1.1, robustnější.

Analýza je provedena použitím třídy `sklearn.manifold.TSNE` z knihovny `scikit learn`. Na rozdíl od metody k-Means umožňuje algoritmus t-SNE redukci dat do menšího počtu dimenzí, a tudíž i přímou vizualizaci výsledků. Toho bylo využito v první variantě této analýzy. Druhým zkoumaným postupem bylo nejprve zredukovat dataset do 3 dimenzí pomocí metody PCA a následně provést analýzu t-SNE s další redukcí do 2 dimenzí. Účelem bylo prozkoumat vliv postupné redukce dimenzí na výsledky analýzy.

1. Algoritmus t-SNE je aplikován přímo na z-scorovaný dataset, výstupní matice $X_{t-SNE} = (N \times 2)$ je následně zobrazena bodovým diagramem a jednotlivé body jsou vyneseny v barvě statusu stanoviště, stejně jako v předchozích případech.

2. Data jsou nejprve metodou PCA zredukována do použitím 3 hlavních komponent do 3 dimenzí, $X(N \times 4) \longrightarrow X_{PCA}(N \times 3)$ a následně jsou data zpracována metodou t-SNE a výstupní matice $X_{t-SNE} = (N \times 2)$ je zobrazena bodovým diagramem.

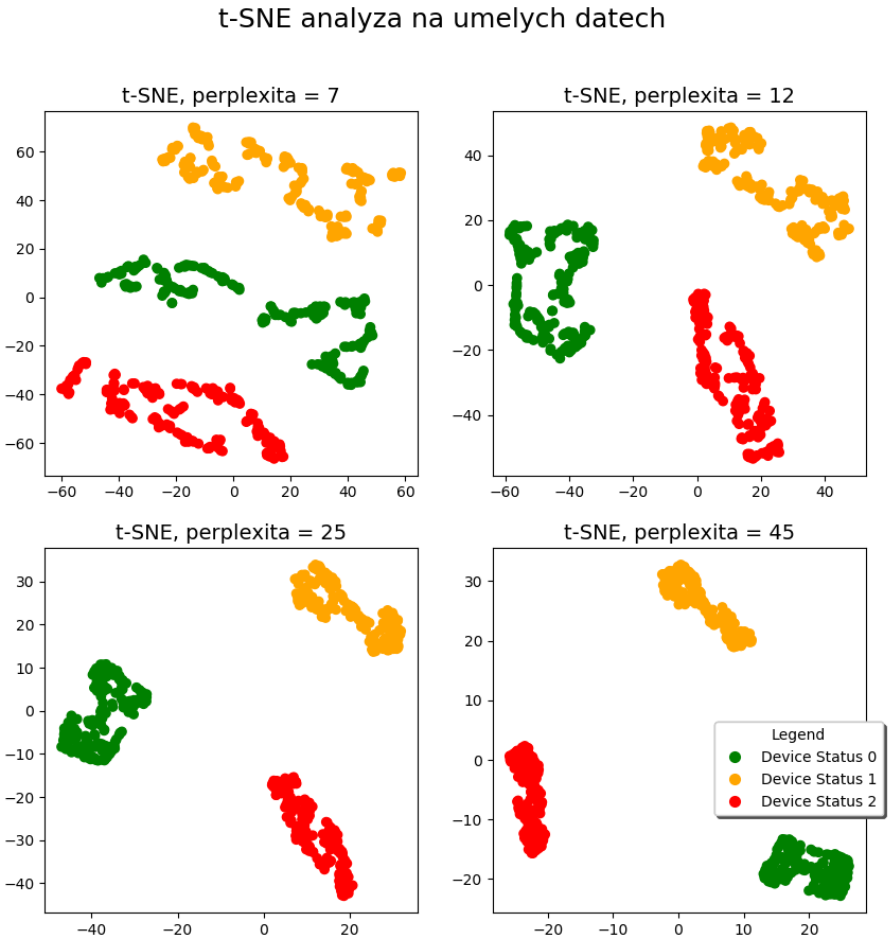
Pro výpočet byla použita třída `tSNE` opět z knihovny scikit-learn:

```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2,          # model pro 2 hlavni komponenty
            init='random',           # nahodna inicializace vah
            random_state=5,          # generator nah. cisel
            perplexity=perplexity,   # perplexita
            early_exaggeration=15,   # pomerna vzdalenost clusteru
            learning_rate=500)       # koeficient uceni
tsneX = tsne.fit_transform(pcaX)     # fitovani modelu a transformace
```

Parametry použitého algoritmu t-SNE jsou vypsány v tab. 6, zbytek parametrů byl ponechán v defaultním nastavení.

t-SNE		
počet hlavních komponent	`n _ components`	2
inicializace vah	`init`	náhodná
perplexita	`perplexity`	viz obrázky
koeficient učení μ	`learning _ rate`	500
poměrná vzdálenost shluků	`early _ exaggeration`	15
PCA + t-SNE		
počet hlavních komponent	`n _ components`	2
inicializace vah	`init`	náhodná
perplexita	`perplexity`	viz obrázky
koeficient učení μ	`learning _ rate`	600
poměrná vzdálenost shluků	`early _ exaggeration`	15

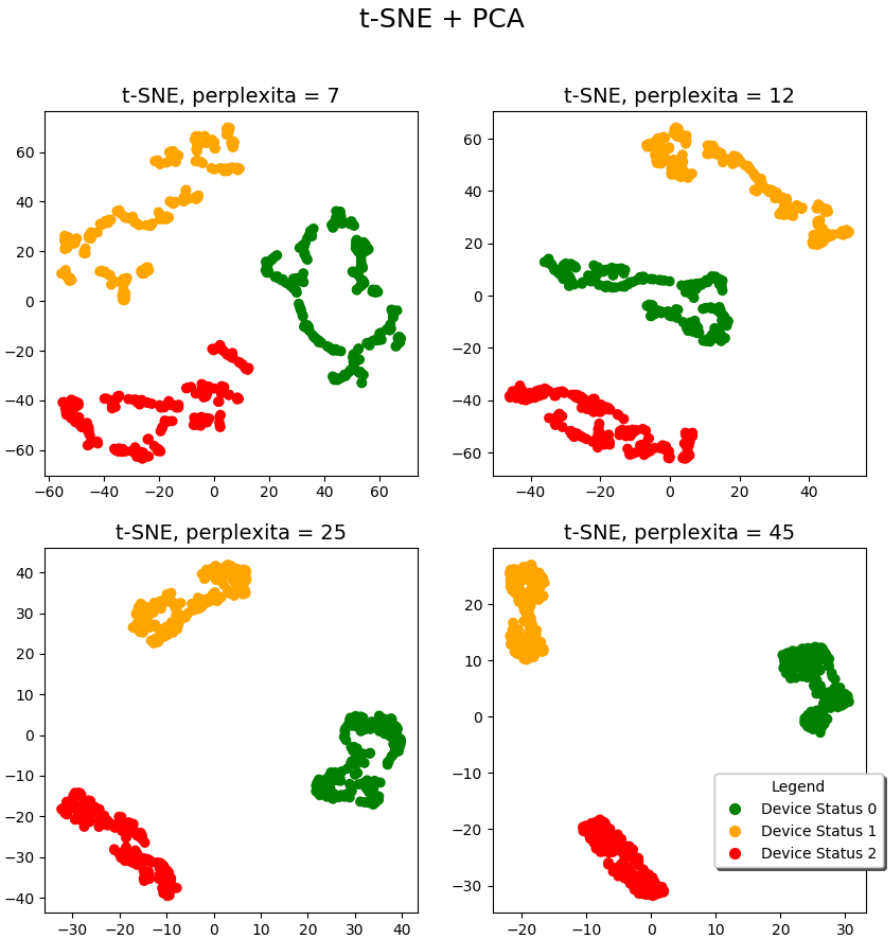
Tabulka 6: Parametry algoritmu t-SNE



Obrázek 13: Analýza t-SNE na umělých datech - při všech hodnotách perplexity bylo dosaženo vytvoření oddělených konzistentních clusterů

Na obr. 13 je zobrazena analýza t-SNE popsaná v bodě 1, parametry výpočtu jsou shrnuty v tab. 6. Na obrázku je zřejmé, že perplexita výrazně ovlivňuje zobrazení jednotlivých shluků, pro vyšší perplexitu se shluky jeví více kompaktní a více vzdálené od sebe, při nízké perplexitě jsou clustery rozprostřené po celém prostoru, pro všechny zobrazené perplexity se jeví shluky přibližně stejně velké.

Na obr. 14 je zobrazena analýza popsaná v bodě 2, parametry jsou shrnuty v tab. 6. Stejně jako v předchozím případě lze konstatovat, že shluky jsou pro všechny velikosti perplexity dobře oddělené, konzistentní a přibližně stejně velké. Použití metody s postupnou redukcí dimenzí nemá na výsledek žádný výrazný vliv.



Obrázek 14: Analýza PCA a t-SNE na umělých datech - postupnou redukcí dimenzí bylo dosaženo přibližně stejného výsledku.

4.1.5 Samoorganizační mapy

Samoorganizační mapy (angl. Self-Organizing Maps) jsou nesupervizorované neuronové sítě používané k dimenzionální redukci vstupních dat a jejich vizualizaci, dalším běžným využitím je clustering a klasifikace dat. SOM jsou schopné převést komplexní statistické vztahy mezi vysokodimenzionálními daty do jednoduchých geometrických zobrazení (map). Samoorganizační mapy se také nazývají Kohonenovy, podle svého tvůrce, finského profesora Tuevo Kohonena, který tento algoritmus představil v 80. letech 20. století.

Princip metody je založen na kompetitivním učení. Vstupní data se „naladí“ na různé výstupy a za předpokladu, že SOM je dvourozměrná, lze pak výstupy zobrazit na dvourozměrné mapě odrážející shluky obsahující vstupní data se společnými znaky.

Výpočet probíhá následujícím způsobem [39] [40]:

1. Inicializace vektoru vah $\vec{w_{ij}}$ v uzlech mapy.

2. Výběr vstupního vektoru $\vec{x}$, obvykle náhodný.

3. Pro všechny uzly mapy se provedou kroky 3a a 3b.

   (a) Výpočet Euklidovské vzdálenosti mezi vstupním vektorem $\vec{x}$ a váhovým vektorem $\vec{w_{ij}}$ v uzlových bodech mapy $\|\vec{x}(k) - \vec{w_{ij}}(k)\|$.

   (b) Výběr BMU (angl. best matching unit), tedy uzel s nejmenší Euklidovskou vzdáleností od vstupního vektoru.

$$d = min\left(\|\vec{x}(k) - \vec{w_{ij}}(k)\|\right) \tag{21}$$

4. Výpočet topologického okolí BMU $\beta_{ij}(k)$ a jeho poloměru $\sigma(k)$ a dále aktuální hodnoty koeficientu učení, zde $\alpha(k)$ (angl. learning rate), který se v průběhu algoritmu zmenšuje, k je současná iterace.

$$\beta_{ij}(k) = exp\left(\frac{-d^2}{2\sigma^2(k)}\right), \quad k = 1, 2, \ldots n \tag{22}$$

$$\sigma(k) = \sigma_0 \cdot exp\left(\frac{-k}{\lambda}\right), \quad k = 1, 2, \ldots n \tag{23}$$

$$\alpha(k) = \alpha_0 \cdot exp\left(\frac{-k}{\lambda}\right), \quad k = 1, 2, \ldots n \tag{24}$$

5. Výpočet nového váhového vektoru $\vec{w_{ij}}(k+1)$.

$$\vec{w_{ij}}(k+1) = \vec{w_{ij}}(k) + \alpha(k)\beta_{ij}(k)\left[\vec{x}(k) - \vec{w_{ij}}(k)\right] \tag{25}$$

6. Kroky 2 až 5 se opakují, dokud $k = n$, kde k je aktuální iterace a n je celkový počet iterací.

Podobně jako v případě t-SNE je výsledek silně závislý na zvolených parametrech a nastavení SOM. Jako první je potřeba zvážit velikost sítě. Giuseppe Vettigli, autor minimalistické NumPy implementace SOM pro Python [41], navrhuje vztah:

$$n_{neuron} \geq 5 \cdot \sqrt{N}, \tag{26}$$

kde n_{neuron} je celkový počet neuronů v síti a N je celkový počet vzorků. Například pro 150 analyzovaných vzorků, tedy $n_{neuron} = 5 \cdot \sqrt{160} = 63,25$. Pro analýzu 160 vzorků by měla postačovat síť o velikosti $8x8$.

