UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

DETEKCIA ÚNIKOV Z POTRUBÍ S VYUŽITÍM METÓD UMELEJ INTELIGENCIE

DIPLOMOVÁ PRÁCA

2024 **Bc. Lukáš Grofčík**

UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

DETEKCIA ÚNIKOV Z POTRUBÍ S VYUŽITÍM METÓD UMELEJ INTELIGENCIE

DIPLOMOVÁ PRÁCA

Študijný odbor:	18. Informatika
Študijný program:	Aplikovaná informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	doc. Martin Drlík, PhD.

Nitra 2024 Bc. Lukáš Grofčík

Univerzita Konštantína Filozofa v Nitre
Fakulta prírodných vied a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Lukáš Grofčík
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: Diplomová práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Detekcia únikov z potrubí s využitím metód umelej inteligencie

Anotácia: Transport rôznych surovín produktovodmi na dlhé vzdialenosti je permanentne kontrolovaný sieťou senzorov, ktoré zbierajú rôzne fyzikálne dáta. Prípadný únik suroviny z produktovodu sa prejaví v dátach získaných týmito senzormi. Detekcia únikov sa v súčasnosti realizuje hlavne s pomoocu fyzikálných hydraulických modelov. Práca sa zameriava na overenie potenciálneho prínosu metód strojového a hlbokého učenia pri detekcii týchto únikov a ich porovnanie s tradičnými metódami.
Cieľ práce:
Cieľom práce je na príklade dát o rôznych stavoch produktovodov získaných zo simulátora overiť potenciál metód detekcie únikov s využitím metód strojového a hlbokého učenia. Študent analyzuje dostupné riešenia, architektúry, metódy a technológie. V praktickej časti podrobne popíše jednotlivé fázy procesu objavovania znalostí pri riešení problému detekcie únikov, zosumarizuje získané výsledky a porovná ich s výsledkami dosiahnutými tradičnými metódami.
Charakter práce:
Práca má aplikačný charakter, bude obsahovať popis riešeného problému, návrh riešenia, metodiku prípravy dát, implementáciu, popis vytvoreného riešenia, interpretáciu a zhodnotenie získaných výsledkov.
Predmetové prerekvizity:
• Objavovanie znalostí (1. Mgr.)
• Neurónové siete (1. Mgr.)
• HĺbkovÁ analýza dát (1. Mgr.)
• Technológie spracovania veľkých dát (1., Mgr.)
Najdôležitejšie kompetentnosti získané spracovaním témy (cca 5):
• poznať spôsoby ukladania a získavania štruktúrovaných a neštruktúrovaných dát,
• rozumieť jednotlivým činnostiam a fázam procesu objavovania znalostí,
• ovládať programovací jazyk dátovej vedy,
• pracovať s rôznymi technológiami pre spracovanie a pokročilú analýzu dát,
• prepájať oblasť dátovej vedy a softvérového inžinierstva,
• referovať o výsledkoch analýzy a správne ich interpretovať.

Kľúčové slová: vývoj systémov, strojové učenie, hlboké učenie, MLOps

Školiteľ: doc. Mgr. Martin Drlík, PhD.
Oponent: doc. Ing. Štefan Koprda, PhD.
Katedra: KI - Katedra informatiky

Dátum zadania: 31.10.2022

Dátum schválenia: 07.03.2024

RNDr. Ján Skalka, PhD., v. r.
vedúci/a katedry

POĎAKOVANIE

V tejto časti by som chcel vyjadriť poďakovanie môjmu školiteľovi pánovi doc. Mgr. Martinovi Drlíkovi, PhD. za odbornú pomoc a rady, ktoré mi poskytol a za ochotu a trpezlivosť, ktorú so mnou mal pri vypracovaní tejto práce. Taktiež by som sa chcel poďakovať firme ttc, s.r.o. za príležitosť sa podieľať na vypracovaní projektu tejto práce a za konzultácie, ktoré nám pomohli vypracovať túto prácu.

ABSTRAKT

GROFČÍK, Lukáš: Detekcia únikov z potrubí s využitím metód umelej inteligencie. [Diplomová práca]. Univerzita Konštantína Filozofa v Nitre. Fakulta prírodných vied a informatiky. Katedra informatiky. Školiteľ: doc. Mgr. Martin Drlík, PhD. Stupeň odbornej kvalifikácie: Magister odboru Aplikovaná informatika. Nitra: FPVaI, 2024. 60 s.

Táto práca má za cieľ natrénovať funkčnú neurónovú sieť určenú na predikciu únikov tekutín v potrubí v reálnom čase. Budeme používať technológie, ktoré sú momentálne využívané v praxi. V teoretickej časti uvádzame analýzu súčasného stavu vývoja strojového učenia a umelej inteligencie. Vysvetľujeme metodiku používanú pri vývoji. Praktická časť opisuje postup vývoja neurónovej siete typu Autoencoder. Opisujeme jednotlivé fázy a úlohy vývoja, ktoré sme dodržovali podľa metodiky CRISP-DM. Používame aktuálne nástroje pre automatizáciu jednotlivých fáz postupu. Výsledkom práce je model, ktorý dokáže rozpoznať únik v potrubí, pokiaľ nie je používané čerpadlo. V práci vysvetľujeme, že technológia neurónovej siete Autoencoder s využitím rekurentných vrstiev LSTM by mohla poskytnúť dodatočný náhľad k detekcii únikov v potrubí s tekutinou.

ABSTRACT

GROFČÍK, Lukáš: Detection of pipeline leaks using artificial intelligence methods. [Master Thesis]. Constantine the Philosopher University in Nitra. Faculty of Natural Sciences and Informatics. Department of informatics. Supervisor: doc. Mgr. Martin Drlík, PhD. Degree of Qualification: Master of Applied Informatics. Nitra: FNSaI, 2024. 60 p.

This work has a goal of developing a functional neural network, which can be used to predict fluid leaks in a pipeline in real time. We will be using technologies, which are currently being used in practice. In theoretical part we mention the current state of development of machine learning and artificial intelligence. We explain the methodic which is used during the development. Practical part describes the procedure of the development of neural network Autoencoder. We describe individual phases and tasks of development, which we complied with according to CRISP-DM. We use tools which are nowadays used for automatization of these phases. The output of this work is a model, which is able to recognize a leak in a pipeline if a pump is not being used. In this work we explain, that the technology of neural network Autoencoder used with recurrent layers LSTM could provide additional insight into the detection of leaks in a pipeline with fluids.

OBSAH

Úvod. 8

1 Analýza súčasného stavu 9

1.1 Neurónová sieť 9

1.1.1 Dopredná neurónová sieť 9

1.1.2 Rekurentná neurónová sieť 10

1.1.3 Long short-term memory 11

1.2 Detekcia anomálií 12

1.2.1 Štatistické metódy 13

1.2.2 Klasifikačné metódy 13

1.2.3 Zhlukové metódy 13

1.2.4 Metódy založené na časových rádoch 14

1.2.5 Metriky pre meranie algoritmov 14

1.2.6 Kolektívne anomálie 15

1.3 Modely vývoja 15

1.3.1 CRISP-DM 15

1.3.2 ASUM-DM 21

1.4 Princípy vývoja 22

1.5 MLOps 22

1.5.1 GitHub Actions 25

2 Ciele záverečnej práce 27

3 Vývoj neurónovej siete 28

3.1 Porozumenie problému 28

3.2 Porozumenie dát 29

3.3 Príprava dát 32

3.4 Modelovanie 35

3.5 Evalvácia 43

3.5 Nasadenie 44

4 Diskusia 54

Záver 56

Zoznam bibliografických odkazov 58

Zoznam príloh 1

ÚVOD

V dnešnej dobe môžeme pozorovať obrovský vzostup používania technológií strojového učenia a umelej inteligencie v oblasti informačných technológií. Veľké množstvo firiem a spoločností sa potrebuje adaptovať požiadavkám svojich zákazníkov a z tohto dôvodu sa začínajú sústreďovať na strojové učenie, ktoré im dokáže zjednodušiť podnikové procesy, znížiť náklady na využívané zdroje alebo potenciálne zvýšiť zisk.

Cieľom našej práce bude predstaviť technológie, ktoré sú používané v dnešnej dobe pri práci so strojovým učením a predstaviť ako sú tieto technológie používané. Umelá inteligencia je využívaná na riešenie problémov kde ľudský faktor zlyháva. Jeden z takýchto faktorov je aj rýchlosť, ktorou dokáže človek zachytiť problém. Medzi takéto problémy patrí únik cenných tekutín v potrubiach. Spoločnosti, ktoré tieto potrubia prevádzkujú, potrebujú získať spôsob, ktorým by dokázali čo najrýchlejšie a najpresnejšie určiť, či sa v potrubí odohráva únik danej tekutiny. Keďže údaje o týchto únikoch sú ovplyvnené mnohými faktormi, ktoré nemožno merať, ako napríklad veľkosť otvoru, cez ktorý prechádza únik, tak tieto spoločnosti potrebujú využiť technológie umelej inteligencie a strojového učenia.

Výsledkom našej práce je natrénovaná neurónová sieť, ktorá dokáže rozpoznať výskyt úniku v potrubí s tekutinou podľa požiadaviek nášho klienta. Tento model bude veľkým prínosom pre klienta, pretože mu umožní zistiť, kedy sa nachádza v potrubí otvor, ktorý spôsobuje únik tekutiny. Týmto spôsobom bude možné vyriešiť problémy, ktoré nastanú kvôli strate tekutiny v rýchlejšom čase a zamedzí sa potenciálne vysokým stratám zdrojov.

V praktickej časti použijeme metodiku CRISP-DM na vývoj neurónovej siete. Budeme dodržovať jednotlivé kroky a úlohy zadefinované touto metodikou, aby sme docielili funkčný model schopný, ktorý je možné používať v praxi. Dokončené úlohy nakoniec automatizujeme pre budúce používanie pomocou technológie GitHub Actions, ktorá je v dnešnej dobe používaná na automatizáciu a nasadzovanie projektov do produkčných prostredí.

1 ANALÝZA SÚČASNÉHO STAVU

V posledných rokoch sa objavovanie znalostí z údajov a prediktívna analýza stávajú rastúcim trendom v priemyselných podnikoch. Spoločnosti potrebujú zabezpečiť svoj rast pomocou informovaných rozhodnutí a kvalitu svojich služieb pomocou predikcie problémov. Tieto riešenia je momentálne možné získať iba pomocou dolovania údajov.

Pre vývoj daných riešení je potrebné pre vývojárov použiť moderné princípy na zabezpečenie kvality, technológie na samotná vývoj riešenia a metodológie, ktoré sú overené a využívané na dnešnom trhu, aby mohli vývojári priniesť svoj produkt rýchlo, ale kvalitne.

1.1 NEURÓNOVÁ SIEŤ

Neurónová sieť počíta funkciu od vstupných uzlov po výstupné uzly. Každý uzol má vo vnútri premennú, ktorá môže byť natvrdo priradená alebo je výsledkom výpočtu. Štruktúra neurónovej siete je orientovaný acyklický graf. Hrany sú parametrizované váhami, čo znamená, že funkcie vypočítané na jednotlivých uzloch sú ovplyvnené váhami prichádzajúcich hrán. Celková funkcia vypočítaná sieťou je výsledkom kaskádových výpočtov funkcií na jednotlivých uzloch. V jednovrstvovej sieti je množina vstupov priamo spojená s jedným alebo viacerými výstupnými uzlami a výstupné uzly počítajú funkciu svojich vstupov a váh na hranách. Cieľom neurónovej siete je naučiť sa funkciu, ktorá spája jeden alebo viacero vstupov s jedným alebo viacerými výstupmi pomocou príkladov tréningu (Aggarwal, 2023).

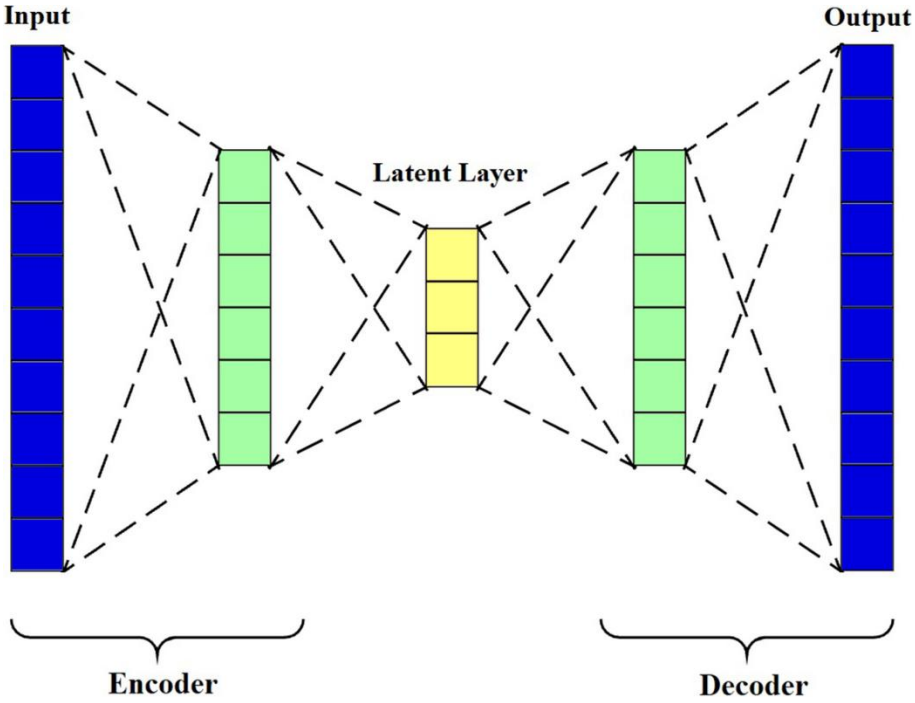
1.1.1 Dopredná neurónová sieť

Dopredná neurónová sieť (*Feedforward Neural Network*, ďalej FFNN) je jedným z dvoch širokých typov umelej neurónovej siete, ktorá sa vyznačuje smerom toku informácií medzi jej vrstvami. Jej tok je jednosmerný, čo znamená, že informácie v modeli prúdia iba jedným smerom (Wesley, 1994).

Autoencoder

Podľa autora Arafa (2023) je Autoencoder neurónová sieť s latentnou vrstvou (latent space), v ktorej sú uložené stlačené verzie pôvodných vstupných údajov. Ako môžeme vidieť na obrázku 1, Autoencoder sa skladá z dvoch častí. Prvou časťou je Encoder, ktorý vykonáva nelineárnu transformáciu vstupných dát na dáta väčšinou nižšej dimenzie. Druhá časť, Decoder, rekonštruuje zakódované dáta do pôvodnej

podoby. Autoencoder bol vyvinutý hlavne na extrakciu prvkov, čím sa znížila vysoká dimenzionalita údajov, vďaka čomu následne mohli byť údaje pripravené na klasifikáciu, ale používa sa aj na detekciu anomálií alebo klasifikačné problémy. Keďže Autoencoder je nelineárna metóda redukcie vlastností, má lepší výkon v porovnaní s inými prístupmi redukcie vlastností.



Obrázok 1 Architektúra neurónovej siete Autoencoder [1]

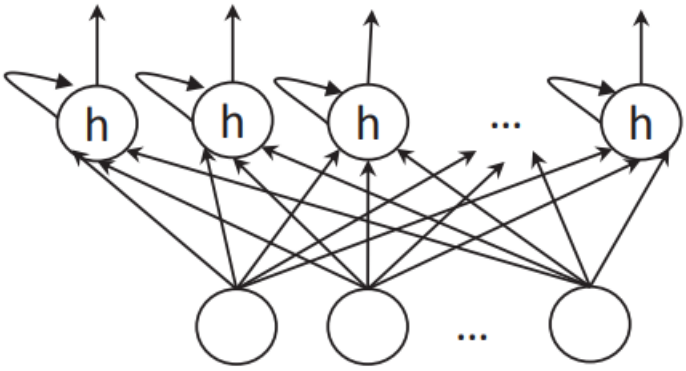
Autor Salvaris (2018) ďalej vysvetľuje, že Autoencoder nie je navrhnutý tak, aby presne kopíroval vstupné dáta, ale je obmedzený tak, že sa môže učiť len približne. Kvôli tomu, že Autoencoder ukladá vzory v menšej dimenzii, učí sa najdôležitejšie vlastnosti na rekonštrukciu vstupných údajov. Z tohto dôvodu môže byť užitočný pre aplikácie učenia bez učiteľa, kde neexistuje žiadna cieľová hodnota pre predikciu alebo učenie. Ukázali sa ako sľubné pre mnohé aplikácie, ako napríklad systémy odporúčaní.

1.1.2 Rekurentná neurónová sieť

Rekurentná neurónová sieť (*Recurrent Neural Network*, ďalej RNN) priamo využívajú sekvenčné informácie. Sekvencie, ktoré prešli do siete môžu byť na vstupe, výstupe alebo dokonca v oboch. RNN spracováva sekvencie údajov prostredníctvom tzv. stavu. Na rozdiel od Konvolučnej neurónovej siete (*Convolutional Neural*

[1] Zdroj Obrázok 1: https://jbioleng.biomedcentral.com/articles/10.1186/s13036-022-00319-3/figures/1

Network), ktorá je typu FFNN, RNN obsahuje slučky v štruktúre siete, ako je znázornené na obrázku 2 (Salvaris, 2018).



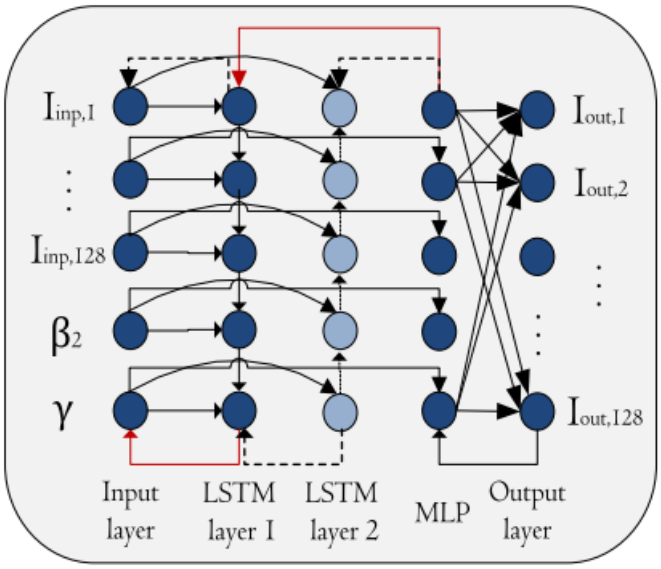
Obrázok 2 Rekurentná neurónová sieť [2]

Pozícia v poradí sa označuje aj ako časová pečiatka. Preto namiesto premenlivého počtu vstupov v jedinú vstupnú vrstvu, sieť obsahuje premenlivý počet vrstiev a každá vrstva má jeden vstup zodpovedajúci tejto časovej pečiatke. Vstupy tiež priamo interagujú so skrytými vrstvami v závislosti od ich pozícií v sekvencii (Aggarwal, 2023).

1.1.3 Long short-term memory

Long short-term memory (ďalej LSTM) je druh rekurentnej neurónovej siete. Typická LSTM Sieť (Obrázok 3) pozostáva iba z jednej vrstvy LSTM, ktorá obsahuje informácie z predchádzajúcich časových krokov. Prítomnosť dvoch vrstiev LSTM znamená, že sieť má informácie z minulosti aj budúcnosti. Takýto druh LSTM sa označuje ako BiLSTM (Gautam, 2021).

[2] Zdroj Obrázok 2: https://link.springer.com/chapter/10.1007/978-1-4842-3679-6_7



Obrázok 3 Štruktúra neurónovej siete LSTM [3]

Autor (Ushiku, 2021) ďalej vysvetľuje, že LSTM, ktorá je variáciou RNN, je využitá na spracovanie dlhých sekvenčných dát. Poskytuje nápravu pre problém miznúceho gradientu a problém explodujúceho gradientu pôvodného RNN.

Problém miznúceho gradientu nastáva počas trénovania neurónových sietí, ktoré obsahujú väčšie množstvo skrytých vrstiev. Počas metód, ktoré využívajú učenie založené na gradientoch, ako napríklad spätná propagácia, váhy sú aktualizované úmerne hodnote gradientu. Počas niektorých prípadov je hodnota gradientu malá a kým sa počas propagácie dostane k počiatočným vrstvám je príliš malá na to, aby dokázala ovplyvniť váhy vrstiev (Sunitha, 2020).

1.2 DETEKCIA ANOMÁLIÍ

Detekcia anomálií je proces identifikácie údajových hodnôt (sekvencie), ktoré sa odchyľujú od očakávaní. Existuje množstvo aplikácií v reálnom svete na detekciu anomálií, vrátane identifikácie porúch senzorov, kontrola kvality výroby produktu, detekcia narušenia alebo pandemické geopriestorové modelovanie. Efektívna detekcia anomálií v továrni má za následok lepšiu dostupnosť, kvalitu produktu, bezpečnosť pracovníkov a znížené náklady na prepracovanie (Siegel, 2020).

Výskumníci zvyčajne definujú anomáliu spôsobom, ktorý najlepšie vyhovuje cieľovej aplikácii (Chandola, 2017).

[3] Zdroj Obrázok 3: https://www.sciencedirect.com/science/article/abs/pii/S1068520021000894

1.2.1 Štatistické metódy

Detekciu anomálií si môžeme predstaviť ako jednoduchý štatistický problém, kde potrebujeme nájsť prvky pod alebo nad kvartilovým rozsahom. Dátové body pod minimom a nad maximom sú anomálne dátové inštancie (Mukherjee, 2020).

Autor (Chandola, 2017) vysvetľuje, že štatistické metódy odhadujú parametrický alebo neparametrický model z údajov a aplikujú štatistický test pravdepodobnosti na výskyty generované odhadovaným modelom na priradenie skóre anomálie v testovacej inštancii. Takéto štatistické modely predpokladajú, že normálne body sa objavujú v oblastiach s vysokou pravdepodobnosťou distribúcie, čím je vysoká pravdepodobnosť výskytu, a tým aj nízke skóre anomálií. Na druhej strane sa objavujú anomálne body v oblasti s nízkou pravdepodobnosťou distribúcie a majú vysoké skóre anomálií. Takéto metódy sú účinné, ak je možné modelovať normálne prípady štatistickým rozdelením.

1.2.2 Klasifikačné metódy

Metódy založené na klasifikácii sa učia klasifikátor z označených alebo neoznačených trénovacích údajov a priraďujú skóre anomálie alebo označenie inštancií testovacích údajov. Metódy zisťovania anomálií založené na klasifikácii možno kategorizovať na metódy jednej triedy, ktoré majú jeden model pre normálnu triedu a akýkoľvek bod, ktorý tomuto modelu nevyhovuje, sa považuje za anomálny, a metódy viacerých tried, ktoré majú viacero normálnych tried, a body ktoré nezodpovedajú žiadnej z bežných tried sa považujú za anomálne (Chandola, 2017).

Autor (Vishwakarma, 2020) navrhuje použitie neurónovej siete pri detekcii anomálií, keďže štatistické postupy sú vo všeobecnosti založené na niektorých predchádzajúcich predpokladoch, ako napríklad distribúcia údajov alebo predchádzajúce znalosti o parametroch.

1.2.3 Zhlukové metódy

Algoritmy detekcie anomálií bez učiteľa vychádzajú z dvoch predpokladov o údajoch. Prvým predpokladom je, že počet normálnych dát výrazne prevyšuje anomálne dáta. Druhým predpokladom je, že samotné anomálie sú kvalitatívne odlišné od normálnych dát (Feng, 2007).

Podľa autora (Schneider, 2022) sa techniky zhlukových metód, ktoré sú využité pri detekcii anomálií, dajú kategorizovať do dvoch skupín:

- techniky, ktoré predpokladajú, že anomálie spadajú do zhlukov s malým počtom údajových bodov alebo majú nízku hustotu,
- techniky, pri ktorých sa zisťuje vzdialenosť od údajových bodov k ich najbližším ťažiskám klastrov.

V určitých prípadoch, ak samotné anomálie môžu tvoriť zhluk, predpokladá sa, že normálne body tvoria veľké a husté zhluky, pričom vznikajú anomálne body ako malé zhluky alebo zhluky s nízkou hustotou. Zatiaľ čo takéto metódy identifikujú anomálie v post-klastrovacej fáze, začínajú existovať metódy, ktoré sa zameriavajú na identifikáciu anomálie súčasne s klastrami (Chandola, 2017).

1.2.4 Metódy založené na časových rádoch

Dáta časových radov sú dôležitým viac-rozmerným dátovým typ. Sú to sekvencie zložené z hodnôt určitej fyzikálnej veličiny objektov pre rôzne časové body usporiadané v chronologickom poradí. Pri analýze údajov časových radov chceme zistiť, ako údaje časových radov súvisia v rôznych časových obdobiach. Tento vzťah sa všeobecne prejavuje ako časté vzory zmien v časových radoch a tieto zmeny môžu byť minimálne. Tento zriedkavý vzor zmeny sa nazýva anomálny vzor (Weng, 2018).

Deep Neural Network (ďalej DNN) je typ neurónovej siete obsahujúcej väčšie množstvo skrytých vrstiev. Neexistuje presné kategorizovanie sietí na základe počtu vrstiev, ale ľubovoľná architektúra, ktorá sa skladá z dvoch a viac vrstiev môže byť považovaná za DNN. Neurónové siete typu DNN sa používajú na riešenie zložitých problémov, pri ktorých je potrebné sa naučiť veľké množstvo údajov (Nasimul, 2020).

Neurónové siete sa čoraz viac využívajú na modelovanie časových radov. DNN s kompletnou doprednou väzbou ukazujú, že úspešne modelujú časové rady, napr. pri modelovaní prevádzky webových stránok (Rebala, 2019).

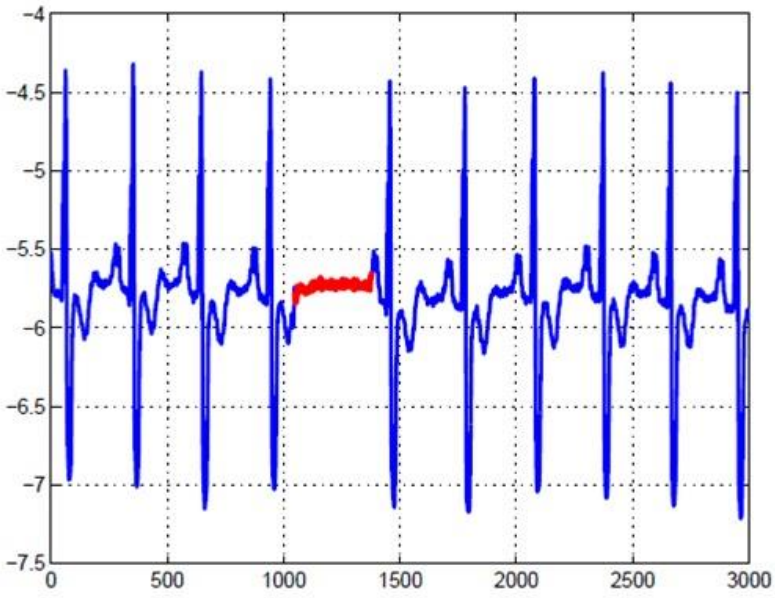
1.2.5 Metriky pre meranie algoritmov

Autor (Mehrotra, 2017) vysvetľuje, že na vyhodnotenie výkonnosti algoritmov pre detekciu anomálií sa často používajú tri metriky:

- Precision,
- Recall,
- Rank-Power – v prípade detekcie anomálií na základe hodnosti.