Po stanovení velkosti sítě se provádí inicializace vektoru vah. V původním algoritmu navrhoval Tuevo Kohonen náhodnou inicializaci vah [39]. Pro správný průběh je vhodné velikost počátečního vektoru vah normalizovat. Dalším přístupem, který se rozvíjí zejména v posledních letech je vybrat vektor vah z prostoru hlavní komponenty. Tento přístup je vhodný pro lineárně závislá data, která mohou být dobře reprezentována hlavní komponentou. Pokud jde o data vysoce nelineární, je vhodnější použít náhodou inicializaci. [39][41]

Z popisu algoritmu, viz. výše, je zřejmé, že na výsledek a zejména na proces učení mají velký vliv také koeficienty α_0 a σ_0. Koeficient učení α_0 se pohybuje v intervalu $[0, 1]$, pokud má α_0 hodnotu vyšší, např. $0,8$ probíhá učení rychleji, změna vah v jednotlivých iteracích má vyšší hodnotu, ale je problematické dosáhnout přesného výsledku. Proto se z této počáteční vyšší hodnoty v průběhu výpočtu koeficient α_0 snižuje. Analogicky se postupuje pro koeficient σ_0, σ_0 je poloměr sousední funkce topologického okolí β_{ij} a určuje 2D vzdálenost v SOM mapě, kde se při výpočtu nového vektoru vah zkoumají sousední uzly. [39][40]

Pro analýzu byl použit algoritmus miniSOM, minimalistická implementace samoorganizačních map s využitím knihovny NumPy od Giuseppe Vettigliho [41]:

```
class MiniSom(self, x, y, input_len, sigma=1.0, learning_rate=0.5,
              decay_function=asymptotic_decay,
              neighborhood_function='gaussian', topology='rectangular',
              activation_distance='euclidean', random_seed=None).
```

Použití je následující:

1. Převod vstupních dat do formátu `NumPy.array`.
2. Import třídy MiniSom z knihovny minisom.
3. Inicializace, např. `som = MiniSom(parametry)`.
4. Inicializace vah, např. `som.random_weights_init(data)` pro inicializaci vah náhodným vektorem.
5. Trénování mapy, např. `som.train_batch(data, epoch)`, pro trénování celého datasetu v původním pořadí řádků.
6. Zobrazení jednotlivých bodů v mapě.

Pro výsledné zobrazení dat byla nejprve vynesena matice U. Matice U (angl. U-Matrix - Unified Distance Matrix) se používá jako reprezentace samoorganizačních map. Je to matice, která obsahuje normalizované euklidovské vzdálenosti mezi váhovými vektory sousedících neuronů.

$$U_n = \sum_{m=0}^{m_{max}} \left(d(w_n, w_m)\right), \tag{27}$$

kde U_n je prvek matice U pro neuron n, w_n jsou váhy neuronu n a w_m jsou váhy neuronů $1, 2, \ldots m_{max}$, které s neuronem n sousedí. Prvky matice nabývají hodnot $(0, 1)$, 0 pokud jsou si sousední body „podobné“, tedy leží ve stejném clusteru, 1 pokud body nevykazují žádnou míru shody. Světlá místa na mapě jsou následně interpretována jako clustery, tedy oblasti se stavy se stejnými znaky. Tmavá místa na mapě značí, že váhy sousedících neuronů jsou více vzdálené a nemají společné znaky, tato místa jsou označována jako hranice clusterů.

K získání matice U v knihovně minisom slouží funkce `som.distance_map()`.

```
U_Matrix = som.distance_map(data)
print(U_Matrix.round(decimals=1))
[[0.1  0.4  0.3  0.4  0.3  ...  0.2  0.4  0.1]
 [0.2  0.4  0.5  0.5  0.6  ...  0.4  0.5  0.3]
 [0.2  0.3  0.4  0.5  0.7  ...  0.4  0.5  0.3]
 [0.1  0.2  0.3  0.4  0.7  ...  0.4  0.7  0.4]
 [..................................]
 [ 0.   0.   0.   0.   0.  ...  0.3  0.2  0.2]]
```

Dále byly pomocí funkce `som.winner(point)` vyneseny jednotlivé body do mapy, stejně jako v předchozích případech mají jednotlivé body barvu v závislosti na svém statusu stanoviště.

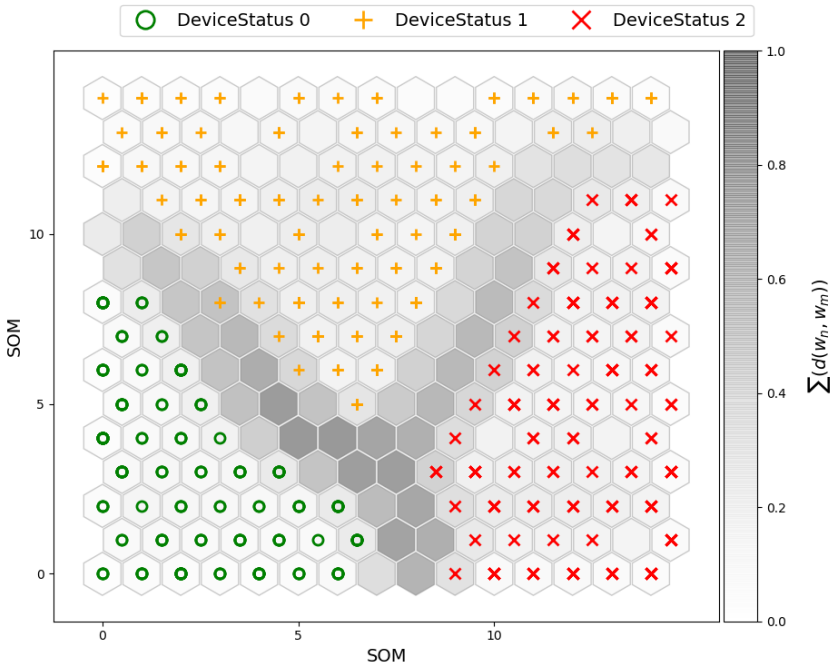
Nastavení parametrů algoritmu bylo optimalizováno pro data ze simulátoru a následně aplikováno i na umělá data s cílem porovnat výsledky. Parametry jsou uvedeny v tab. 7, neuvedené parametry byly ponechány v deafaultním nastavení.

velikost mapy	`som_size`	15×15
dimenze vstupních dat	`input_len`	4
σ_0	`sigma`	1,6
koeficient učení α_0	`learning_rate`	0,6
topologie mapy	`topology`	hexagonální
inicializace vah		náhodná
inicializace generátoru náhodných čísel	`random_seed`	10

Tabulka 7: Parametry použitého algoritmu MiniSom

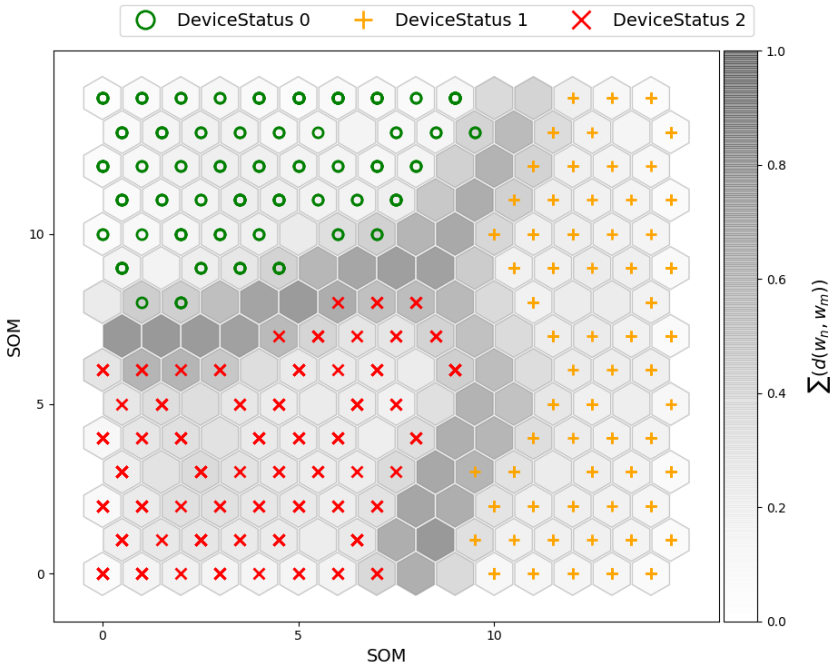
Velikost mapy (15×15) byla určena dle vztahu (26), tedy $n_{neuron} \geq 5 \cdot \sqrt{N} = 5 \cdot \sqrt{561} = 118,42$, což při použití čtvercové sítě znamená minimální velikost $10,88 \times 10,88$, metodou střelby byla následně zvolena velikost (15×15).

Na obr. 15 je zobrazen výsledek pro analýzu SOM na umělých datech s trénováním v původním pořadí řádků. Jsou zde 3 shluky zřetelně oddělené tmavými uzly. Jak již bylo zmíněno výše, barva uzlů je úměrná euklidovské vzdálenosti vektoru vah sousedních neuronů. Zelené body (status stanoviště 0) se téměř výhradně nachází ve světlých uzlech, oranžové a červené se nachází v tmavších uzlech, což značí, že i uvnitř celistvého shluku mohou mít jednotlivé body odlišné znaky. Žádný z clusterů neobsahuje body „cizí“ barvy.

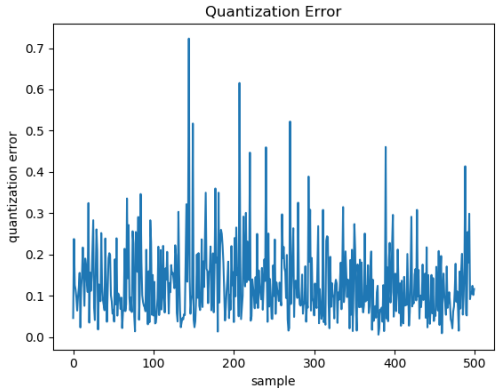


Obrázek 15: Analýza SOM na umělých datech, trénování dávkovým způsobem v původním pořadí řádků - vytvoření tří oddělených shluků

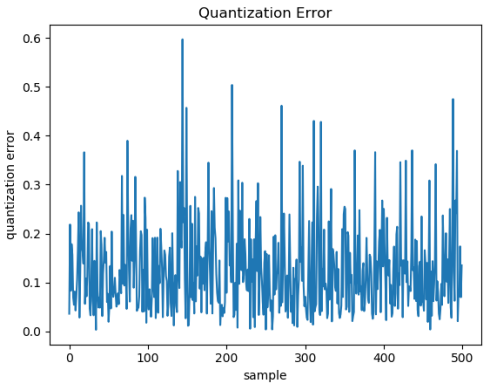
Pro porovnání je na obr. 16 uveden výsledek pro trénování s náhodným pořadím řádků vstupní matice X. Byla použita mapa o velikosti 15×15 a trénování probíhalo pro 1000 iterací, stejně jako v předchozím případě. Ačkoli se i zde vytvořili zřetelně 3 shluky oddělené uzly s tmavou barvou je patrné, že uspořádání clusterů v mapě je rozdílné.



Obrázek 16: Analýza SOM na umělých datech, trénování s náhodným pořadím řádků - vytvoření tří oddělených shluků s tmavšími barvami uvnitř clusterů něž na obr. 15



Obrázek 17: Chyba kvantizace pro učení s daným pořadím řádků



Obrázek 18: Chyba kvantizace pro učení s náhodným pořadím řádků

Na obr. 17 a 18 Je zobrazena hodnota chyby kvantizace pro oba způsoby učení. Chyba kvantizace (angl. Quantization Error) je vyjádřena jako euklidovská norma rozdílu vstupního vektoru a BMU (viz. 4.1.5). Chyba kvantizace udává míru, se kterou je možno aproximovat vstupní data

samoorganizační mapou.

$$QE = d(x_i, BMU(x_i)), \tag{28}$$

kde $x_i \in X$ je vstupní vzorek a $BMU(x_i)$ je jeho „vítězný“ neuron. $\overline{QE}$ je potom střední hodnota pro celý X v dané iteraci [42].

Na obr. 17 a 18 je zobrazena chyba kvantizace pro poslední iteraci.

Chyba kvantizace při poslední iteraci	
učení s původním pořadím řádků	$\overline{QE} = 0,1305$
učení s náhodným pořadím řádků	$\overline{QE} = 0,1270$

Tabulka 8: Porovnání chyby kvantizace pro různé druhy učení na umělých datech

V tabulce 8 je uvedena střední hodnota chyby kvantizace pro jednotlivé druhy učení.