1.2.6 Kolektívne anomálie

Ak je súbor inštancií súvisiacich údajov anomálny vzhľadom na celý súbor údajov, označuje sa to ako kolektívna anomália (Obrázok 4). Jednotlivé inštancie údajov v kolektívnej anomálii nemusia byť samy o sebe anomáliami, ale ich výskyt spolu ako kolekcie je anomálny (Chandola, 2017).



Obrázok 4 Kolektívna anomália[4]

1.3 MODELY VÝVOJA

V 80. rokoch vznikla oblasť dolovania údajov. So vznikom tejto oblasti boli zavedené rôzne modely procesov pre efektívne dolovanie údajov. Tieto procesné modely popisujú úlohy dolovania údajov a ich aplikácie. Medzi známe modely dolovania údajov patria:

- CRISP-DM,
- ASUM-DM,
- SEMMA (Shafique, 2014).

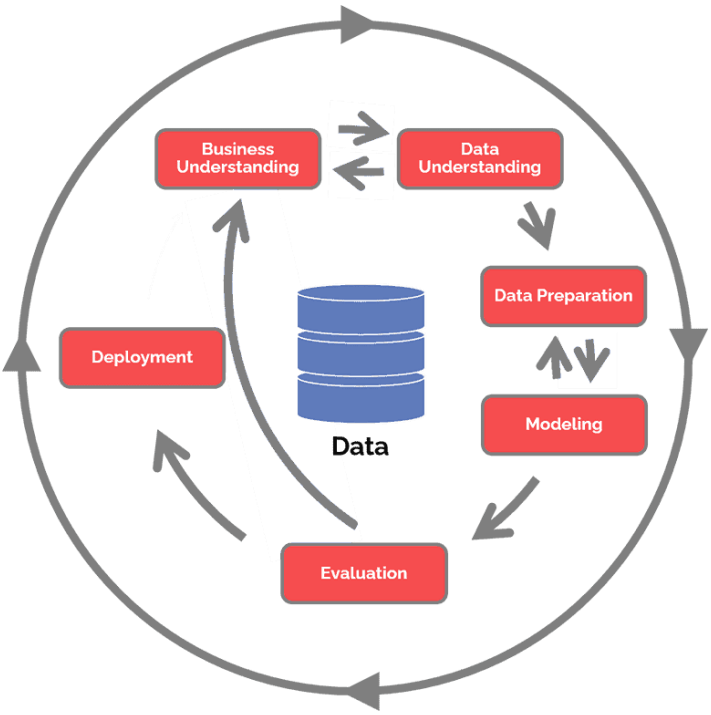
1.3.1 CRISP-DM

Cross Industry Standard Process (ďalej CRISP-DM) je procesný model pre dolovanie dát, ktorý pozostáva zo šiestich iteračných fáz (Obrázok 5):

- Porozumenie problému,

[4] Zdroj Obrázok 4: https://stats.stackexchange.com/questions/323553/difference-between-contextual-anomaly-and-collective-anomaly

- porozumenie dátam,
- príprava dát,
- modelovanie,
- evalvácia modelu,
- nasadenie modelu (Schröer, 2000).

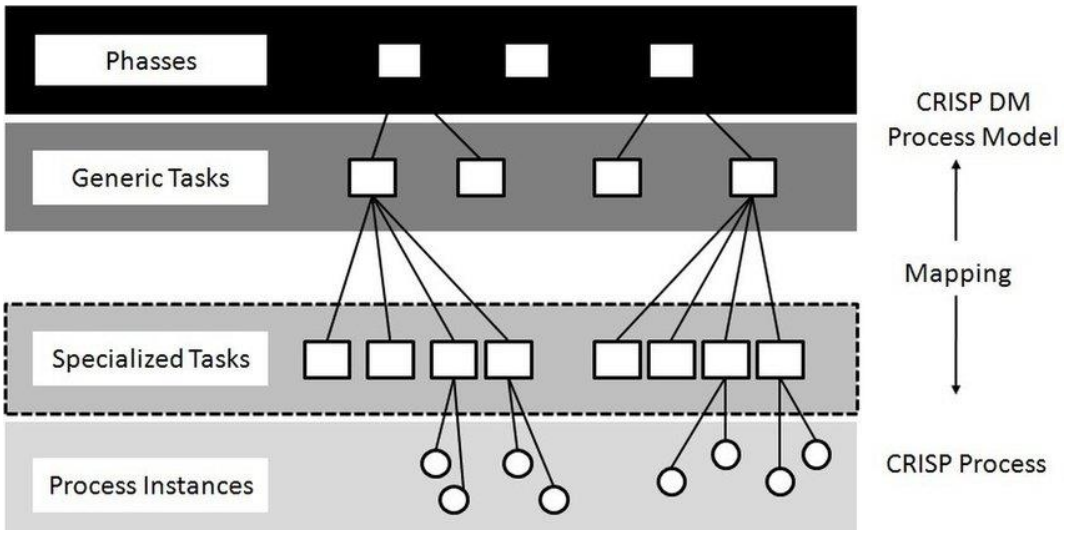


Obrázok 5 Diagram fáz modelu CRISP-DM [5]

Rozšírené opísanie metodológie CRISP-DM nájdeme u autorov (Chapman, 2000), ktorý vysvetľujú, že metodológia CRISP-DM je opísaná ako hierarchický procesný model (Obrázok 6), ktorý pozostáva zo súborov úloh opísaných v štyroch úrovniach abstrakcie (od všeobecnej po špecifickú):

- fáza,
- generická úloha,
- špecializovaná úloha,
- inštancia procesu.

[5] Zdroj Obrázok 5: https://www.datascience-pm.com/crisp-dm-2/



Obrázok 6 Štyri úrovne úloh v modeli CRISP-DM [6]

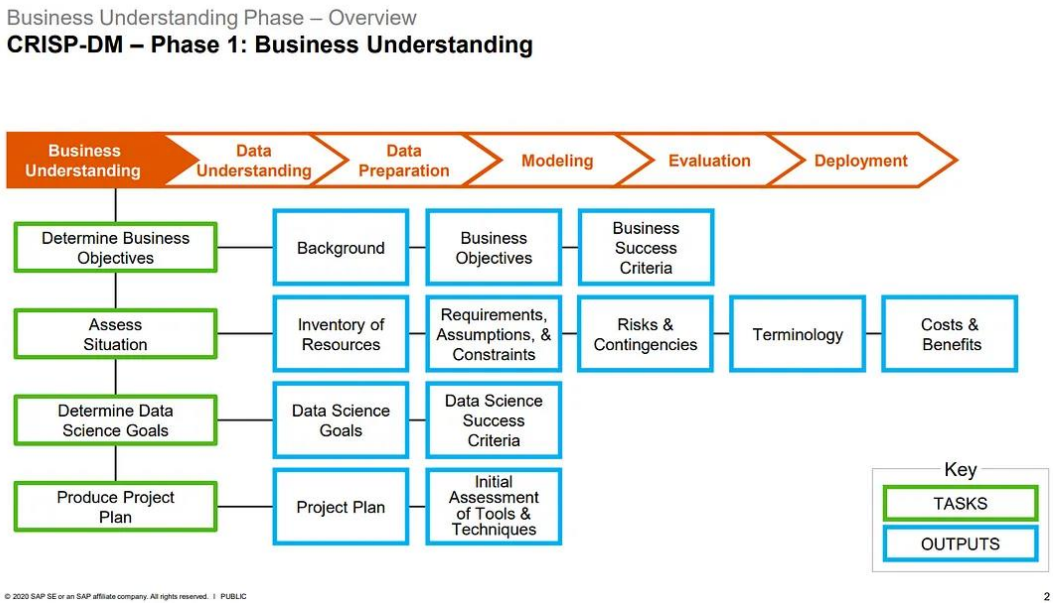
CRISP-DM je štandard procesu analýzy údajov, ktorý popisuje bežne používané prístupy na vykonávanie analýzy údajov. Tento proces je použiteľný v rôznych odvetviach priemyslu, pokiaľ je objem údajov dostatočne vysoký (Nabati, 2017).

Porozumenie problému

Porozumenie problému (Obrázok 7) poukazuje na požiadavky problému analýzy údajov z obchodného hľadiska. Kladie dôraz na pochopenie cieľov analýzy dát a dostatočné rozpoznanie aspektov problému (Nabati, 2017).

Táto počiatočná fáza sa zameriava na pochopenie cieľov a požiadaviek projektu, následné prevedenie týchto znalostí na definíciu problému dolovania údajov a predbežný plán navrhnutý na dosiahnutie cieľov (Chapman, 2000).

[6] Zdroj Obrázok 6: https://www.researchgate.net/figure/Four-level-breakdown-of-CRISP-DM-methodology-Chapman-et-al-2000_fig2_282975205

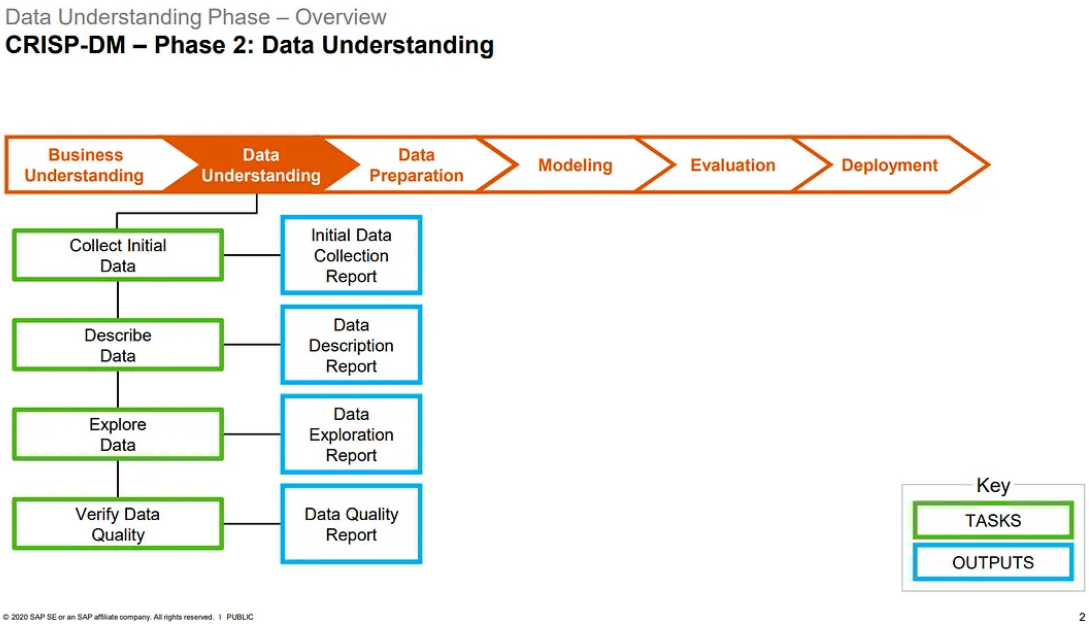


Obrázok 7 Generické a špecializované úlohy fázy Porozumenie problému [7]

Porozumenie údajov

Fáza porozumenia údajov (Obrázok 8) začína počiatočným zberom údajov a pokračuje aktivitami, ktoré umožňujú oboznámiť sa s údajmi, identifikovať problémy s kvalitou údajov, objaviť prvé pohľady na údaje a odhaliť zaujímavé podmnožiny na vytváranie hypotéz týkajúcich sa skrytých informácií (Chapman, 2000).