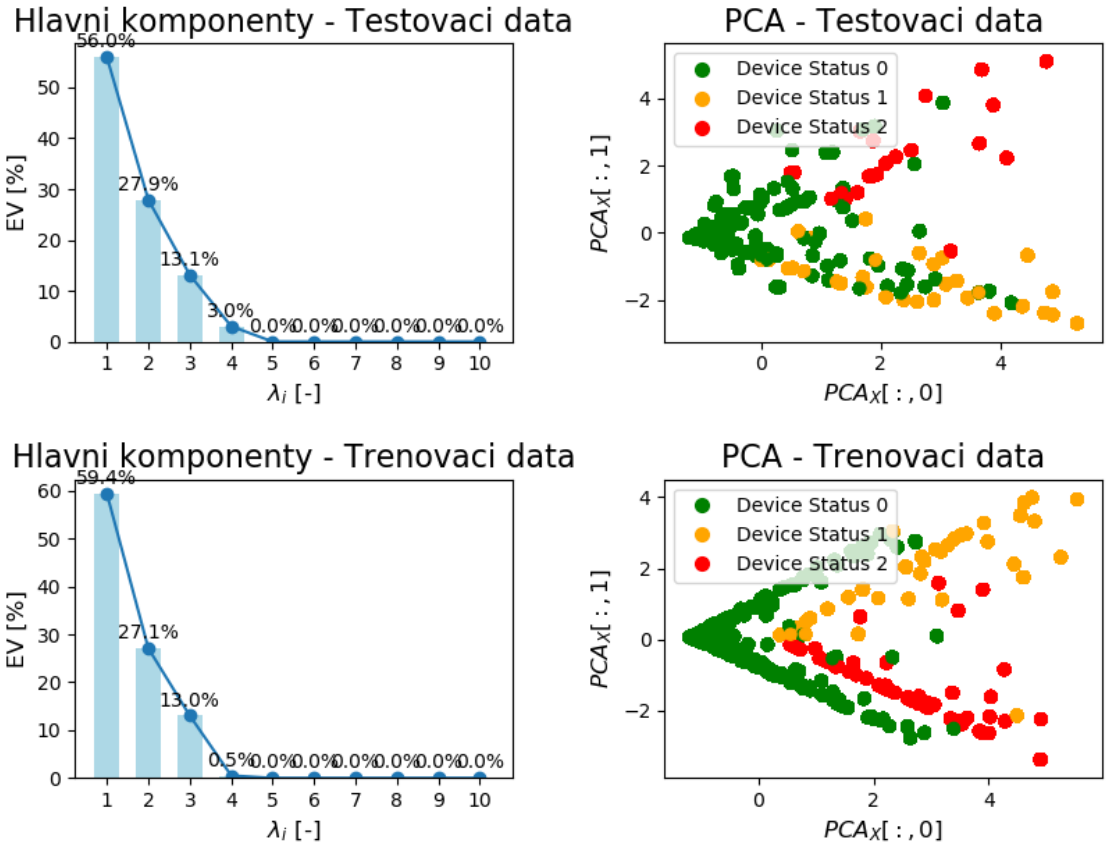
4.2 Aplikace dat z procesu

V této sekci jsou představené metody aplikovány na data z procesu (ze simulátoru výrobní linky).

4.2.1 Analýza dat z procesu metodou PCA

Nejprve byla provedena analýza metodou PCA, analogicky jako v odst. 4.1.1. Bylo určeno množszví vysvětleného rozptylu v jednotlivých komponentách a byla provedena redukce dimenzí touto metodou. Výsledky byly vyneseny do grafů.

Z obr. 19 pro simulovaná data lze vyčíst, že v prvních dvou hlavních komponentách je obsaženo asi $83,9\%$ informací o testovacím datasetu a $86,5\%$ o trénovacím, což je v obou případech méně než pro umělá data. Na obr. 19 vpravo, je zobrazena metoda PCA pro 2 hlavní komponenty. Pro testovací data (vpravo nahoře) lze sice přibližně identifikovat 3 oblasti, kde se nachází vysoká koncentrace bodů stejné barvy, ale nelze říci, že se jedná o oddělené shluky. Jiná situace je u trénovacích dat (vpravo dole), kde se vytvořily 2 oddělené shluky ve tvaru „V“, první je tvořen téměř výhradně zelenými body (tzn. status stanoviště 0), druhé V má horní část tvořenou převážně oranžovými body (tzn. status stanoviště 1), spodní část je tvořena převážně červenými body (tzn. status stanoviště 2). Lepší výsledek pro trénovací data může být dán vyšším počtem bodů (stavů) v datasetu.

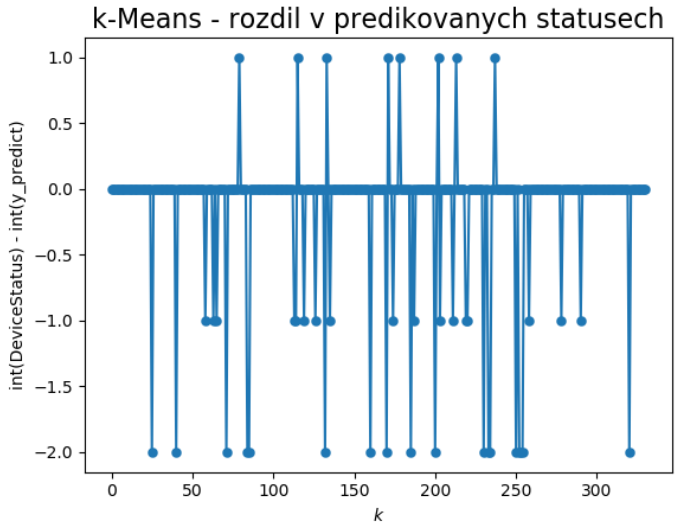


Obrázek 19: Grafické znázornění metody hlavních komponent na testovacím a trénovacím datasetu ze simulátoru, na levé straně je znázorněno množství informace obsažené v jednotlivých komponentách dle vztahu (8), v pravé části je metoda hlavních komponent pro 2 hlavní komponenty

4.2.2 Analýza a predikce stavu metodou k-Means na datech z procesu

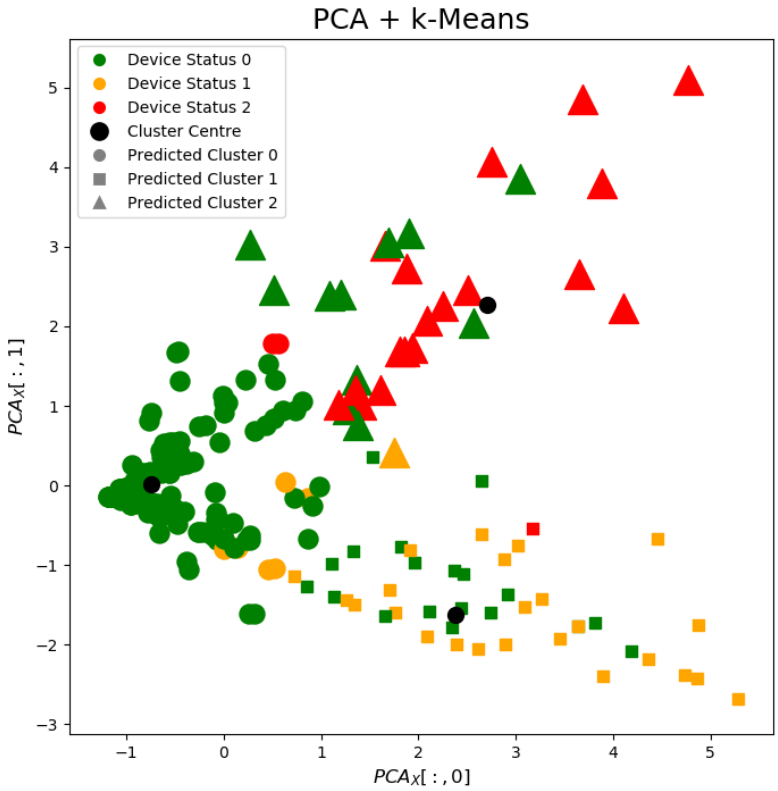
Metoda k-Means byla použita pro predikci stavu a následně v kombinaci s metodou PCA pro shlukovou analýzu.

Na obr. 20 jsou zobrazené výsledky predikce pro data z procesu. Bylo dosaženo 87% úspěšnosti, tzn. predikovaný cluster se u 43 bodů z celkových 331 neshoduje se statusem stanoviště.



Obrázek 20: Rozdíl v predikovaných a původních clusterech na datech z procesu - správná predikce 87% stavů

Na obr. 21 je provedena výše popsaná analýza k-Means. V souladu s výsledky uvedenými v obr. 20, kde se u asi 13% stavů neshodovaly clustery dle statusu stanoviště a predikce, je i zde vidět stavy, kterým algoritmus predikoval jiný status, než který byl exportován ze simulace v simulátoru dat. Je tedy možné, že ačkoli jsou průběhy veličin v simulátoru řízeny aktivním alarmem, nemusí být zcela korektně určena severita daného alarmu, respektive nastalé události. Na obr. 21 je totiž zřejmé, že statusy stavů predikované metou k-Means celkem dobře odpovídají clusterům zobrazeným metodou PCA. Tyto metody samozřejmě třídí data pouze na základě geometrické podobnosti, tudíž těmito metodami nemusí být zcela korektně podchyceny všechny reálné závislosti.



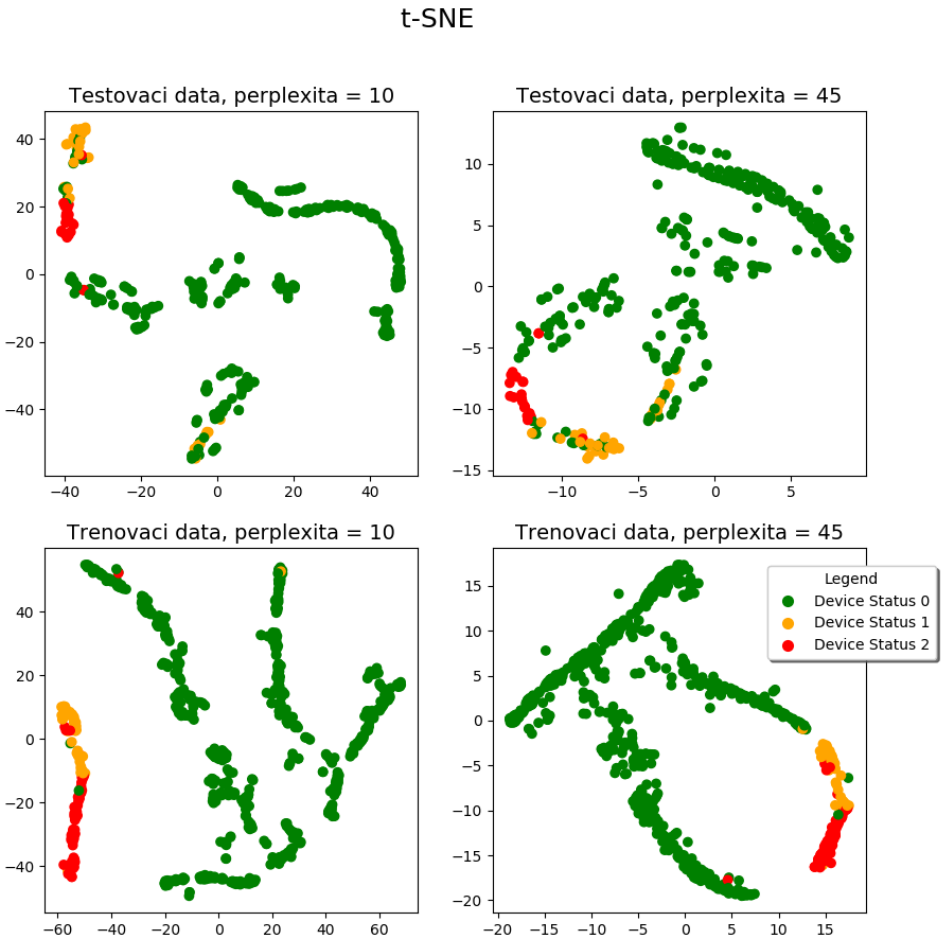
Obrázek 21: Analýza k-Means na datech ze simulátoru - dochází k nekonzistentnostem v jednotlivých shlucích, shluky nejsou od sebe oddělené

4.2.3 Analýza dat z procesu metodou t-SNE

Analýza metodou t-SNE je stejně jako v odst. 4.1.4 provedena dvěma způsoby: přímá aplikace algoritmu t-SNE na data z procesu a postupná redukce dimenzí nejprve metodou PCA a následně použití metody t-SNE pro vizualizaci.

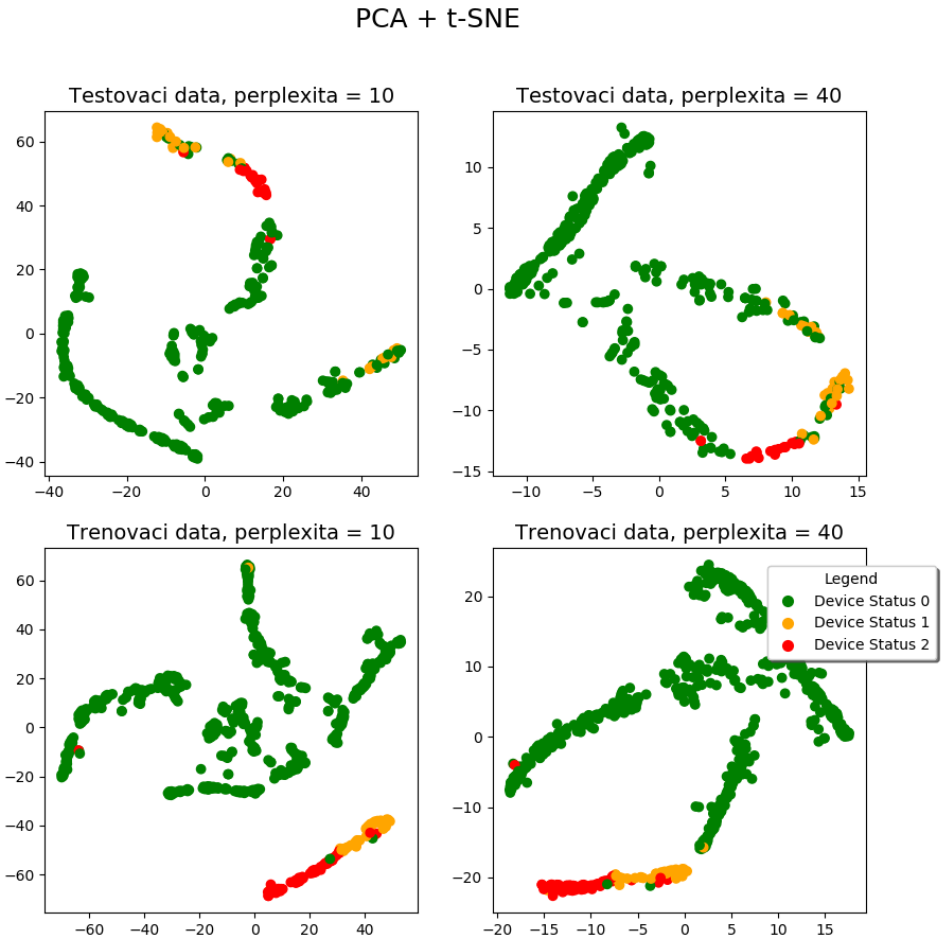
Na obr. 22 je zobrazena analýza t-SNE na datech ze simulátoru popsaná v odst. 4.1.4. Na rozdíl od výsledků pro umělá data uvedené na obr. 13 a 14 se nevytvořily konzistentní oddělené shluky bodů. V analýze testovacích dat uvedených na obr. 22 nahoře bylo pro perplexitu 10 dosaženo vytvoření dvou kompaktních oddělených shluků pro statusy stanoviště 1 a 2, ovšem tyto shluky obsahují i body se statusem 0 (zelená barva). Pro trénovací data se statusy 1 a 2 nachází ve dvou částech stejného shluku (opět připomíná tvar „V“). Tento cluster se nepodařilo rozdělit ani s použitím vyšších hodnot perplexity.

Obr. 23 znázorňuje analýzu dat z procesu postupnou redukcí dimenze (tedy nejprve redukce do 3D pomocí PCA a následně aplikace t-SNE na takto připravený dataset). Výsledky jsou podobné předchozímu případu. Pro testovací data nedošlo k výraznému zlepšení výsledků, shluky se jeví



Obrázek 22: Analýza t-SNE na datech z procesu - na testovacích datech bylo dosaženo lepšího oddělení jednotlivých clusterů, na trénovacích datech lepší konzistentnosti clusterů

sice celistvější, ale zároveň bylo více bodů se statusem 1 a 2 zařazeno do zelených shluků (status 0). Totéž platí i pro trénovací data, červenooranžový cluster se nepodařilo rozdělit, místo písmene „V“ jsou body seřazeny přibližně v přímce.



Obrázek 23: Analýza PCA a t-SNE na datech z procesu- postupnou redukcí dimenzí bylo dosaženo „těsnějšího“ uspořádání clusterů, shluky se ale nepodařilo více osamostatnit, ani nebyl odstraněn výskyt stavů s různými statusy ve stejném clusteru

4.2.4 SOM na datech z procesu

Pro data ze simulátoru byla provedena analýza odlišným způsobem než u umělých dat. Nejprve byla neuronová síť natrénována pomocí trénovacích dat, váhy $\vec{w}$ byly následně použity jako inicializační váhy pro testovací data. Zároveň byly kromě statusů stanoviště pro daný stav použity do výsledků také aktivní alarmy.

Zobrazení výsledků proběhlo stejně jako v modelovém případě pro umělá data, tedy zobrazení matice U, pro zobrazení clusterů a jejich hranic. Dále byly do této matice zobrazeny stavy jako body s barvou a tvarem značky odpovídající jejich statusu. V posledním kroku byly v mapě k bodům přidány popisky s alarmy. K tomu byla použita metoda knihovny Matplotlib:

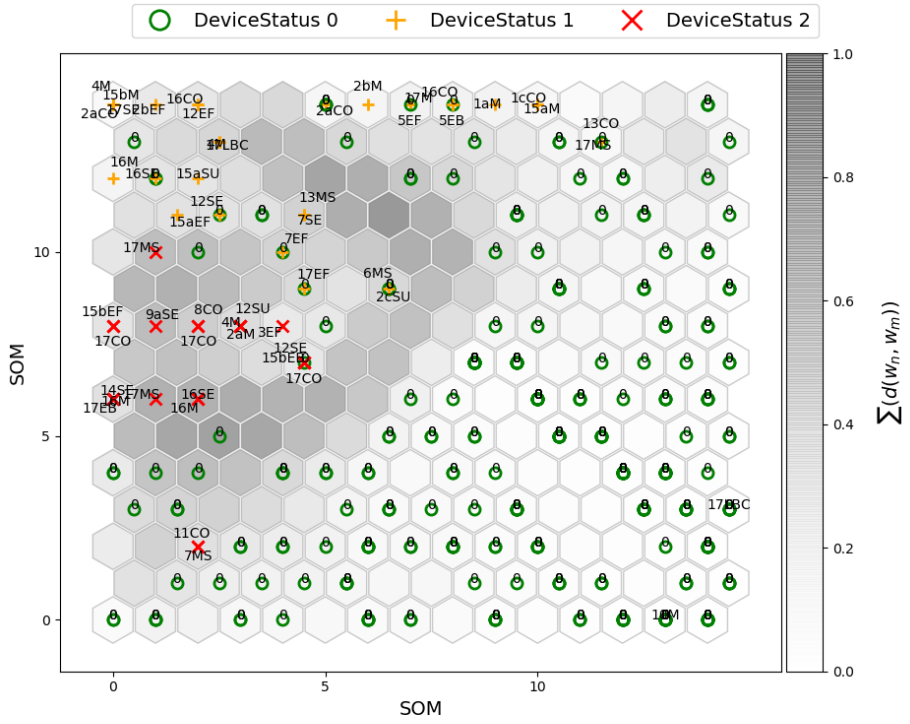
```
import matplotlib.pyplot as plt

plt.annotate(label,                          # popisek
             (x,y),                          # souradnice pro zobrazeni popisku
             textcoords="offset points",     # pozice popisku
             xytext=(0,10),                  # vzdalenost od souradnice
             ha='center')                    # centrovani
```

Alarmy jsou popsány v odst. 3.1. Jak je uvedeno v tomto odstavci, kompletní informace o alarmu je např. `15a MAINTENANCE`, což znamená, že na stanici 15a probíhá údržba. Za účelem jednoduššího zobrazení byly pro jednotlivé druhy alarmů zavedeny zkratky, ty jsou uvedeny v tabulce 9.

Původní alarm	zkratka
MAINTENANCE	M
Low buffer count	LBC
Empty buffer	EB
MATERIAL STUCK	MS
Casing open	CO
Setup	SE
EQUIPMENT FAULT	EF
Sensor error	SE

Tabulka 9: Seznam zkratek alarmů použitých ve výstupu SOM

Předpoklad je takový, že se vytvoří ideálně 3 clustery, jako v předchozím případě, oddělené tmavými uzly mapy. Dále se předpokládá, že se ve stejných uzlech mapy budou vyskytovat body se stejným statusem a s podobným alarmem.

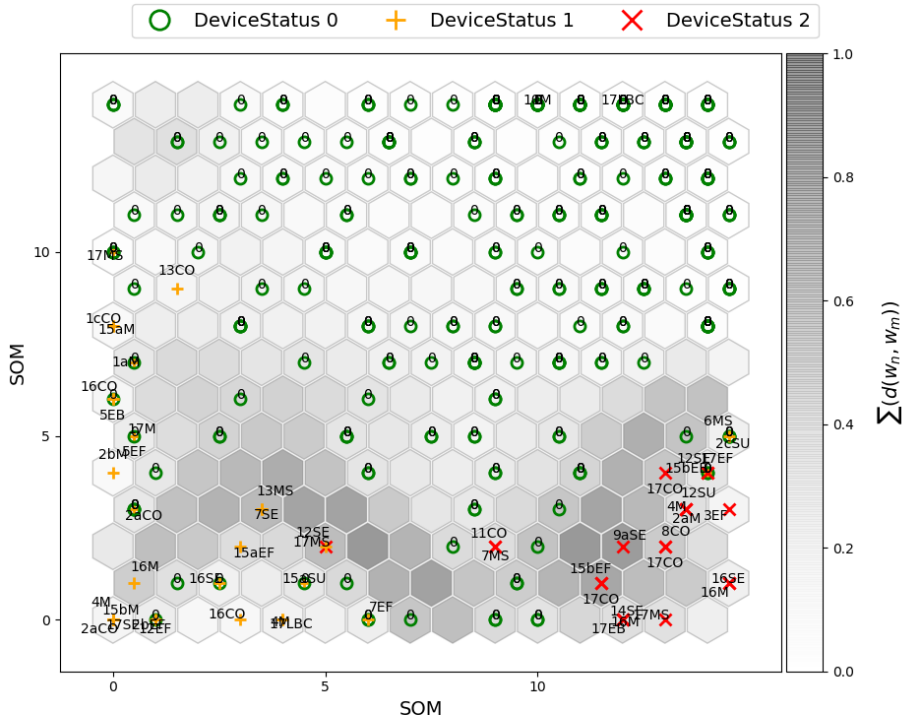


Obrázek 24: Výsledek SOM pro simulovaná data s použitím učení v původním pořadí řádků - tvorba 3 špatně oddělených clusterů, výskyt stavů v cizích clusterech

Na obr. 24 a 25 jsou výsledky pro trénování s původním, respektive náhodným pořadím řádků. Stejně jako v případě umělých dat jsou výsledky rozdílné.

V obou případech lze rozeznat 3 shluky, na obr. 24 lze identifikovat cluster pro status 1 (oranžová) asi na první třetině diagonály matice. Pod ním zhruba v polovině mapy je cluster pro status 2. Zbytek mapy tvoří cluster statusu 0. Stejně jako při použití předchozích metod se však v jednotlivých clusterech vyskytují i body z jiných shluků.

Na obr. 25 pro trénování v náhodném pořadí řádků jsou clustery výrazně lépe rozlišitelné, cluster statusu 1 se nachází v levé spodní části, cluster statusu 2 v pravé dolní části, zbytek mapy tvoří cluster statusu 0. Na rozdíl od trénování v původní posloupnosti řádků jsou shluky výrazně lépe oddělené (hranice clusteru má výrazně tmavší barvu než jeho vnitřek).



Obrázek 25: Výsledek SOM pro simulovaná data s použitím učení v náhodném pořadí řádků - tvorba 3 oddělených clusterů, výskyt stavů v cizích clusterech

Chyba kvantizace při poslední iteraci		
metoda učení	trénovací dataset	testovací dataset
učení s původním pořadím řádků	$\overline{QE} = 0,1142$	$\overline{QE} = 0,1699$
učení s náhodným pořadím řádků	$\overline{QE} = 0,1225$	$\overline{QE} = 0,1791$

Tabulka 10: Porovnání chyby kvantizace pro různé druhy učení, data z procesu

Tab. 10 ukazuje střední chybu kvantizace při poslední iteraci. Zajímavé je, že ačkoli z pohledu tvorby oddělených clusterů byly lepších výsledků dosaženo při použití učení s náhodným pořadím řádků, má tato metoda vyšší střední chybu kvantizace, tedy horší shodu mezi původními daty a samoorganizační mapou.