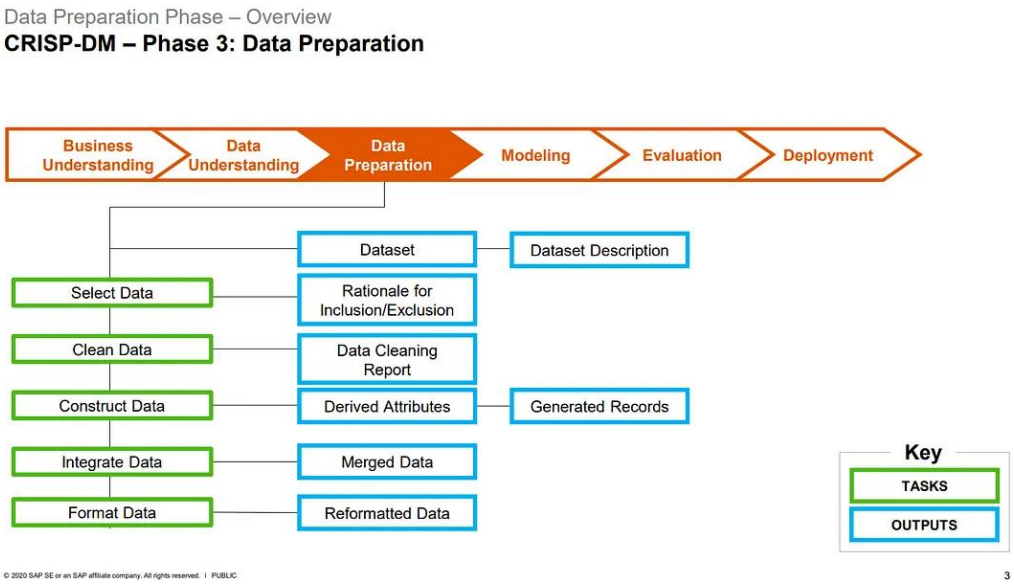
[7] Zdroj Obrázok 7: https://medium.com/analytics-vidhya/crisp-dm-phase-1-business-understanding-255b47adf90a



Obrázok 8 Generické a špecializované úlohy fázy Pochopenie údajov [8]

Príprava dát

Príprava údajov (Obrázok 9) zahŕňa všetky činnosti potrebné na zostavenie konečného súboru údajov, ktoré budú vložené do modelovacieho nástroja, z pôvodných nespracovaných údajov. Úlohy prípravy údajov sa pravdepodobne vykonajú viackrát a nie v akomkoľvek predpísanom poradí. Úlohy zahŕňajú výber tabuliek, záznamov a atribútov, ako aj transformáciu a čistenie údajov pre modelovacie nástroje (Chapman, 2000).

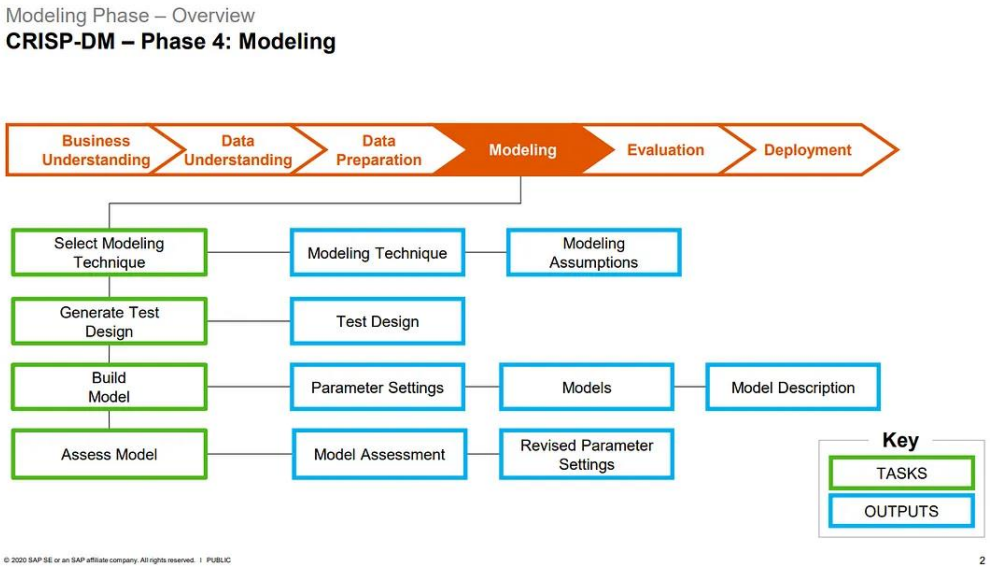


[8] Zdroj Obrázok 8: https://medium.com/analytics-vidhya/crisp-dm-phase-2-data-understanding-b4d627ba6b45

Obrázok 9 Generické a špecializované úlohy fázy Príprava dát [9]

Modelovanie

V tejto fáze sa vyberú a aplikujú rôzne techniky modelovania (Obrázok 10) a ich parametre sa kalibrujú na optimálne hodnoty. Typicky existuje niekoľko techník pre rovnaký typ problému dolovania údajov. Niektoré techniky majú špecifické požiadavky na formu údajov, a preto je často potrebné vrátiť sa do fázy prípravy údajov (Chapman, 2000).



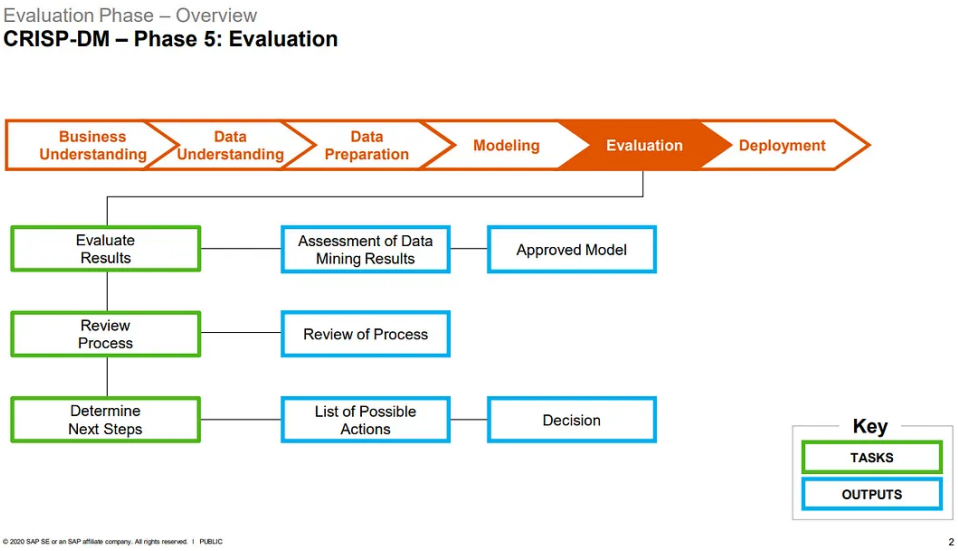
Obrázok 10 Generické a špecializované úlohy fázy Modelovanie [10]

Evalvácia modelu

Pred konečným nasadením modelu (Obrázok 11) je dôležité dôkladne vyhodnotiť daný model a skontrolovať kroky, ktoré viedli k jeho vytvoreniu, aby bolo zaistené, že model správne dosahuje obchodné ciele. Hlavným cieľom je zistiť, či existuje nevyriešený obchodný problém, ktorý môže mať negatívny vplyv na výsledok projektu. Na konci tejto fázy by sa malo dospieť k rozhodnutiu o použití výsledkov dolovania údajov (Chapman, 2000).

[9] Zdroj Obrázok 9: https://medium.com/analytics-vidhya/crisp-dm-phase-3-data-preparation-faf5ee8dc38e

[10] Zdroj Obrázok 10: https://medium.com/analytics-vidhya/crisp-dm-phase-4-modeling-phase-b81f2580ff3



Obrázok 11 Generické a špecializované úlohy fázy Evalvácia modelu [11]

Nasadenie modelu

Vytvorením modelu sa vo všeobecnosti projekt nekončí. Aj keď účelom trénovania modelu je zvýšenie jeho vedomostí o údajoch, získané poznatky bude potrebné usporiadať do takej formy v ktorej ich je zákazník schopný použiť. To často zahŕňa aplikáciu „živých“ modelov v rámci rozhodovacích procesov organizácie (Chapman, 2000).

1.3.2 ASUM-DM

ASUM-DM je rozšírenie metodológie CRISP-DM pre projekty dolovania údajov a prediktívnej analýzy. Úlohy modelu CRISP-DM a cyklus dolovania údajov boli zachované a úplne akceptované, ale vylepšené vo fáze nasadenia modelu, kde má CRISP-DM slabé miesto. Okrem toho boli pridané úlohy v oblasti infraštruktúry/prevádzky a projektového manažmentu a metóda bola rozšírená o niekoľko užitočných šablón a pokynov (IBM, 2015).

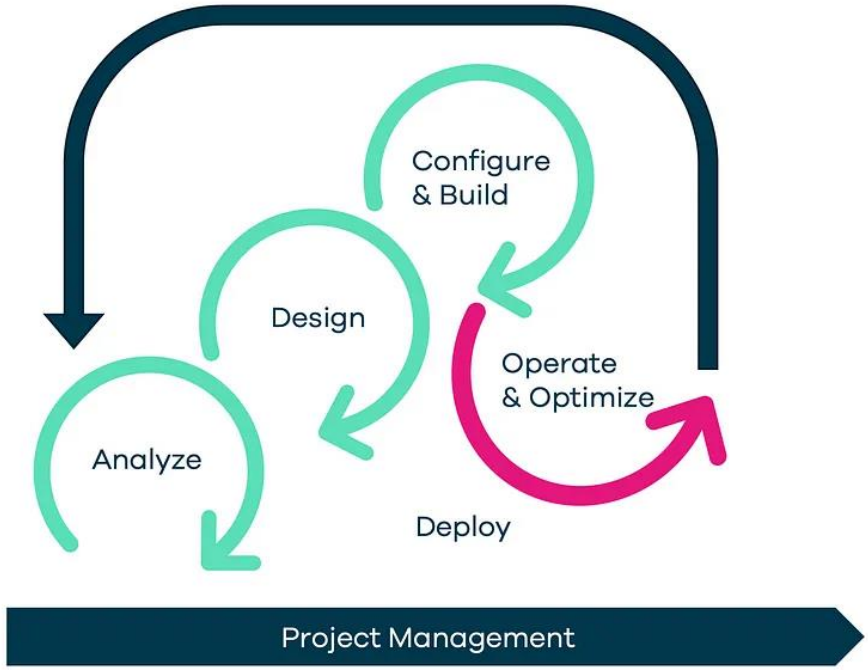
ASUM-DM je iteratívny proces (Obrázok 12) na implementáciu dolovania údajov alebo prediktívnej analýzy. ASUM využíva implementáciu agilných metodík a tzv. tradičné princípy softvérového inžinierstva na dosiahnutie cieľov a poskytnutie optimálneho výsledku.

Delí sa na fázy:

- analýza,

[11] Zdroj Obrázok 11: https://medium.com/analytics-vidhya/crisp-dm-phase-5-evaluation-phase-d7bb3c75220a

- dizajn,
- konfigurácia a zostavenie,
- nasadenie,
- prevádzka a optimalizácia (Fois, 2020).



Obrázok 12 Fázy metodológie ASUM-DM [12]

1.4 PRINCÍPY VÝVOJA

Pri používaní vývojových modelov je potrebné písať zdrojový kód vo viacerých programovacích a značkovacích jazykoch. V snahe o písanie vysokokvalitného kódu vývojári prijali rôzne architektonické a dizajnové princípy. Pochopenie a uplatnenie týchto princípov môže výrazne zlepšiť čitateľnosť, udržateľnosť a výkon našej kódovej základne (Kumar, 2022).

V našej predošlej práci sme vysvetlili princípy DRY, KISS a SOLID.

1.5 MLOPS

Autor (Kreuzberger, 2023) definuje MLOps (Machine Learning Operations) ako paradigmu, ktorá zahŕňa osvedčené postupy, súbory konceptov, ako aj kultúru vývoja,

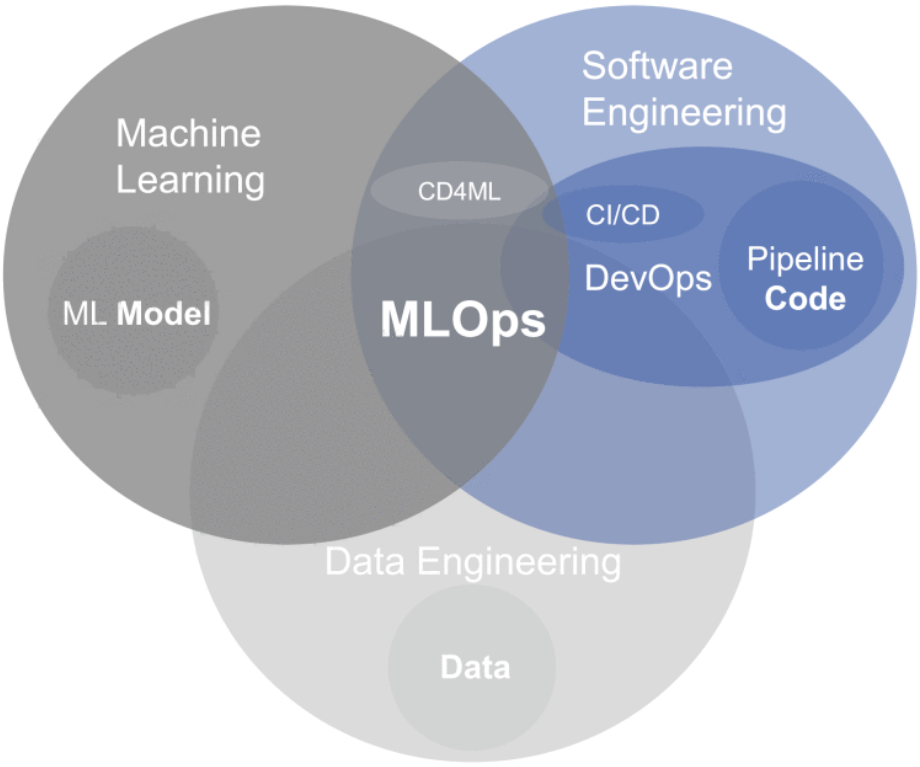
[12] Zdroj Obrázok 12: https://medium.com/data-smart-services/standard-compliant-data-projects-50f75bd12cdd

pokiaľ ide o komplexnú koncepciu, implementáciu, monitorovanie, nasadenie a škálovateľnosť produktov strojového učenia. Kombinuje tri disciplíny (Obrázok 13):

- strojové učenie,
- softvérové inžinierstvo (najmä DevOps),
- dátové inžinierstvo.

MLOps sa zameriava na produkciu systémov strojového učenia preklenutím priepasti medzi vývojom a prevádzkou. Cieľom je uľahčiť vývoj produktov strojového učenia využitím týchto princípov:

- automatizácia CI/CD,
- orchestrácia pracovného toku,
- reprodukovateľnosť,
- verziovanie údajov, modelu a kódu,
- nepretržité monitorovanie (Kreuzberger, 2023).



Obrázok 13 Disciplíny zahrnuté v MLOps [13]

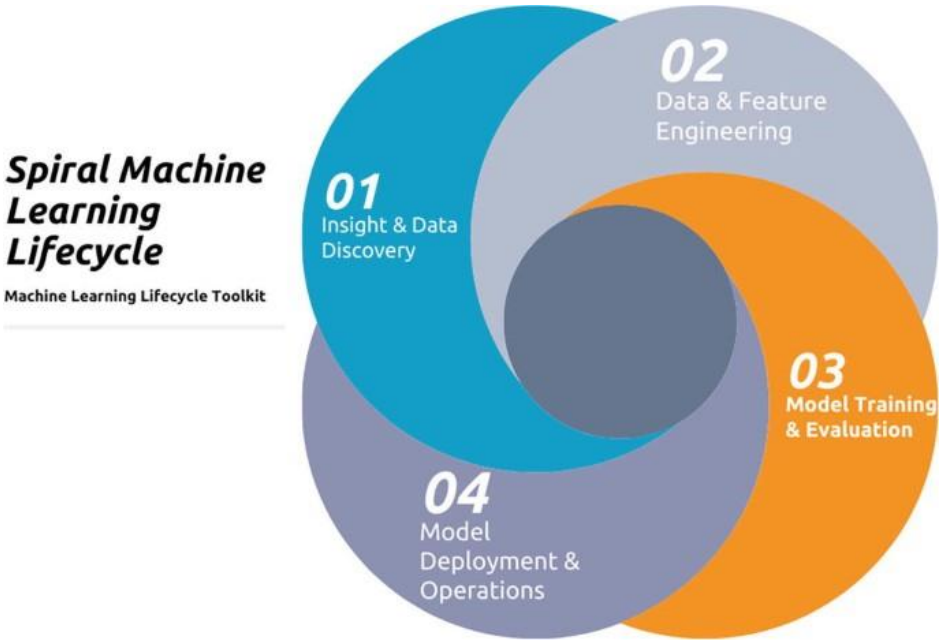
Prehľadnejšie vysvetlenie, prečo MLOps je dôležitou súčasťou vývoja strojového učenia, nám ponúka autor Alla v jeho knihe *Beginning MLOps with MLFlow* (2021). Opisuje, že MLOps výrazne uľahčuje nasadenie a údržbu riešení strojového učenia automatizáciou väčšiny ťažkých častí, masívne urýchľuje vývoj a procesy údržby. Plná automatizácia ponúka rýchlosť pri nasadení nových modelov, ale zachováva vysokú úroveň výkonu a môže ponúkať zlepšenie, keďže vývojári môžu nasadiť novšie a lepšie modelové architektúry.

Životný cyklus MLOps sa všeobecne rozdeľuje na fázy projektu:

1. prehľad a objavovanie údajov,
2. dátové a funkčné inžinierstvo,
3. trénovanie a evalvácia modelu,
4. nasadenie a orchestrácia modelu.

Životný cyklus je typu špirály (Obrázok 14), pretože každá z týchto fáz môže byť spätnou väzbou pre predchádzajúce fázy (Sorvisto, 2023).

[13] Zdroj Obrázok 13: https://ieeexplore.ieee.org/document/10081336



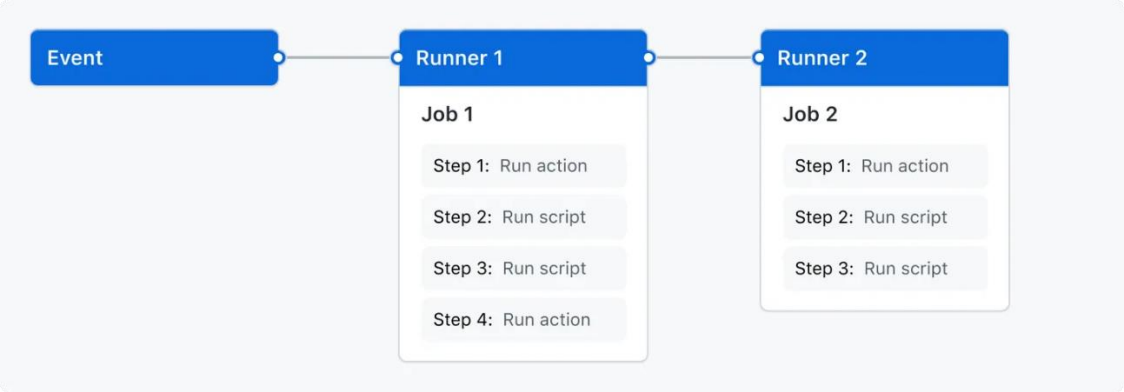
Obrázok 14 Špirálový životný cyklus MLOps [14]

1.5.1 GitHub Actions

Podľa GitHub dokumentácie (2024) je GitHub Actions platforma kontinuálnej integrácie a kontinuálneho doručovania (CI/CD). Umožňuje automatizovať zostavovanie, testovanie a nasadzovanie aplikácií pomocou vytvárania tzv. pracovných postupov (Workflow), ktoré zostavujú a testujú každý pull request do repozitára, alebo nasadzujú zlúčené pull requesty do produkcie.

Workflow (Obrázok 15) je konfigurovateľný automatizovaný proces, ktorý spustí jednu alebo viac úloh. Pracovný postup je definovaný v súbore typu YAML, ktorý sa nachádza v repozitári aplikácie a spustí sa pri spustení udalosti v repozitári, alebo môžu byť spustený manuálne, či podľa definovaného plánu (GitHub, 2024).

[14] Zdroj Obrázok 14: https://link.springer.com/chapter/10.1007/978-1-4842-9642-4_1



Obrázok 15 Architektúra pre Workflow [15]

Udalosť (Event) je špecifická aktivita v repozitári, ktorá spúšťa Workflow. Aktivita môže napríklad pochádzať z GitHubu, keď niekto vytvorí pull request alebo odošle commit do repozitára. Workflow je možné spustiť aj podľa plánu odoslaním do REST API alebo manuálne (GitHub, 2024).

Úloha (Job) je súbor krokov vo Workflowe, ktorý sa vykonáva na rovnakom Runnerovi (bežcovi). Každý krok je buď shell skript, ktorý sa vykoná, alebo akcia, ktorá sa spustí. Kroky sa vykonávajú v poradí a sú na sebe závislé. Keďže každý krok sa vykonáva na tom istom Runnerovi, je možné zdieľať údaje z jedného kroku do druhého (GitHub, 2024).

Runner je server, ktorý spúšťa Workflow, keď je spustený. Každý Runner môže vykonávať len jednu úlohu naraz. GitHub poskytuje Runnera Ubuntu Linux, Microsoft Windows a macOS na spustenie Workflowu. Každé spustenie pracovného toku sa vykoná na novo poskytnutom virtuálnom stroji (GitHub, 2024).

[15] Zdroj Obrázok 15: https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions

2 CIELE ZÁVEREČNEJ PRÁCE

Hlavným cieľom tejto záverečnej práce, overiť potenciál metód detekcie únikov využívajúcich metódy strojového učenia a hĺbkového učenia použitím vygenerovaných dát o toku tekutiny v potrubí.

Pre dosiahnutie tohto cieľu je potrebné analyzovať dostupné architektúry, metódy a technológie, ktoré sa využívajú na detekciu anomálií a metodiky a je možné ich použiť pri vývoji modelu strojového učenia.

Nasledovným cieľom je popísať jednotlivé fázy vývojového modelu použitého pri vývoji modelu strojového učenia na detekciu anomálií v údajoch získaných zo simulátora tekutiny v potrubí. Výsledky získané z modelu strojového učenia je potrebné sumarizovať a porovnať.

Podciele alebo čiastkové ciele:

- analyzovať technológie, ktoré sú používané na detekciu anomálií,
- predstaviť metodiky používané pri vývoji umelej inteligencie,
- opísať spôsoby nasadenia a automatizácie,
- zvoliť vhodné nástroje na analýzu a prípravu dát,
- implementovať paradigmu MLOps pri cykle vývoja modelu,
- vyvinúť model na predikciu únikov v potrubí s tekutinou,
- otestovať funkčnosť a kvalitu modelu,
- vytvoriť aplikáciu, ktorá môže používať model v produkčnom prostredí
- vytvoriť automatizovaný proces nasadenia modelu.

3 VÝVOJ NEURÓNOVEJ SIETE

V tejto práci sme sa rozhodli použiť metodiku CRISP-DM na analýzu a vývoj modelu predikcie únikov v potrubí. Postupovali sme v súlade s jednotlivými krokmi metodiky.

3.1 POROZUMENIE PROBLÉMU

Prvým krokom bolo porozumenie problematike. V tejto fáze sme analyzovali klientove požiadavky, ktoré model musí spĺňať, stanovili sme jasné ciele a podciele a vytvorili sme projektový plán pre nadchádzajúce fázy projektu.

Určenie obchodných cieľov

Počas rozhovoru s klientom sme porozumeli, že náš klient potrebuje vytvoriť neurónovú sieť, ktorá dokáže rozoznať, či v potrubí s tekutinou nastáva únik alebo nie. Táto sieť musí pracovať v prostredí, kde môže byť používané čerpadlo, ktoré vpúšťa tekutinu do potrubia. Toto čerpadlo je prepínané podľa neznámych požiadaviek. Potrubie, ktoré používa čerpadlo, môže obsahovať iba dva senzory na začiatku pre tlak a prietok a dva senzory na konci, tiež pre tlak a prietok. K dispozícii nám klient poskytol simulátor dát, ktorý potrebujeme na generovanie dát.

Následne sme s klientom usúdili, že model musí spĺňať dve kritéria. Prvým kritériom bolo, že model musí byť schopný určiť, že únik nastal, ale aj nenastal. Druhé kritérium spočívalo v tom, že tento model dokáže rozpoznanie úniku vykonať aj v prípadoch, kedy je používané čerpadlo potrubia.

Posúdenie situácie

Na základe kritérií, ktoré boli stanovené počas rozhovoru s klientom sme určili nasledovné požiadavky pre projekt:

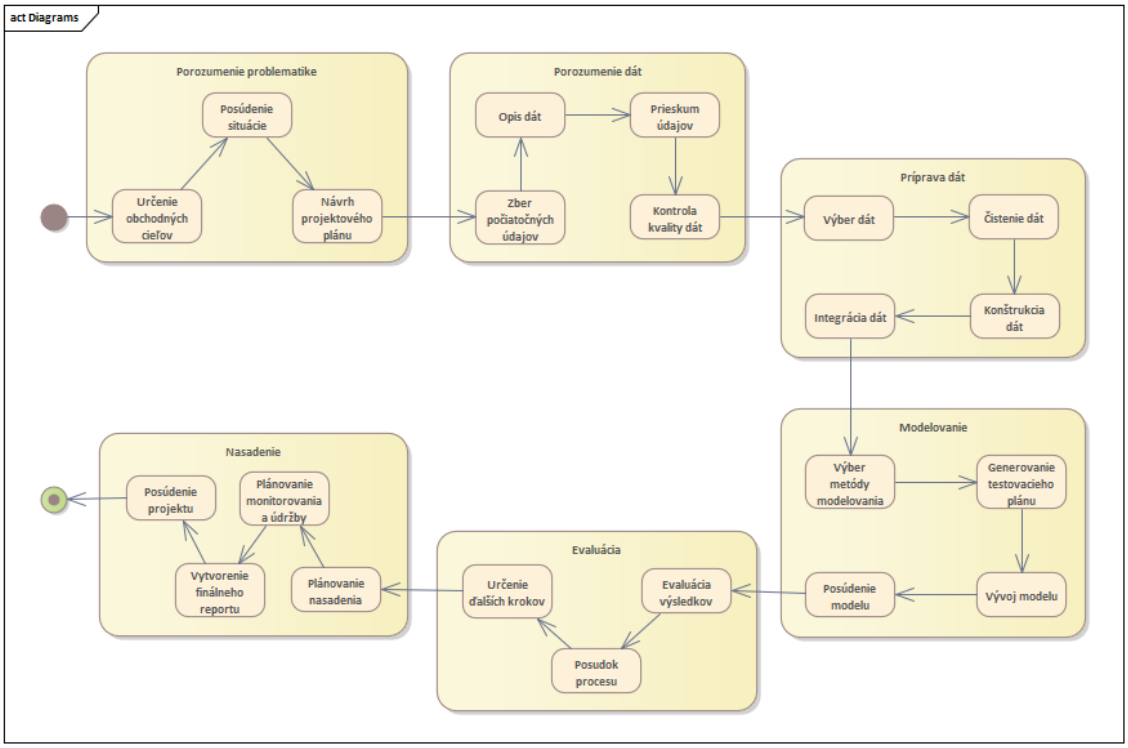
- vytvorenie modelu s použitím neurónovej siete,
- model musí byť schopný rozoznať, či v potrubí s tekutinou nastal únik na základe prietoku a tlaku v potrubí,
- model musí byť schopný rozoznať únik v stave používania čerpadla, ale aj v stave kedy sa čerpadlo nepoužíva.