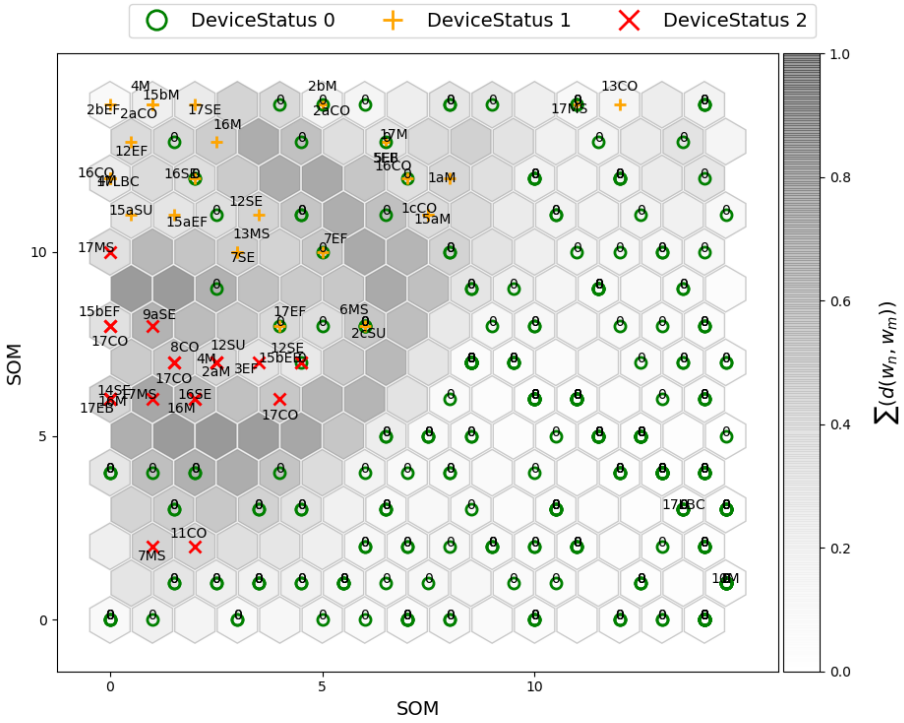
4.2.5 SOM z rozšířeného stavu

Předchozí varianta byla provedena na vstupních datech ve formátu uvedeném ve vztahu (1).

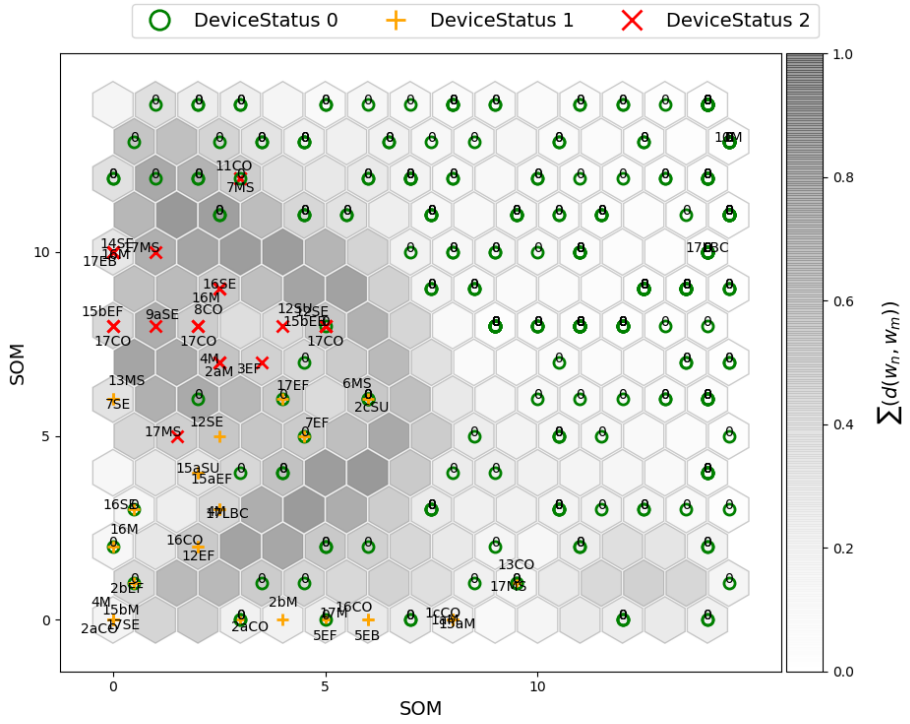
Další myšlenka byla taková, že pokud do stavu budou přidány hodnoty na předchozích stanovištích, měly by se rozdíly v jednotlivých stavech zvýšit a tím pádem by mělo být možné jednotlivé statusy lépe identifikovat. Vstupní matice vypadá následovně:

$$X_{rozsireny} = \begin{matrix} \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{timeRun16} & \text{timeWait16} & \text{timeStop16} & \text{Speed16} & \text{timeRun17} & \text{timeWait17} & \text{timeStop17} & \text{Speed17} \end{matrix} \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} & x_{1,6} & x_{1,7} & x_{1,8} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} & x_{2,6} & x_{2,7} & x_{2,8} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N,1} & x_{N,2} & x_{N,3} & x_{N,4} & x_{N,5} & x_{N,6} & x_{N,7} & x_{N,8} \end{bmatrix} \begin{matrix} \text{k=1} \\ \text{k=2} \\ \vdots \\ \text{k=N} \end{matrix} \tag{29}$$

Algoritmus zůstává stejný jako v předchozím případě.



Obrázek 26: SOM pro simulovaná data s rozšířeným stavem, učení v původním pořadí řádků - oddělení zeleného clusteru, hranice mezi oranžovým a červeným clusterem není výrazná



Obrázek 27: Som pro simulovaná data s rozšířeným stavem, učení v náhodném pořadí řádků - podobný výsledek jako na obr. 26

Na obr. 26 a 27 jsou zobrazeny výsledky.

Pro variantu učení s původním pořadím řádků, obr. 26 se podařilo více oddělit cluster statusu 0, hranice tohoto clusteru je tvořena tmavými uzly. Body, které se nachází v „cizím“ shluku, se nepodařilo přemístit.

Učení s náhodným pořadím řádků, obr. 27, nepřineslo zlepšení. Naopak, hranice mezi shluky se stavy 1 a 2 se stala méně výraznou.

Chyba kvantizace při poslední iteraci		
metoda učení	trénovací dataset	testovací dataset
učení s původním pořadím řádků	$\overline{QE} = 0,1161$	$\overline{QE} = 0,1780$
učení s náhodným pořadím řádků	$\overline{QE} = 0,1232$	$\overline{QE} = 0,1712$

Tabulka 11: Porovnání chyby kvantizace pro různé druhy učení, simulovaná data s rozšířeným stavem $X_{rozsireny}$

V tab. 11 jsou vypsány střední chyby kvantizace pro poslední iteraci. Oproti vstupním datům X jsou rozdíly tohoto ukazatele pro vstup $X_{rozsireny}$ mezi metodami menší rozdíly.

Lze konstatovat, že výsledky pro původní vstup X dosahují lepšího rozlišení jednotlivých shluků a stav je tedy s největší pravděpodobností určen pouze aktuálními hodnotami veličin na daném stanovišti.

4.2.6 Analýza alarmů v samoorganizační mapě

Pro analýzu byl vybrán výstup z učení s náhodným pořadím řádků a vstupními daty X, obr. 25.

Alarm	Četnost výskytu
`17 Casing Open`	3
`17 MATERIAL STUCK`	1
`17 Empty buffer`	1
`17 EQUIPMENT FAULT`	1
`16 MAINTENANCE`	2
`16 Sensor error`	1
`15b EQUIPMENT FAULT`	2
`14 Sensor error`	1
`12 Sensor error`	1
`12 Setup`	1
`9a Sensor error`	1
`8 Casing open`	1
`6 MATERIAL STUCK`	1
`4 MAINTENANCE`	1
`3 EQUIPMENT FAULT`	1
`2c Setup`	1
`2a MAINTENANCE`	1

Tabulka 12: Seznam alarmů nacházejících se v clusteru statusu 2

V tabulce 12 jsou uvedeny alarmy, které se vyskytují v uzavřeném clusteru se stavy s převažujícím statusem 2. Obecně lze říci, že se v clusteru vyskytují hlavně alarmy, které způsobí zastavení linky a vyskytují se hlavně v zadní části linky. Nejvyšší četnost je pro alarm `17 Casing open`, tedy otevřený kryt na stanovišti 17. Pro tyto stavy platí, že je snížený čas běhu `timeRun`, vysoký čas čekání (prostoje) `time Wait`, pohybuje se mezi 20 a 30 s a čas `timeStop` a rychlost jsou nulové. Obecně lze říci, že společným jmenovatelem většiny těchto stavů je nulová rychlost.

Pro stavy nacházející se v oranžovém shluku s převažujícími statusy 1 bychom mohli jmenovat společný znak: nulový `timeWait`. Neplatí to ovšem stejně jako v zeleném clusteru pro všechny stavy.

Alarm	Četnost výskytu
17 Sensor error	1
17 MATERIAL STUCK	1
17 Low buffer count	1
16 MAINTENANCE	1
16 Sensor error	1
16 Casing open	1
15b MAINTENANCE	1
15a Setup	1
15a EQUIPMENT FAULT	1
13 MATERIAL STUCK	1
12 EQUIPMENT FAULT	1
12 Sensor error	1
7 Sensor error	1
7 EQUIPMENT FAULT	1
4 MAINTENANCE	1
2a Casing open	1

Tabulka 13: Seznam alarmů nacházejících se v clusteru statusu 1

Alarm	Četnost výskytu
17 MAINTENANCE	1
17 MATERIAL STUCK	1
16 Casing open	1
15a MAINTENANCE	1
13 Casing open	1
5 Empty buffer	1
5 EQUIPMENT FAULT	1
2b MAINTENANCE	1
2a Casing open	1
1c Casing open	1
1a MAINTENANCE	1

Tabulka 14: Seznam alarmů se severitou 1 nacházející se mimo cluster

Pro stavy se statusem 1 nacházející se mimo svůj cluster lze zobecnit, že stavy mají: snížený `timeRun`, nulový `timeWait`, zvýšený `timeStop` a rychlost pohybující se v druhé polovině rozmezí.

Alarm	Četnost výskytu
`11 Casing open`	1
`7 MATERIAL STUCK`	1

Tabulka 15: Alarmy se severitou 2 nacházející se mimo cluster

5 Návrh modulu s využitím shlukové analýzy pro prediktivní údržbu

Výše uvedené metody shlukových analýz mohou být využity v modulu pro prediktivní údržbu. Uvažujeme-li, že data z reálného provozu budou mít podobnou strukturu jako data ze simulátoru firmy mySCADA s.r.o., lze navrhnout jejich potenciální využití.

5.1 Popis navržených algoritmů

Kromě dat z výrobní linky, které jsou získávány v tomto simulátoru, by bylo potřeba také vést záznamy o údržbářských pracích na jednotlivých zařízeních. Součástí těchto záznamů by měl být druh zásahu a perioda od posledního takového zásahu. Tyto aktivity by mohli být podobně jako statusy stanoviště rozčleněny do 3 kategorií dle závažnosti, např. 0 - preventivní převážně vizuální kontrola, 1 - preventivní zásah (výměna oleje, mazání, čištění), 2 - zásah pro odstranění akutní závady (opotřebená ložiska, závažné mechanické závady, apod.). K těmto činnostem by bylo vhodné přiřadit i jejich ekonomickou a časovou náročnost, jelikož to jsou hlavní parametry, které je potřeba z hlediska výrobního procesu optimalizovat.

Data pro vyhodnocování by potom mohla mít například následující strukturu:

$$X_{all}^{T} = \begin{array}{c} \begin{matrix} \text{k=1} & \text{k=2} & \dots & \text{k=N} \end{matrix} \\ \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,N} \\ x_{2,1} & x_{2,2} & \dots & x_{2,N} \\ x_{3,1} & x_{3,2} & \dots & x_{3,N} \\ x_{4,1} & x_{4,2} & \dots & x_{4,N} \\ x_{5,1} & x_{5,2} & \dots & x_{5,N} \\ x_{6,1} & x_{6,2} & \dots & x_{6,N} \\ x_{7,1} & x_{7,2} & \dots & x_{7,N} \\ x_{8,1} & x_{8,2} & \dots & x_{8,N} \\ x_{9,1} & x_{9,2} & \dots & x_{9,N} \end{bmatrix} \end{array} \begin{array}{l} \text{timeRun} \\ \text{timeWait} \\ \text{timeStop} \\ \text{Speed} \\ \text{DeviceStaus} \\ \text{Alarm} \\ \text{MaintenanceAction} \\ \text{Cost} \\ \text{MaintenancePeriod} \end{array} \tag{30}$$

V dalším kroku by bylo nutné zjistit, které veličiny je vhodné volit jako vstupní a které jako výstupní. V této práci se původní data rozdělila již na počátku na vstupní a výstupní, jak je uvedeno ve vztazích (1) a (2). Analogicky je potřeba rozdělit data o údržbářských akcích na vstupní a výstupní. Stejně tak jako pro status stanoviště a alarmy budou vstupními parametry časy a

rychlost, dále také doba trvání akce údržby a náklady. Výstupním parametrem by tedy mohl být druh údržbářského zásahu. Vstupní a výstupní matice jsou uvedeny ve vztazích (31) a (32).