Tento projekt taktiež obsahuje obmedzenie z hľadiska dát, ktoré sme schopní použiť. K dispozícii budú iba štyri senzory (dva pre tlak a dva pre prietok) na začiatku a na konci potrubia.

Tento projekt dokáže vytvoriť benefity pre klienta pri čerpaní potrubia s tekutinou, pokiaľ model bude schopný dostatočne rýchlo a správne určiť, že v potrubí s tekutinou nastáva únik. V takomto prípade strata tekutiny z potrubia bude minimálna a klient predíde potenciálnym stratám cenných zdrojov.

Návrh projektového plánu

Plán projektu bude pozostávať zo šiestich fáz, ktoré korešpondujú k jednotlivým fázam metodiky CRISP-DM. V pláne sme sa sústredili na dodržanie jednotlivých úloh, ktoré sú súčasťou každej fázy v metodike. Tieto úlohy môžeme vidieť vo vytvorenom diagrame aktivít (Obrázok 16).



Obrázok 16 Diagram aktivít projektu

3.2 POROZUMENIE DÁT

Počas fázy porozumenia dátam sme vygenerovali dáta, ktoré sme následne popísali a vizualizovali, aby sme objavili nedostatky v dátach.

Zber počiatočných údajov

Na zbieranie údajov sme použili simulátor, ktorý nám poskytol klient. V tomto simulátore sme nakonfigurovali štyri senzory na základe požiadaviek z predošlej fázy. Následne sme vygenerovali údaje v štyroch krokoch:

1. generovanie dát, bez použitia čerpadla a zapnutia úniku,
2. generovanie dát s prepínaním čerpadla,

3. generovanie dát so zapnutím úniku tekutiny,
4. generovanie dát s prepínaním čerpadla a zapnutím úniku tekutiny.

Opis údajov

Na obrázku 17 môžeme vidieť údaje, ktoré sme vygenerovali pomocou simulátora. Tieto údaje sú štruktúrované nasledovne:

- *id* – unikátny identifikátor senzora,
- *size* – nepotrebný údaj vygenerovaný simulátorom,
- *header_position* – nepotrebný údaj vygenerovaný simulátorom,
- *name* – názov senzora,
- *value* – hodnota senzora,
- *flag* – nepotrebný údaj vygenerovaný simulátorom,
- *time* – časová známka, počas ktorej senzor zachytil hodnotu.

id	size	header_position	name	value	flag	time
1	8	12	ActTime	5.047375e+183	0	2024-04-07 22:26:42.254000
2	8	25	ActTime2	5.047375e+183	0	2024-04-07 22:26:42.255000
1	8	12	ActTime	5.047375e+183	0	2024-04-07 22:26:42.269000
2	8	25	ActTime2	5.047375e+183	0	2024-04-07 22:26:42.270000
3	8	32	F1	0.000000e+00	0	2009-04-22 21:25:53.537000

Obrázok 17 Dáta vygenerované simulátorom

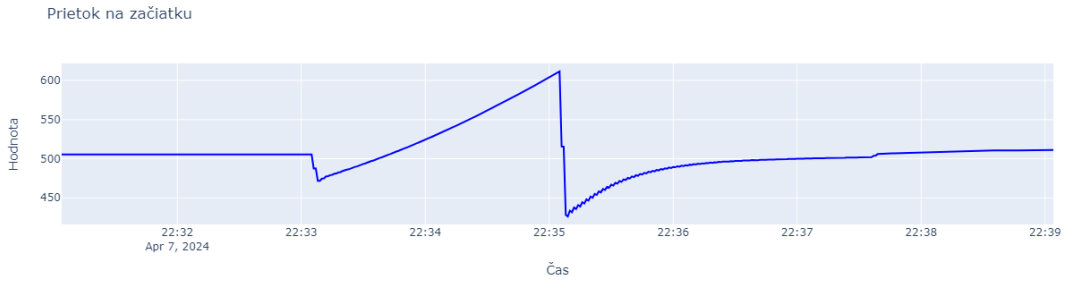
Dáta sme generovali do viacerých súborov, pre jednoduchšiu manipuláciu a rozpoznanie zdroju dát. Zozbierané množstvo dát je možné pozorovať v tabuľke 1.

Tabuľka 1 Množstvo vygenerovaných dát

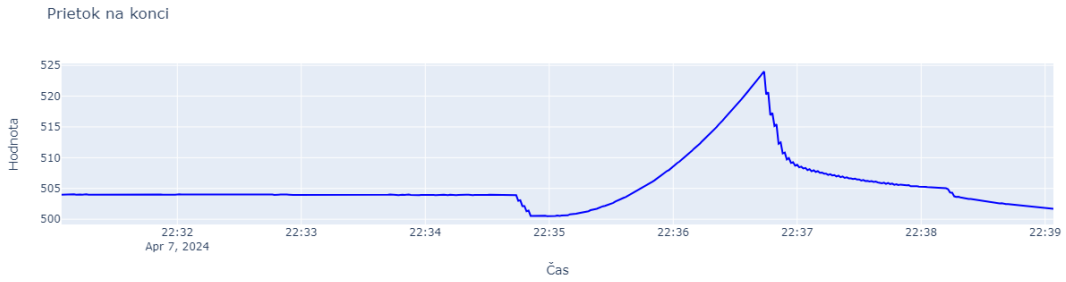
Spustenie úniku	Prepínanie čerpadla	Počet riadkov
Áno	Nie	2447
Nie	Nie	113711
Nie	Áno	45669

Prieskum údajov

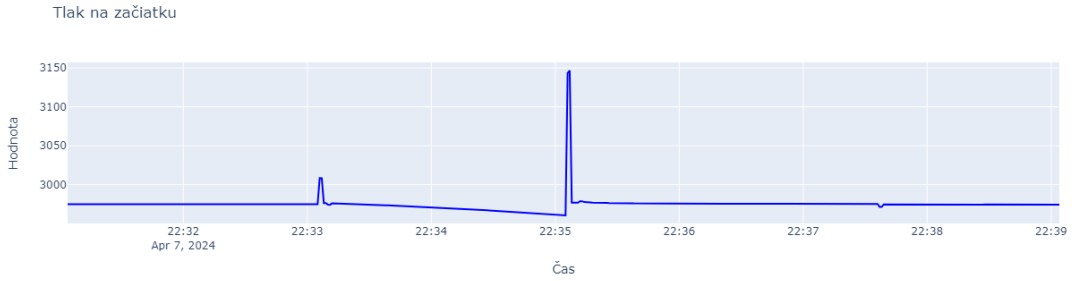
Údaje, ktoré nám vygeneroval simulátor, sme vizualizovali vo forme bodových grafov, aby sme boli schopní pochopiť obdobný priebeh a zmenu údajov, ale taktiež, aby bolo možné vidieť vzory, ktoré sa musí model naučiť. Na obrázkoch 18, 19, 20, 21 je možné pozorovať tieto bodové grafy.



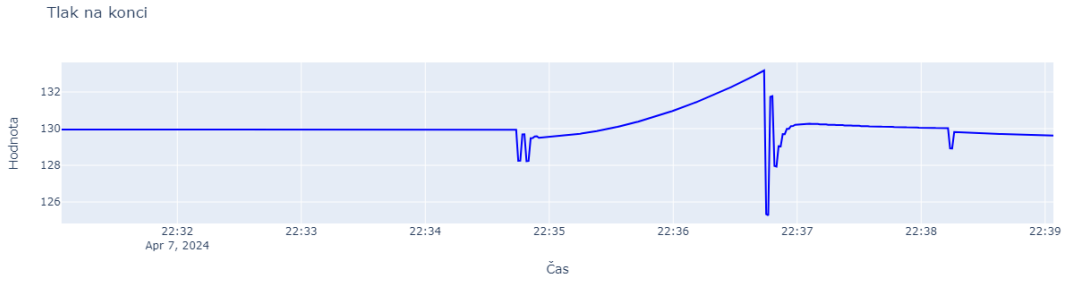
Obrázok 18 Prietok na začiatku potrubia



Obrázok 19 Prietok na konci potrubia



Obrázok 20 Tlak na začiatku potrubia



Obrázok 21 Tlak na konci potrubia

Po preskúmaní grafov sme zistili, že premenné grafu v prípade, kedy nenastáva únik v potrubí, boli pomerne lineárne, no v prípade spustenia čerpadla alebo v prípade,

že nastal únik, tieto premenné mali výkyv hodnoty. Taktiež sme pozorovali, že v prípade únikov tieto výkyvy boli výrazne vyššie ako v prípade čerpadla.

Kontrola kvality údajov

Po vizualizácii údajov sme zistili, či sa v datasete nenachádzajú údaje, ktoré by mohli obmedziť trénovanie neurónovej siete. Jednotlivé riadky neobsahovali žiadne chýbajúce údaje.

Senzory zaznamenali nulovú hodnotu pre premenné potrubia z roku 2009. Tieto údaje sme považovali za zavádzajúce údaje, keďže by poškodili trénovanie neurónovej siete.

3.3 PRÍPRAVA DÁT

V tejto fáze sme vybrali údaje z nášho datasetu a určili, na čo boli jednotlivé skupiny dát použité. Následne sme vytvorili nové vlastnosti pre jednoduchšiu manipuláciu s dátami a vykonali sme agregačné operácie, aby sme získali formát údajov, na ktorých môžeme trénovať model.

Výber dát

Pre modelovanie a evalváciu modelu sme vytvorili štyri skupiny súborov. Prvá skupina údajov bola použitá na trénovanie modelu. Táto skupina dát neobsahovala žiadne údaje, počas ktorých by nastal únik.

Jedna skupina údajov bola určená na evalváciu modelu. Tieto údaje sme použili na určenie hranice rekonštrukčnej chyby, ktorú model používa na vyhodnotenie, či nastane únik alebo nie.

Posledné dve skupiny sme použili na testovanie. Údaje sme použili na zistenie, či model dokáže určiť, že únik nastal.

Čistenie dát

Pre zjednodušenie načítania viacerých dátových súborov sme vytvorili nasledovnú funkciu (Kód 1), ktorá dynamicky načíta všetky súbory z priečinku. Túto logiku sme následne používali aj pri načítavaní modelov, ktoré sme potrebovali evalvovať.

```
def loadDataFiles(subdirectory):
    return [
        pandas.read_csv(f'data-files/{subdirectory}/{file}')
        for file in os.listdir(f'data-files/{subdirectory}')
        if not file.startswith('.')
    ]
```

Kód 1 Funkcia na dynamické načítanie súborov

Z našich dátových súborov sme vybrali iba údaje, ktoré obsahovali hodnoty zo senzorov, ktoré sme nakonfigurovali v simulátore. Vynechali sme senzory, ktoré simulátor vygeneroval sám pre senzory *ActTime, ActTime2* a *ActTime3,* keďže tieto premenné nie sú relevantné pre náš model a dokázali by negatívne ovplyvniť trénovanie neurónovej siete (Kód 2).

Týmto spôsobom nám ostali iba záznamy potrebných senzorov. Následne sme vymazali prvý záznam každého z týchto senzorov (Kód 2). Dôvod bol ten, že simulátor vždy vygeneroval iniciálny záznam, ktorý bol z roku 2009 a jeho hodnota bola nulová.

Druhou časťou čistenia dát bolo ponechanie iba potrebných stĺpcov v tabuľke. Tieto stĺpce boli:

- name – názov, podľa ktorého vieme určiť pôvod daného záznamu,
- value – hodnota, ktorú budeme vkladať do modelu,
- time – časová známka, ktorú potrebujeme pre chronologické zoradenie záznamov.

```
def transformDataframe(dataframe):
    dataframe = dataframe[dataframe['name'].isin(['F1', 'F2', 'P1', 'P2'])]
    dataframe = dataframe[dataframe['time'] != '2009-04-22 21:25:53.537000']
    dataframe = dataframe[dataframe['time'] != '2009-04-22 21:25:53.537']

    return dataframe[['name', 'value', 'time']]
```

Kód 2 Funkcia na vymazanie nepotrebných záznamov

Integrácia dát

Nasledujúcim krokom pre nás bolo pripravenie dát do formy, ktorú je možné použiť pri trénovaní modelu. Agregovali sme dáta do menších časových rádov, ktoré obsahujú záznamy pre 100 sekúnd (Kód 3).

```
def getFramesFromData(dataframe, frame_row_count=100):
    F1 = dataframe[dataframe['name'] == 'F1'].sort_values('time')
    F2 = dataframe[dataframe['name'] == 'F2'].sort_values('time')
    P1 = dataframe[dataframe['name'] == 'P1'].sort_values('time')
    P2 = dataframe[dataframe['name'] == 'P2'].sort_values('time')

    frames = []

    for i in range(0, len(F1) - frame_row_count + 1):
        frame = pandas.concat([
            F1[i: i + frame_row_count],
            F2[i: i + frame_row_count],
            P1[i: i + frame_row_count],
            P2[i: i + frame_row_count]
        ])

        frames.append(frame)

    return frames
```

Kód 3 Funkcia na rozdelenie údajov do 100-sekundových časových rádov

Po rozdelení dát do časových rádov sme extrahovali jednotlivé body a časové známky z týchto časových rádov. Tieto body sú potrebné pre model, ako vstup a časové známky sú potrebné pre evalvovanie časovej odozvy, ktorá modelu trvala, kým zistil, že nastal únik. Tieto extrahované dáta sme uložili do jednotlivých súborov. Vytvorili sme nasledovné 4 súbory:

- trénovací súbor – tento súbor obsahoval iba údaje z obdobia, v ktorom potrubie nemalo únik a použili sme ho na trénovanie modelu,
- testovací súbor bez úniku – tento súbor neobsahoval údaje s únikom a použili sme ho na evalváciu modelu a hranice rekonštrukčnej chyby,
- dva testovacie súbory s únikom – obsahovali údaje s únikom v potrubí a tieto súbory sme využili na testovanie, či model spĺňa požiadavky dané klientom.

Ako je vidieť na obrázku 22, výsledné súbory obsahovali tabuľkové údaje pre nasledovné premenné:

- 100 hodnôt senzoru prietoku na začiatku potrubia (*F1-1* až *F1-100*),
- 100 hodnôt senzoru tlaku na začiatku potrubia (*P1-1* až *P1-100*),
- 100 hodnôt senzoru prietoku na konci potrubia (*F2-1* až *F2-100*),
- 100 hodnôt senzoru tlaku na konci potrubia (*P2-1* až *P2-100*),
- časovú známku posledného posledného záznamu predošlých hodnôt (*timestamp*).

F1-1	F1-2	F1-3	...	P2-99	P2-100	timestamp
507.510123	507.514126	507.499791	...	129.947881	129.947820	2024-02-17 10:21:28
507.514126	507.499791	507.503306	...	129.947820	129.947822	2024-02-17 10:21:29
507.499791	507.503306	507.507160	...	129.947822	129.947731	2024-02-17 10:21:30
507.503306	507.507160	507.507160	...	129.947731	129.947725	2024-02-17 10:21:31
507.507160	507.507160	507.540618	...	129.947725	129.947715	2024-02-17 10:21:32
...	...	...	...	...	...	...
582.482292	582.404708	582.478538	...	139.739054	139.739054	2024-02-17 13:00:14
582.404708	582.478538	582.454817	...	139.739054	139.739107	2024-02-17 13:00:15
582.478538	582.454817	582.454445	...	139.739107	139.739007	2024-02-17 13:00:16
582.454817	582.454445	582.424069	...	139.739007	139.738959	2024-02-17 13:00:17
582.454445	582.424069	582.452635	...	139.738959	139.738959	2024-02-17 13:00:17

Obrázok 22 Výsledný súbor obsahujúci údaje na trénovanie

Posledným krokom v príprave dát bolo validovanie týchto súborov (Kód 4), či neobsahujú nečíselné alebo nulové hodnoty. V takomto prípade sme vedeli, že v algoritme prípravy dát nastala chyba a museli sme opraviť náš kód.

```
def validateDataFile(filename):
    dataframe = pandas.read_csv(f'./python/csv/output/{filename}.csv')

    has_nan_values = dataframe.isna().any().any()
    has_zero_values = (dataframe == 0).any().any()

    if has_nan_values:
        raise Exception(f'CSV file {filename}.csv contains NaN values!')
    if has_zero_values:
        raise Exception(f'CSV file {filename}.csv contains Zero values!')
```

Kód 4 Validačná funkcia súborov

3.4 MODELOVANIE

Fázu modelovania sme rozdelili do 2 častí. V prvej časti sme model trénovali na dátach, v ktorých sa nenachádzala činnosť čerpadla a v druhej časti sme použili dáta, v ktorých sa čerpadlo prepínalo a dáta, v ktorých sa neprepínalo.

Výber metódy modelovania

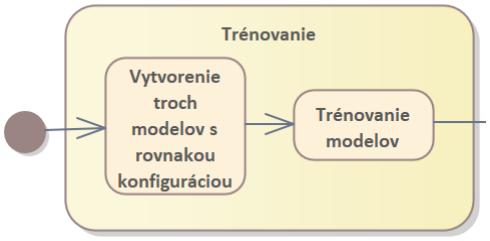
Na základe požiadavky od klienta bola zvolená neurónová sieť ako model strojového učenia. Keďže naše údaje neobsahovali žiadnu cieľovú premennú, tak sme

zvolili neurónovú sieť typu Autoencoder. Táto sieť nám umožní naučiť model rôznym vzorom, ktoré sa nachádzajú v grafoch premenných zo senzorov. V prípade, že daný vzor neurónová sieť nerozpozná, bude model schopný určiť, že v danom čase pravdepodobne nastáva únik.

Generovanie testovacieho návrhu

Klasický spôsob, akým Autoencoder vyhodnocuje správnosť predikcie, je metrika straty. Keďže potrebujeme, aby náš model penalizoval vysokú chybovosť, rozhodli sme sa zvoliť Strednú Kvadratickú Chybu (Mean Squared Error) ako metriku straty.

Navrhli sme jednoduchý plán trénovania (Obrázok 23), ktorý spočíval z vytvorenia troch rovnakých modelov a ich následné trénovanie. Týmto spôsobom môžeme porovnávať tri modely s rovnakou konfiguráciou a po evalvácii sa vyberie najlepší model.

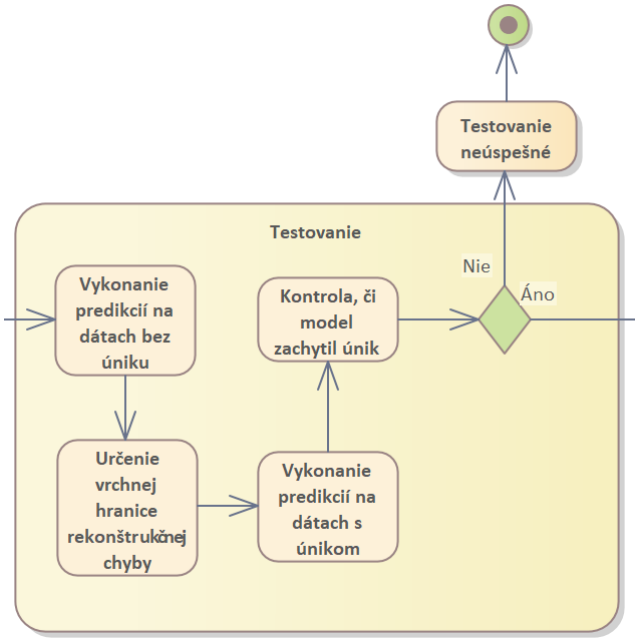


Obrázok 23 Diagram aktivít trénovania modelu

Počas testovania (Obrázok 24) sa pre každý variant modelu vygenerovali predikcie na dátach bez úniku. Tieto predikcie boli použité na výpočet rekonštrukčnej chyby, ktoré nám po dosadení do vzorca určili vrchnú hranicu.

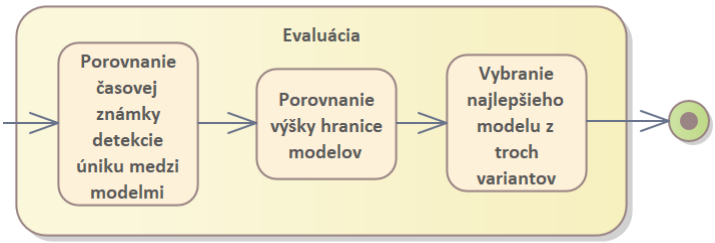
$$Threshold = 2 \times max(Reconstruction\ Error) - \min(Reconstruction\ Error)$$

Keď tieto modely mali určené vrchné hranice rekonštrukčnej chyby, tak na nich boli predikované dáta s únikom. Tieto modely využili hranicu na určenie, či únik nastal alebo nenastal. Nasledovala kontrola, či tento variant modelu únik zachytil. V prípade, že žiadny variant kontrolou neprešiel, testovanie sa neúspešne skončilo a bolo potrebné sa vrátiť do jednej z predošlých fáz. Pokiaľ bol objavený aspoň jeden model, ktorý únik zachytil, tak testovanie sa skončilo a prešlo sa do fázy evalvácie.



Obrázok 24 Diagram aktivít testovanie modelu

Plán evalvácie (Obrázok 25) pozostával z porovnania funkčných variantov na základe časovej známky zachytenia úniku a výšky hranice. Podľa týchto dvoch premenných bol určený najlepší model, ktorý predstavoval neurónovú sieť schopnú predikovať únik v najmenšom čase a s najmenšou hranicou rekonštrukčnej chyby.



Obrázok 25 Diagram aktivít evalvácie modelu

Vývoj modelu

Po vytvorení testovacieho návrhu sme vytvorili kód, ktorý nám umožnil vykonávať jednotlivé akcie z diagramov (Obrázky 23, 24, 25).

Na modelovanie neurónovej siete sme vytvorili funkciu (Kód 5) s parametrami, ktoré sme mohli meniť. Týmto sa nám uľahčila práca a vedeli sme trénovať viac modelov za sebou. Neurónovú sieť typu Autoencoder sme vytvorili prepojením dvoch dopredných sietí, ktoré zohrávali rolu komponentov Encoder a Decoder. Výstupná vrstva Encoderu a vstupná vrstva Decoderu spolu vytvárajú latentný priestor.

```
def trainModel(train_x, train_y, epochs = 20, batch = 4, neurons = 512):
    model = Sequential([
        # Encoder
        Sequential([
            Dense(400, input_shape = (400,), activation = 'relu'),

            Dense(neurons, activation = 'relu'),
        ]),
        # Decoder
        Sequential([
            Dense(neurons, input_shape = (neurons,), activation = 'relu'),

            Dense(400, activation = 'linear'),
        ]),
    ])

    model.compile(optimizer = 'adam', loss = 'mean_squared_error')

    callbacks = [
        EarlyStopping(
            monitor = 'loss', patience = 5,
            restore_best_weights = True
        )
    ]

    model.fit(
        train_x, train_y,
        epochs = epochs, batch_size = batch, shuffle = True,
        callbacks = callbacks
    )

    return model
```

Kód 5 Funkcia na trénovanie modelu podľa parametrov

Použili sme aktivačnú funkciu ReLU, keďže používame konvolučnú neurónovú sieť. Nastavili sme podmienku skoršieho zastavenia, ktoré meralo, či sa hodnoty straty nezačali zvyšovať, a zároveň, či sa vybrali najlepšie posledné váhy modelu.

Model sme trénovali vo viacerých krokoch a s rôznymi konfiguráciami. Pri výbere počtu neurónov sme použili konfigurácie 256, 512, 1024 neurónov, aby sme zachytili, či model potrebuje vyšší počet neurónov pre lepšie zachytenie zložitosti dát vo vzoroch alebo, či nižšie množstvo neurónov bude stačiť, aby nenastalo pretrénovanie modelu.

Počet epoch sme stanovili na číslach 50 a 100, aby sme zistili, či model potrebuje viac opakovaní trénovania alebo či nenastane pri väčšom množstve epoch pretrénovanie.

Veľkosť dávky sme určili v menších hodnotách 4 a 8, aby nenastalo pretrénovanie, ale taktiež, aby model dokázal vzory lepšie generalizovať.

Pre každú kombináciu parametrov sme vytvorili tri modely a tieto modely sme následne ukladali, aby sme mohli v testovacej a evalvačnej fáze porovnať ich výsledky.

Posúdenie modelu

Po testovaní rôznych modelov natrénovaných v prvej fáze sme objavili päť modelov, ktorých výsledky mali **hranicu** pod hodnotou 1. Tieto výsledky je možné vidieť v tabuľke 2.

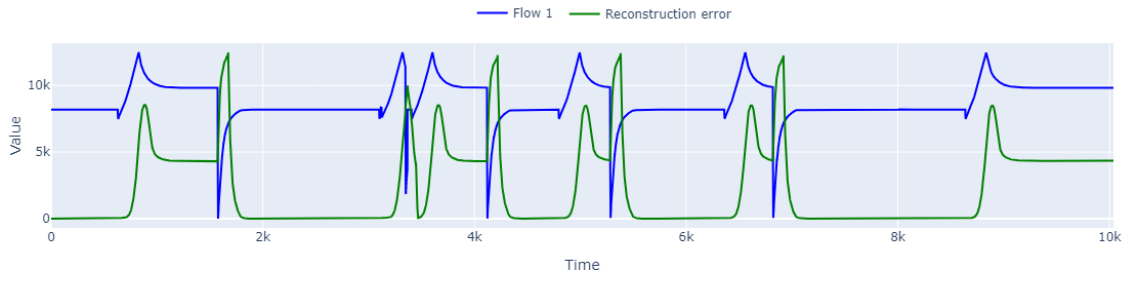
Tabuľka 2 Výsledky testovania 5 najlepších modelov

Neuróny	Dávka	Epochy	Hranica	Zaznamenanie úniku
1024	4	50	0.599463	2024-01-13 19:04:24
256	4	100	0.130326	2024-01-13 19:04:20
256	8	100	0.111130	2024-01-13 19:04:20
512	8	100	0.279510	2024-01-13 19:04:21
512	8	50	0.132239	2024-01-13 19:04:21

Vybrané modely sme následne evalvovali druhý krát na dátach, ktoré obsahovali únik. Podľa hranice sme určili časovú známku, v ktorej nastalo zaznamenanie úniku.

Z tabuľky 2 je vidieť, že 2 modely, ktoré v najrýchlejšom čase zaznamenali únik, boli modely, ktoré mali v strednej vrstve 256 neurónov.

Po vybraní najlepšej neurónovej siete sme tento model použili na rekonštrukciu grafu trénovacích dát, pri generovaní ktorých bolo používané čerpadlo. Ako je vidieť na obrázku 26, náš model nedokázal správne rekonštruovať dané úseky grafu a z tohto dôvodu bolo potrebné vrátiť sa späť k modelovaniu.