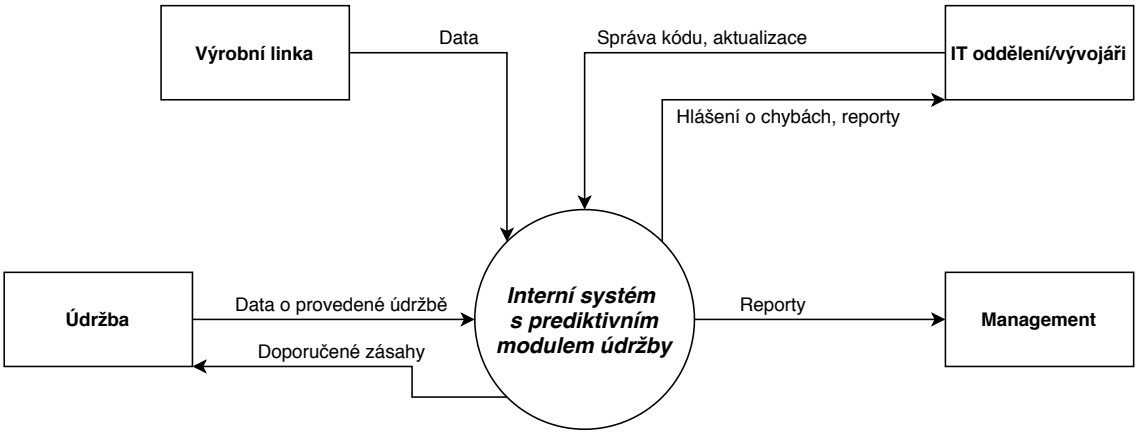
$$X^T = \begin{array}{c} \begin{matrix} \text{k=1} & \text{k=2} & \dots & \text{k=N} \end{matrix} \\ \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,N} \\ x_{2,1} & x_{2,2} & \dots & x_{2,N} \\ x_{3,1} & x_{3,2} & \dots & x_{3,N} \\ x_{4,1} & x_{4,2} & \dots & x_{4,N} \\ x_{8,1} & x_{8,2} & \dots & x_{8,N} \\ x_{9,1} & x_{9,2} & \dots & x_{9,N} \end{bmatrix} \end{array} \begin{array}{l} \\ \text{timeRun} \\ \text{timeWait} \\ \text{timeStop} \\ \text{Speed} \\ \text{Cost} \\ \text{MaintenancePeriod} \end{array} \tag{31}$$

$$y^T = \begin{array}{c} \begin{matrix} \text{k=1} & \text{k=2} & \dots & \text{k=N} \end{matrix} \\ \begin{bmatrix} x_{5,1} & x_{5,2} & \dots & x_{5,N} \\ x_{6,1} & x_{6,2} & \dots & x_{6,N} \\ x_{7,1} & x_{7,2} & \dots & x_{7,N} \end{bmatrix} \end{array} \begin{array}{l} \\ \text{DeviceStaus} \\ \text{Alarm} \\ \text{MaintenanceAction} \end{array} \tag{32}$$

Tyto stavy by byly použity nejprve pro prostou vizualizaci aktuálního stavu linky a ukládány společně s informacemi o receptu a stavu výrobků na výstupu pro možnost zpětné analýzy potenciálních problémů ve výrobě, nebo kvalitě produktů.

Zároveň by měla být takto předzpracovaná data odesílána v reálném čase do modulu pro prediktivní údržbu, kde by měl být pomocí samoorganizační mapy a dalších případných nástrojů prozkoumán stav linky a pomocí rozhodovacího algoritmu odeslat informaci o potřebě zásahu.

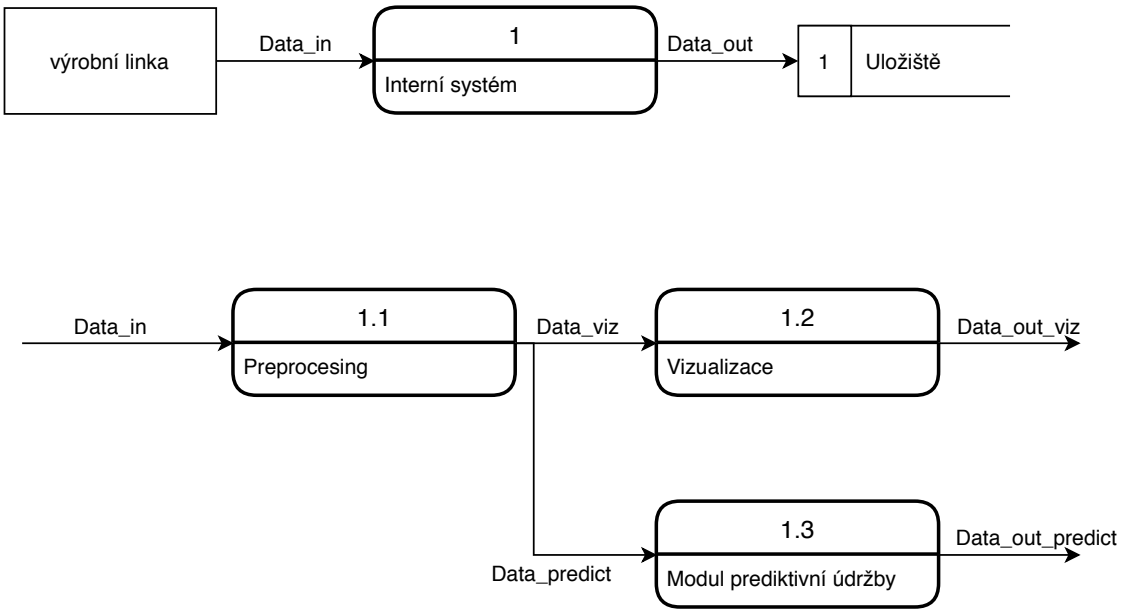
Na obr. 28 je uveden základní use-case diagram pro možnou podobu interního firemního systému s rozšířením o modul prediktivní údržby.



Obrázek 28: Use-case diagram pro interní systém s modulem prediktivní údržby

Na obr. 29 je navržen diagram datových toků pro takový systém. Data z výrobní linky by šla nejprve do modulu zajišťující předzpracování dat a přípravu datasetů, viz. odst. 3.2.4. Dále systém

obsahuje dva další moduly, první je pro vizualizaci dat, který slouží k zobrazení aktuálního stavu linky. Druhý je modul prediktivní údržby, který vyhodnotí potřebu zásahu údržby. Data z těchto modulů budou odesílána ve formě reportů pro management v pravidelných intervalech, dále pokud modul prediktivní údržby vyhodnotí potřebu akutního zásahu, budou tato data odeslána údržbě. Zároveň budou tato data odesílána na firemní úložiště.



Obrázek 29: Diagram datových toků pro interní systém s modulem prediktivní údržby

Samotná realizace modulu pro prediktivní údržbu vyžaduje komplexní a detailní znalost procesů, způsobu měření a sběru dat, samotných dat a v neposlední řadě také metod. V této práci bylo navrženo několik metod shlukové analýzy, které by se pro takovýto případ daly použít. Jako nejnadějnější se jeví samoorganizační mapy. Ty se pro tento účel dají využít dvěma způsoby:

a) Využití chyby kvantizace QE pro detekci anomálních stavů.

b) Porovnávání trajektorie stavu v samoorganizační mapě.

Pro zvýšení robustnosti systému byly v modulu pro prediktivní údržbu využity oba tyto přístupy. Chyba kvantizace QE představuje základní využití samoorganizačních map pro detekci anomálních stavů a je tedy využita v prvním ze dvou podmodulů. Navržený algoritmus je následující:

1) Výběr „zdravých“ stavů procesu x_t, tedy dat, o kterých lze se 100% jistotou říci, že definují bezproblémový provoz, tzn. status zařízení $y_t = 0$.

$$\forall x_t \in X_t, \quad \text{kde} \quad X_t \subset X; \quad y_t(x_t) = 0$$

2) Následně je pomocí těchto zdravých stavů x_t natrénována samoorganizační mapa SOM 1.

3) Pro stavy z procesu vstupující do modulu prediktivní údržby je určena BMU a příslušná chyba kvantizace.

$$QE_i = d(x_i, BMU(x_i))$$

Stavy, které budou podobné „zdravým“ stavům procesu (linky) půjdou samoorganizační mapou dobře aproximovat a tedy QE_i bude mít přibližně konstantní hodnotu. Pro stavy, které se od „zdravých“ stavů budou lišit, bude mít QE_i vyšší hodnoty.

Druhý modul je zajišťuje sledování trajektorie stavu v samoorganizační mapě. V natrénované samoorganizační mapě se sledují reálně-časové stavy procesu, zaznamenává se jejich poloha a vytváří se tzv. trajektorie degradace. Tyto trajektorie by potom mělo být možné porovnat mezi sebou a určit příčiny jejich vzniku. Navržený algoritmus je následující:

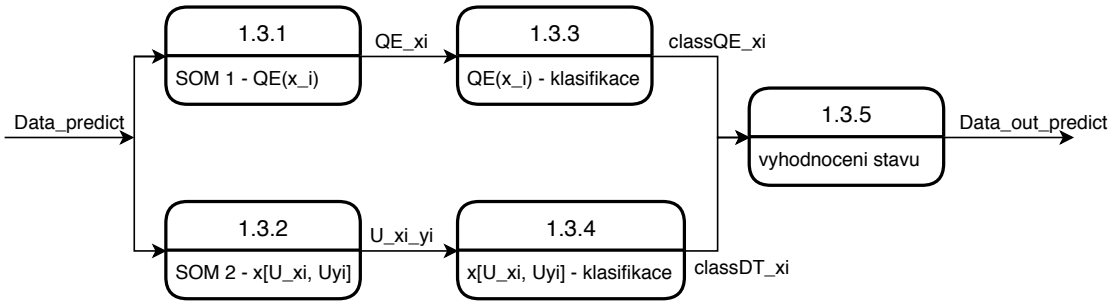
1) Natrénování samoorganizační mapy SOM 2 ze všech stavů matice X.

2) Pro stavy z procesu vstupující do modulu prediktivní údržby je určena BMU a je vyjádřena příslušným prvkem v matici U.

$$d_{BMU} = min\left(\|x_i - w_{i,j}\|\right)$$

3) Pospojováním výskytu stavu v jednotlivých uzlech mapy v průběhu času se určí trajektorie degradace.

Schéma celého modulu je na obr. 30. Předzpracovaný stav x_i jde do podmodulů pro určení chyby kvantizace $QE(x_i)$ a určení souřadnic v matici U, $x[U_{x_i}, U_{y_i}]$. Takto získaná data sou následně v podmodulech pro klasifikaci porovnána se známými scénáři a vyhodnocena. Výstupem těchto klasifikátorů by měl být seznam možných scénářů, které danému průběhu odpovídají a poměr shody s modelovými scénáři pro určení pravděpodobnosti správnosti predikce. V posledním modulu by pak měly být porovnány výstupy z obou klasifikačních podmodulů, určen nejpravděpodobnější scénář následného vývoje a vygenerované doporučení dalšího postupu.

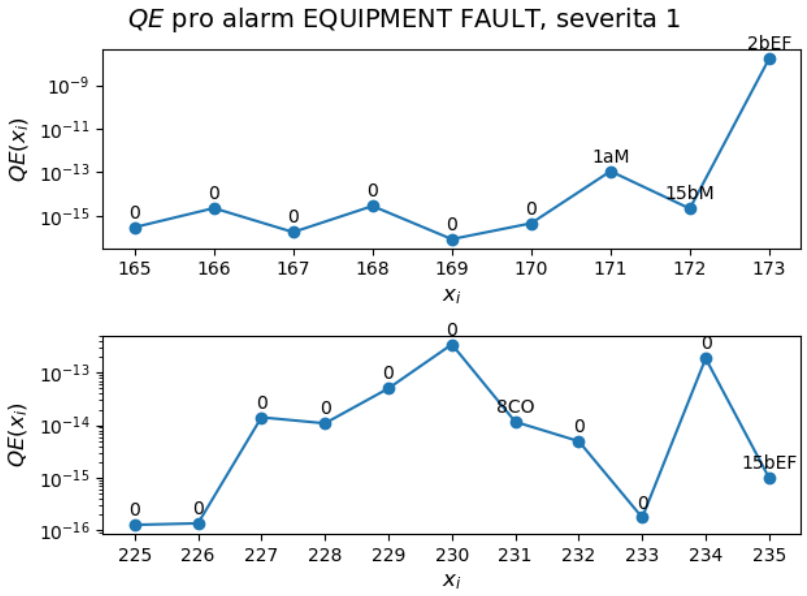


Obrázek 30: Diagram datových toků pro modul prediktivní údržby

5.2 Ověření funkčnosti navržených algoritmů na datech z procesu

Jak vyplývá z popisu algoritmu, základním předpokladem pro tvorbu takového modulu je dostupnost velkého množství historických dat nejen o výrobním systému, ale i o provedených údržbách a zásazích do vybavení. Tato data musí obsahovat příčinné souvislosti, aby bylo možné vysledovat trendy a znaky v datech a jednoznačně identifikovat jednotlivé scénáře. Absence takovýchto souvislostí v datech ze simulátoru byla v této práci již demonstrována.

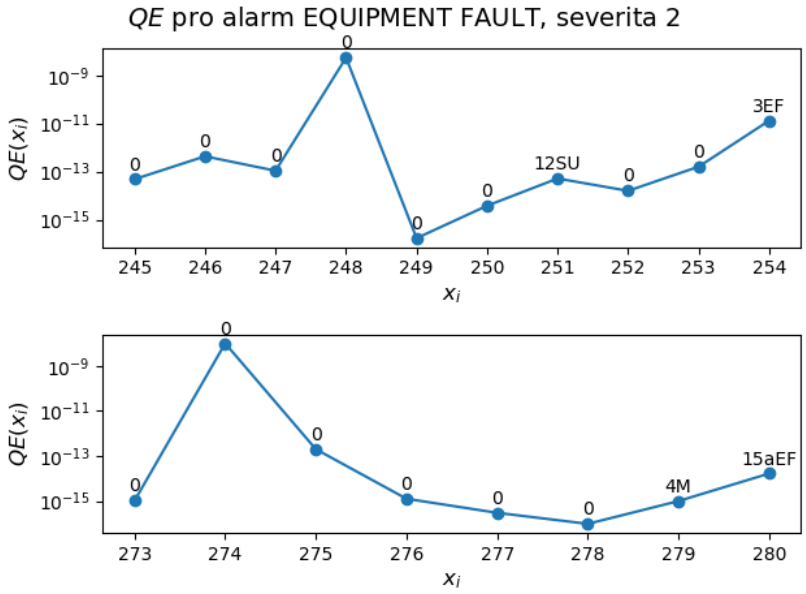
Při podrobném prozkoumání dat ze simulátoru výrobní linky bylo zjištěno, že jediný alarm (scénář), který v některých případech obsahuje trend, který by se dal s určitou mírou jistoty popsat, je `EQUIPMENT FAULT`. Tento alarm indikuje blíže neurčenou závadu na vybavení výrobní linky. Tento alarm se ve zde uvedeném testovacím datasetu vyskytl celkem 8 krát. Problémem je, že jednotlivé stavy s těmito alarmy jsou rozprostřené po celé spodní polovině samoorganizační mapy. Z těchto osmi stavů byly vybrány 2 alarmy severity 1 a 2 alarmy severity 2, na kterých bude demonstrována funkčnost navržených algoritmů.



Obrázek 31: Výsledky pro detekci anomálie pomocí chyby kvantizace pro alarm equipment fault závažnosti 1 - v horním grafu se podařilo demonstrovat vysokou chybu kvantizace pro abnormální stav linky

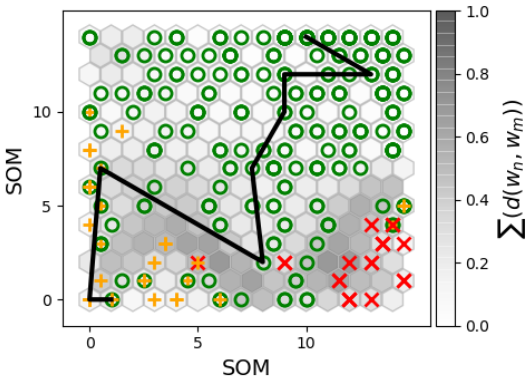
Na obr. 31 a 32 jsou zobrazeny výsledky pro detekci anomálií na základě chyby kvantizace pro alarm `EQIPMENT FAULT` v různých částech linky s různou severitou (závažností). Nejlépe byl tento způsob detekce demonstrován na obr. 31 nahoře, kde před aktivací alarmů dochází k nárůstu QE a stav s tímto alarmem má taktéž nejvyšší hodnotu QE.

Pro závažnost stavu 2, na obr. 32 je v obou případech patrný pozvolný trend zvyšování QE před aktivací daného alarmu, což je kritické pro možnost včasného zásahu, ovšem ani v jednom

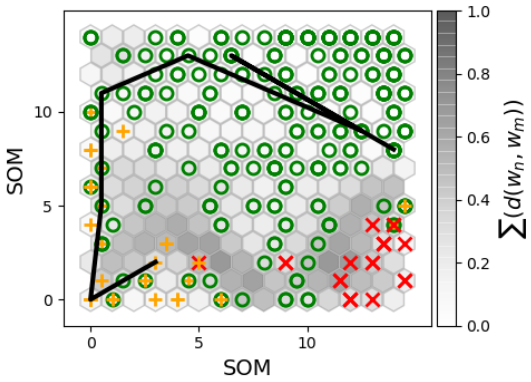


Obrázek 32: Výsledky pro detekci anomálie pomocí chyby kvantizace pro alarm equipment fault závažnosti 2 - v obou případech dochází před aktivací alarmu k pozvolnému nárůstu QE

případě nemá stav s tímto alarmem nejvyšší hodnotu QE.



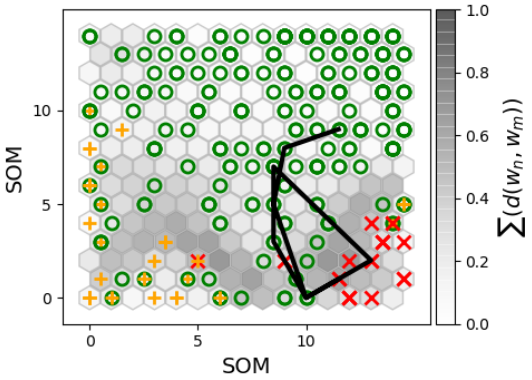
Obrázek 33: Trajektorie degradace pro 2b EF, severita 1 - do oranžového clusteru jde stav přes první sloupec SOM



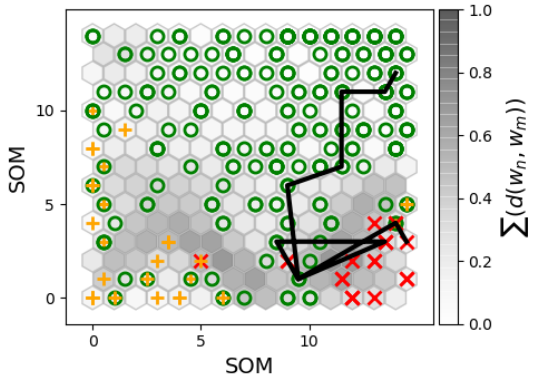
Obrázek 34: Trajektorie degradace pro 15a EF, severita 1 - do oranžového clusteru jde stav přes první sloupec SOM

Na obr. 33 a 34 jsou zobrazeny výsledky algoritmu pro určování trajektorie degradace. Na těchto obrázcích jsou zobrazené trajektorie degradace stavů se severitou 1, jejichž výsledek pro detekci anomálie pomocí QE jsou zobrazeny na obr. 31. Společným ukazatelem je to, že ze zeleného do

oranžového clusteru sestupuje stav po prvním sloupci samoorganizační mapy nejprve do levého dolního rohu a následně dovnitř oranžového clusteru.



Obrázek 35: Trajektorie degradace pro 15b EF, severita 2 - před aktivací červeného alarmu se vytvoří smyčka mezi 2 clustery



Obrázek 36: Trajektorie degradace pro 3 EF, severita 2 - před aktivací červeného alarmu se vytvoří smyčka mezi 2 clustery

Na obr. 35 a 36 jsou zobrazeny výsledky algoritmu pro určování trajektorie degradace stavů ze závažností 2 (výsledky detekce anomálií pomocí QE pro tyto stavy je na obr. 32). Tyto trajektorie mají společné to, že se do červeného clusteru dostávají v poslední „třetině“ SOM a po průniku do červeného clusteru se nejprve vytvoří smyčka zpět do zeleného a až následně při druhém průniku do červeného clusteru začíná být alarm aktivní.

6 Závěr

V této sekci budou shrnuty a diskutovány výsledky jednotlivých částí diplomové práce.

6.1 Shrnutí vstupních dat

V práci byly použity 2 druhy vstupních dat; data z procesu získaná ze simulátoru výrobní linky firmy mySCADA a umělá data odvozená z poznatků o této lince s ohledem na vyšší pravděpodobnost úspěšné implementace zkoumaných metod.

Umělá data byla využita zejména pro demonstraci a ověření funkčnosti zvolených metod a algoritmů. Výsledky těchto analýz byly použity pro porovnání s výsledky dat z procesu.

Data z procesu byla rozdělena do dvou datasetů pro testování a trénování. Již samotné získání vhodných dat bylo problematické a bylo potřeba vybrat pouze úseky, které vykazují podobné chování. Data ze simulátoru byla předzpracována a tento postup byl zaznamenán v odst. 3.2.4.

6.2 Shrnutí návrhu, využití a aplikace metody PCA

Metoda PCA byla použita jako první nástroj analýzy dat. Pro umělá data bylo dosaženo výborného oddělení jednotlivých clusterů. Lze tedy předpokládat, že i u následujících sofistikovanějších metod bude dosaženo dobrého výsledku.

Na rozdíl od výsledků pro umělá data se pro data z procesu nepodařilo vytvořit oddělené shluky. Jak již bylo zmíněno výše, clusterování pomocí metody PCA lze zařadit mezi spíš jednodušší nástroje a tudíž lze říci, že takový výsledek nebyl překvapivý. Pro tréninkový dataset se podařilo v jisté míře oddělit stavy se statusem 0, které značí bezproblémový stav linky. To může být způsobeno i tím, že tréninkový dataset obsahuje více stavů než testovací.

6.3 Shrnutí návrhu, využití a aplikace metody k-Means

Před samotným clusterováním pomocí metody k-Means byla nepjrve provedena analýza s cílem určení optimálního počtu clusterů pro jednotlivé datasety. K tomu byla využita metoda lokte (angl. Elbow Method), metoda siluet (angl. Silhouette Method) a Calinski-Harabaszův index. Postup byl pro všechny metody stejný - výpočet k-Means a pro zvolený počet shluků určit příslušný koeficient. Výsledky pro dané datasety v minimálně dvou ze tří použitých metod potvrdily, že optimální počet clusterů je 3.

Tento poznatek byl následně použit v algoritmu metody k-Means. Nejprve byl algoritmus aplikován na původní 4D dataset a byla zkoumána shoda původních a predikovaných statusů stanoviště. Pro umělá data bylo dosaženo 100% shody výsledků a byl tedy potvrzen dobrý výsledek metody PCA.

Pro data ze simulátoru bylo dosaženo 87% shody. Ačkoli metoda k-Means zkoumá podobnost dat pouze na základě euklidovské metriky, je otázkou, zda status stanoviště určený simulátorem dat skutečně odpovídá reálnému stavu. Toto bohužel ze simulátoru nelze ověřit.

Následná shluková analýza, která kombinovala metody PCA a k-Means potvrdila problematické zařazování některých bodů dat z procesu do příslušných clusterů.

6.4 Shrnutí návrhu, využití a aplikace metody t-SNE

Shluková analýza t-SNE byla provedena dvěma způsoby: přímou aplikací algoritmu na vstupní data a pozvolnou redukcí dimenzí vstupního datasetu nejprve metodou PCA a následně redukce dimenzí a shluková analýza pomocí t-SNE.

Rozdíl ve výsledku těchto dvou přístupů se projevil zejména „těsnějším“ uspořádáním stavů v jednotlivých clusterech pro metodu s postupnou redukcí dimenzí.

Oproti předpokladu, že postupné snižování dimenzí povede k lepší identifikaci jednotlivých znaků ve stavech, nebylo touto metodou ani v datech ze simulátoru dosaženo výraznějšího zlepšení. Vytvořené clustery se sice jeví kompaktnější, ovšem nepodařilo se oddělit oranžový (status

stanoviště 1) a červený (status stanoviště 2) cluster. Rozdíly mezi výsledky pro trénovací a testovací data jsou dány nejspíše pouze počtem vzorků v datasetu. Stejně jako v předchozích metodách dochází k zařazení „cizích“ bodů do clusterů, zejména stavy se statusem 0 se nachází ve všech zobrazených clusterech.

6.5 Shrnutí návrhu, využití a aplikace samoorganizačních map

Analýza samoorganizační mapou představuje nejkomplexnější nástroj použitý v této práci. Parametry algoritmu byly optimalizovány pro data ze simulátoru a následně beze změny použity na analýzu umělých dat. Kromě parametrů výpočtu byl zkoumán také vliv způsobu učení na výsledky. Pro umělá data neměl způsob učení výrazný vliv na výsledek, ovšem pro data ze simulátoru bylo dosaženo výrazně lepších výsledků, pokud byly stavy (řádky vstupní matice X) promíchány v náhodném pořadí. Způsob učení měl v tomto případě vliv zejména na míru oddělení vytvořených clusterů.

Pro umělá data bylo i v tomto případě dosaženo výborných výsledků, všechny clustery jsou od sebe výrazně odděleny a nedochází k „míchání“ stavů do cizích clusterů.

Pro data ze simulátoru bylo touto metodou dosaženo oddělení všech tří clusterů, ovšem stále dochází k výskytu „cizích“ stavů v jednotlivých clusterech.

Jelikož nebylo možno ani jednou metodou rozdělit data do shluků tak, aby se v daném shluku nacházely pouze stavy se stejným statusem stanoviště, lze konstatovat, že takový dataset nelze použít ke spolehlivé predikci budoucího vývoje. Důvodem může být to, že stav použitý k definici statusu stanoviště není kompletní, nebo fakt, že generátor náhodných hodnot, který simulátor řídí, neposkytuje dostačující vzory v datech pro jejich jednoznačné clusterování.

Dalším krokem bylo tedy zjistit, zda lze upravit stav z dostupných veličin tak, aby byly vzory v datech zvýrazněny. První ze zkoumaných možností byla transformace stavu do polynomiálního kernelu:

$$X = \begin{bmatrix} X_{1,1}^2 & X_{1,1}X_{1,2} & X_{1,1}X_{1,3} & X_{1,1}X_{i,4} & X_{1,2}^2 & \dots & X_{1,4}^2 \\ X_{2,1}^2 & X_{2,1}X_{2,2} & X_{2,1}X_{2,3} & X_{2,1}X_{2,4} & X_{2,2}^2 & \dots & X_{2,4}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ X_{N,1}^2 & X_{N,1}X_{N,2} & X_{N,1}X_{N,3} & X_{N,1}X_{N,4} & X_{N,2}^2 & \dots & X_{N,4}^2 \end{bmatrix} \tag{33}$$

Tímto přístupem bylo dosaženo výrazného zhoršení výsledků, a tudíž nebyl v této práci dále rozvíjen.

Druhým přístupem bylo vytvoření stavu i s pomocí hodnot na předchozím stanovišti, jak je popsáno v odst. 4.2.5. Ani touto metodou nebylo dosaženo lepších výsledků.

Kromě shlukové analýzy statusů stanoviště byly také analyzovány alarmy vyskytující se ve stejných clusterech. Byly formulovány některé obecné společné znaky stavů vyskytujících se v jednotlivých clusterech.

6.6 Shrnutí modulu prediktivní údržby

V této části bylo představeno možné využití poznatků získaných shlukovou analýzou v praxi a byl navržen základ modulu prediktivní údržby s využitím shlukové analýzy.

Jelikož jsou alarmy, které řídí průběh ostatních veličin, generovány náhodně, nelze určit dostatečně dlouhodobé trendy, které by ukazovaly na postupné opotřebení jednotlivých komponent, a tudíž nebylo praktické využití dat realizováno. Byla popsána metodika získávání veličin potřebných k realizaci modulu prediktivní údržby a bylo demonstrováno její využití pro simulovaná data. Realizace kompletního modulu pro prediktivní údržbu byla popsána teoreticky.

6.7 Závěr - cíle diplomové práce

V souladu se zadáním byly v diplomové práci navrženy algortimy nesupervizorovaného strojového učení pro monitorování a vyhodnocování stavů. Byla provedena rešerše dostupných metod. Na základě analýzy dat simulovaného výrobního procesu poskytnutého firmou dodávající řešení sběru a vizualizace dat byl následně navržen základ metodiky využití pro reálné aplikace a na těchto datech demonstrován.

Povedlo se ověřit předpoklady využití jednotlivých metod pro data z výrobních procesů a navrhnout metodiku pro využití těchto metod pro prediktivní údržbu.

Nepodařilo se v datech z procesu nalézt dostatečné funkční závislosti k tvorbě plně funkčního modulu pro prediktivní údržbu. Práce na tomto modulu bude dále pokračovat ve spolupráci s firmou mySCADA s.r.o., která ho v budoucnu chce integrovat do svého systému pro sběr a vizualizaci dat.

Reference

[1] R. K. Mobley. *An introduction to predictive maintenance.* Elsevier, 2 edition, 2002.

[2] Údržba - Terminologie údržby. Norma, Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2018.

[3] V. Legát. *Management a inženýrství údržby.* Professional Publishing, 2 edition, 2016.

[4] V. Liška. Řízení preventivní údržby a skladu kritických náhradních dílů, 2018. [Online; 05.06.2020]. URL: Dostupné na: https://dspace.cvut.cz/bitstream/handle/10467/80598/F2-DP-2018-Liska-Vlastimil-Rizeni%20preventivni%20udrzby%20a%20skladu%20kritickych%20nahradnich%20dilu.pdf?sequence=-1isAllowed=y.

[5] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, and J. Loncarski. Machine learning approach for predictive maintenance in industry 4.0. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6. IEEE, 2018.

[6] T. Ohno. *Toyota production system: beyond large-scale production.* Productivity Press, 1 edition, 1988.

[7] S. M. Dahlgaard-Park and J. Pettersen. Defining lean production: some conceptual and practical issues. *The TQM journal*, 2009.

[8] M. Rao, C. Theisen, and J. T. Luxhoj. Intelligent system for air-traffic control. In *Proceedings. 5th IEEE International Symposium on Intelligent Control 1990*, pages 859–863 vol.2, 1990.

[9] K. X. Zhu. Sensor-based condition monitoring and predictive maintenance—an integrated intelligent management support system. *Intelligent Systems in Accounting, Finance & Management*, 5(4):241–258, 1996.

[10] Y. Tu and E. H. H. Yeung. Intelligent decision support system (idss) for equipment diagnosis and maintenance management. *Proceedings of 3rd International Conference on Manufacturing Technology in Hong Kong, pp. 658-667*, 1995.

[11] Y. Tu and E. H. H. Yeung. Integrated maintenance management system in a textile company. *The International Journal of Advanced Manufacturing Technology*, 13(6):453–461, 1997.

[12] G. Doumeingts, E. Dumora, M. Chabanas, and J. F. Huet. Use of grai method for the design of an advanced manufacturing system. In *Proc. 6th Int. Conf. Flexible Manufacturing Systems*, pages 341–358, 1987.

[13] S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen. Hugin-a shell for building bayesian belief universes for expert systems. In *IJCAI*, volume 89, pages 1080–1085, 1989.

[14] G. A. Carpenter and S. Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, 1988.

[15] R. J. McDuff, P. K. Simpson, and D. Gunning. An investigation of neural networks for f-16 fault diagnosis. i. system description. In *IEEE Automatic Testing Conference.The Systems Readiness Technology Conference. Automatic Testing in the Next Decade and the 21st Century. Conference Record.*, pages 351–357, 1989.

[16] G. M. Knapp and H. P. Wang. Machine fault classification: a neural network approach. *International Journal of Production Research*, 30(4):811–823, 1992. https://doi.org/10.1080/00207543.1992.9728458 doi:10.1080/00207543.1992.9728458.

[17] K. Suzuki. *Artificial Neural Networks: Industrial and Control Engineering Applications*. In Tech, 2011.

[18] C. C. Lin and H. P. Wang. Performance analysis of rotating machinery using enhanced cerebellar model articulation controller (e-cmac) neural networks. *Computers & industrial engineering*, 30(2):227–242, 1996.

[19] J. S. Albus. Data storage in the cerebellar model articulation controller (cmac). 1975.

[20] N. S. Pai, H. T. Yau, T. H. Hung, and C. P. Hung. Application of cmac neural network to solar energy heliostat field fault diagnosis. *International Journal of Photoenergy*, 2013, 01 2013. https://doi.org/10.1155/2013/938162 doi:10.1155/2013/938162.

[21] N. Amruthnath and T. Gupta. A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 355–361, 2018.

[22] K. T. P. Nguyen and K. Medjaher. A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliability Engineering & System Safety*, 188:251–262, 2019.

[23] S. Butte, A. R. Prashanth, and S. Patil. Machine learning based predictive maintenance strategy: A super learning approach with deep neural networks. In *2018 IEEE Workshop on Microelectronics and Electron Devices (WMED)*, pages 1–5, 2018.

[24] H. Du, M. Inui, M. Ohkita, K. Fujimura, and H. Tokutaka. Short-term prediction of oil temperature change of an indoor transformer by self-organizing map (som). In *2002 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No.02CH37309)*, volume 2, pages 1366–1371 vol.2, 2002.

[25] Jingjie Chen, Jiusheng Chen, and X. Zhang. Reliability prediction model of aircraft using self-organizing map. In *2007 IEEE International Conference on Automation and Logistics*, pages 680–683. IEEE, 2007.

[26] K. Dhalmahapatra, R. Shingade, H.n Mahajan, A. Verma, and J. Maiti. Decision support system for safety improvement: an approach using multiple correspondence analysis, t-sne algorithm and k-means clustering. *Computers & Industrial Engineering*, 128:277–289, 2019.

[27] L. Moreira, C. Dantas, L. Oliveira, J. Soares, and E. Ogasawara. On evaluating data preprocessing methods for machine learning models for flight delays. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[28] P. Pandey. Data preprocessing : Concepts. Dostupné na https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825. [Online; 09.05.2020].

[29] F. E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, únor 1969. DOI: 10.1080/00401706.1969.10490657.

[30] H. C. Mandhare and S. R. Idate. A comparative study of cluster based outlier detection, distance based outlier detection and density based outlier detection techniques. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 931–935, 2017.

[31] J. Reback a spol. pandas-dev/pandas: Pandas 1.0.3, březen 2020. Dostupné na: https://doi.org/10.5281/zenodo.3715232.

[32] J. Kelbel a D. Šilhán. Shluková analýza. Dostupné na: http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis_prednasky/zapis_02/13/shlukovani.pdf, květen 2002.

[33] Wikipedia contributors. Principal component analysis — Wikipedia, the free encyclopedia. Dostupné na: https://en.wikipedia.org/w/index.php?title=Principal_ component_ analysi-soldid=961671758, 2020. [Online; 14.06.2020].

[34] J. Lever, M. Krzywinski, and Altman. Principal component analysis. *Nat Method*, 14:641–642, červen 2017. https://doi.org/10.1038/nmeth.4346.

[35] Inc. The MathWorks. Calinskiharabaszevaluation. Dostupné na https://https://www.mathworks.com/help/stats/clustering.evaluation.calinskiharabaszevaluation-class.html. [Online; 01.06.2020].

[36] A. Bhardwaj. Silhouette coefficient: Validating clustering techniques. Dostupné na https://https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c. [Online; 01.06.2020].

[37] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[38] G. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.

[39] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358–1384, 1996.

[40] Eklavya. Kohonen self-organizing maps. Dostupné na: https://towardsdatascience.com/kohonen-self-organizing-maps-a29040d688da, 2019. [Online; 12.06.2020].

[41] G. Vettigli. Minisom: minimalistic and numpybased implementation of the self organizing map., červen 2020. URL: Dostupné na: https://github.com/JustGlowing/minisom, https://doi.org/10.5281/zenodo.3715232 doi:10.5281/zenodo.3715232.

[42] A. Von Birgelen, D. Buratti, J. Mager, and O. Niggemann. Self-organizing maps for anomaly localization and predictive maintenance in cyber-physical production systems. *Procedia CIRP*, 72:480–485, 2018.

Seznam obrázků

Seznam tabulek