Obrázok 26 Rekonštrukcia trénovacích dát s čerpadlom

Vývoj modelu

Po natrénovaní predošlého modelu sme začali modelovať druhý model, ktorého trénovacie údaje boli generované s činnosťou čerpadla. Tieto dáta (Obrázok 23) obsahovali výrazné skoky, ktoré sa náš model potreboval naučiť.

Vytvorili sme nový model (Kód 6), ku ktorému sme pridali strednú vrstvu, a taktiež sme použili L1 regularizér. Tieto dve zmeny nám mali zabezpečiť schopnosť neurónovej siete naučiť sa menej všeobecné prípady.

```
model = Sequential([
    # Encoder
    Sequential([
        Dense(
            400, input_shape = (400,),
            activation = 'relu', kernel_regularizer = l1()
        ),
        Dense(neurons_2, activation = 'relu', kernel_regularizer = l1()),
        Dense(neurons, activation = 'relu', kernel_regularizer = l1()),
    ]),
    # Decoder
    Sequential([
        Dense(
            neurons, input_shape = (neurons,),
            activation='relu', kernel_regularizer = l1()
        ),
        Dense(neurons_2, activation = 'relu', kernel_regularizer = l1()),
        Dense(400, activation = 'linear', kernel_regularizer = l1()),
    ]),
])
```

Kód 6 – Nový model pre dáta s čerpadlom

Tento model sme skúsili natrénovať s rôznymi konfiguráciami počtu neurónov, epoch a veľkosťou dávky. Trénovacie dáta sme rozšírili o nové dáta, ktoré obsahovali údaje s prepínaním čerpadla.

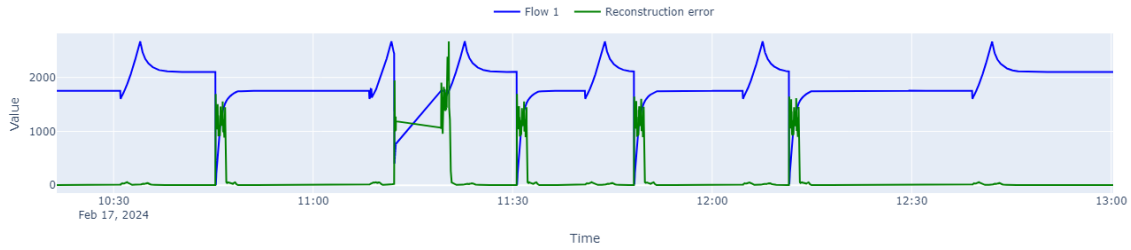
Posúdenie modelu

Natrénované modely sme testovali pomocou predošlej metódy a dostali sme výsledky v tabuľke 3.

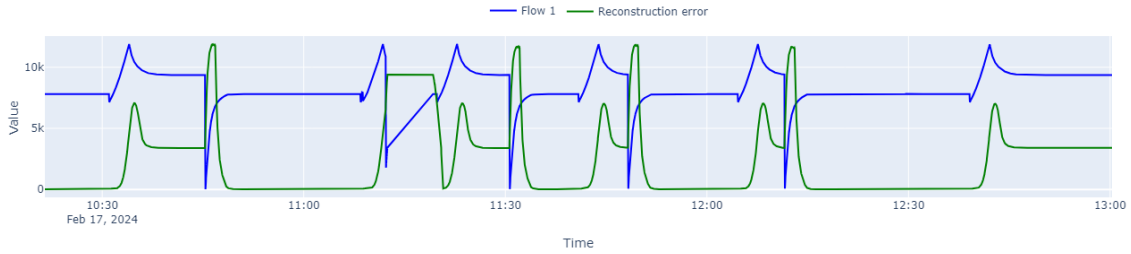
Tabuľka 3 Výsledky evalvácie natrénovaných modelov

Neuróny	Dávka	Epochy	Hranica	Zaznamenanie úniku
512, 512	16	30	5.388449	2024-01-13 19:04:23
256, 256	16	30	4.002168	2024-01-13 19:04:21

Ako je vidieť v tabuľke 3, naše najlepšie modely mali vyššiu hranicu ako modely predošlého testovania. Tieto modely sme následne porovnali na našich testovacích dátach, ktoré obsahovalo prepínanie čerpadla. Obrázky 27 a 28 nám ukazujú, že neurónová sieť, ktorá mala 2 vrstvy s 256 neurónmi, sa dokázala lepšie naučiť vzory z grafu, kde bolo čerpadlo zapnuté alebo vypnuté. Aj napriek zlepšeniu v rozoznávaní týchto vzorov, tento model nedokázal predikovať všetky prípady správne.



Obrázok 27 Rekonštrukčná chyba modelu s 256 neurónmi



Obrázok 28 Rekonštrukčná chyba modelu s 512 neurónmi

Vývoj modelu

Nakoľko náš predošlý model bol natrénovaný na údajoch, v ktorých bolo aj nebolo zaznamenané čerpadlo a nedokázal správne rozoznať únik v potrubí od zapnutia čerpadla, tak sme sa rozhodli vytvoriť nový model, ktorý by sme natrénovali iba na údajoch s čerpadlom.

Na trénovanie modelu sme si vygenerovali nový dátový súbor, ktorý obsahoval iba hodnoty premenných v potrubí počas opakovaného používania čerpadla. Tieto údaje mali rovnakú štruktúru ako doterajšie údaje, a preto sme mohli použiť náš predošlý skript na prípravu dát.

Tento model sme trénovali s rovnakou architektúrou ako predošlý model a použili sme rôzne konfigurácie, aby sme zistili najlepšiu z nich.

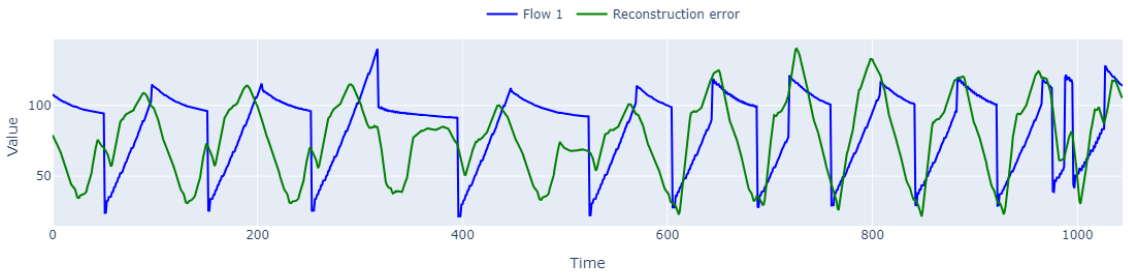
Posúdenie modelu

V tabuľke 4 môžeme vidieť modely, ktorých konfigurácie nám vyšli ako najlepšie spomedzi trénovaných modelov.

Tabuľka 4 Výsledky modelu trénovaného na údajoch s čerpadlom

Neuróny	Dávka	Epochy	Hranica	Zaznamenanie úniku	Rozpoznané čerpadlo
256, 256	16	50	97.808303	2024-01-13 19:04:49	Nie
256, 512	16	50	47.606444	2024-01-13 19:04:56	Nie
512, 512	16	30	85.444754	2024-01-13 19:04:29	Nie
512, 512	16	50	29.848485	2024-01-13 19:04:24	Nie
512, 1024	16	30	73.100396	2024-01-13 19:04:40	Nie

V údajoch z tabuľky sme pozorovali, že žiadny model nebol schopný rozoznať činnosť čerpadla v potrubí od úniku tekutiny v potrubí. Následne sme vytvorili čiarové grafy (Obrázok 29) znázorňujúce ako sa vyvíjala rekonštrukčná chyba modelu s najnižšou hranicou.



Obrázok 29 Rekonštrukčná chyba modelu trénovaného iba na údajoch s čerpadlom

Z údajov v tabuľke 4 a čiarového grafu môžeme usúdiť, že tento model je nefunkčný a nedokáže rozpoznať činnosť čerpadla od úniku v potrubí.

Vývoj modelu

Keďže predošlý model nedokázal správne rekonštruovať graf údajov v prípade, kedy bol použité čerpadlo v potrubí, tak sme sa rozhodli zmeniť architektúru nášho modelu. Prešli sme z architektúry doprednej neurónovej siete a nahradili sme ich vrstvami LSTM (Long-Short Term Memory). Nakoľko si tieto siete dokážu zapamätať údaje z predošlých vstupov, náš model si môže zapamätať údaje z premenných potrubia, ktoré získal v blízkej minulosti.

Ako je vidieť v Kóde 7, vo funkcii, ktorá vytvorí a natrénuje model, sme vykonali úpravy, kde sme nahradili vrstvy Dense vrstvami LSTM a na vstupe sme použili vrstvy Input a RepeatVector. Tieto vrstvy sme poprepájali za sebou a ku koncu sme z vstupnej vrstvy Encoderu a výstupnej vrstvy Decoderu vytvorili Autoencoder.

```
from tensorflow.keras.layers import Input, LSTM, RepeatVector

encoderInput = Input(shape = (400, 1))
encoderOutput = LSTM(512, return_sequences = False)(encoderInput)

decoderInput = RepeatVector(400)(encoderOutput)
decoderOutput = LSTM(400, return_sequences = True)(decoderInput)

model = Model(inputs = encoderInput, outputs = decoderOutput)
```

Kód 7 Kód architektúry LSTM Autoencoderu

Náš nový model sme bohužiaľ nedokázali natrénovať z dôvodu nedostatku technických zdrojov a náročnosti neurónovej siete. Hlavná príčina tohto problému je skutočnosť, že Autoencoder sa musí skladať z dvoch samostatných neurónových sietí, ktoré majú viac vrstiev. Z tohto dôvodu sme ukončili prácu s týmto modelom a nevykonávali sme na ňom posúdenie, evalváciu a ani testovanie.

3.5 EVALVÁCIA

Vo fáze evalvácie sme sa pozreli na výsledky modelov a posúdili sme, či tieto modely splnili požiadavky klienta. Následne sme vyhodnotili, či neboli prehliadnuté faktory, ktoré mohli ovplyvniť doterajšie výsledky a posúdili sme, ako bolo potrebné postupovať.

Evalvácia výsledkov

Na základe pozorovaní na obrázkoch 27, 28 a 29 môžeme usúdiť, že model nespĺňa všetky požiadavky dané klientom. Model bol typu neurónová sieť a bol schopný rozoznať, že v potrubí s tekutinou nastal únik na základe tlaku a prietoku v potrubí, ale napriek tomu nedokázal rozoznať použitie čerpadla od úniku.

Posúdenie procesu

V tejto úlohe sme sa spätne pozreli na doterajší proces a skontrolovali sme jednotlivé faktory, či negatívne neovplyvnili naše výsledky. Tieto faktory sme pozorovali medzi úlohami, v ktorých sme pripravovali údaje, trénovali a testovali náš model.

Vo fáze prípravy dát sme skontrolovali dáta a našli sme 2 faktory, ktoré mohli poškodiť model v trénovacej fáze. Tieto údaje sme odstránili. Na konci fázy prípravy

dát sme naše agregované dátové súbory prešli validačnou funkciou, aby sme sa uistili, že tieto súbory sú v poriadku.

Pri trénovaní modelu sme použili jednu a dve skryté vrstvy. Dôvodom pre naše konanie bolo, aby sme predišli pretrénovaniu nášho modelu.

Pri výbere počtu epoch, neurónov a veľkosti dávky sme sa snažili predísť prípadu, v ktorom by náš model bol taktiež pretrénovaný a nedokázal by správne rekonštruovať vstup, ktorý nepozná.

V úlohe testovania sme používali údaje, ktoré model nepoznal, aby sme zistili, či nenastalo pretrénovanie pri modelovaní. Taktiež sme použili 2 varianty testovacích dát, aby sme určili výsledky modelu v oboch prípadoch.

Určenie ďalších krokov

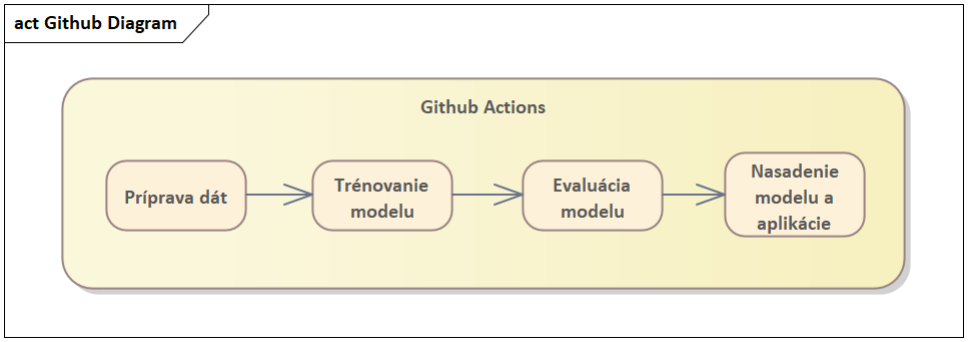
Keďže natrénovaný model nebol schopný rozoznať únik v potrubí v období, kedy bolo čerpadlo prepínané, tak tento model nesplňuje požiadavky, ktoré sme dostali od klienta. Z tohoto dôvodu model nie je vhodný na použitie v produkčnom prostredí. Napriek nevhodnosti tohto modelu v prostredí s čerpadlom, model dokáže správne a rýchlo predikovať únik v potrubí, ktoré nie je ovplyvnené čerpadlom. Tento model teda môžeme nasadiť do produkčného prostredia, v ktorom nie je použité čerpadlo.

3.5 NASADENIE

Vo fáze nasadenia sme vytvorili plán priebehu, podľa ktorého sme následne vytvorili potrebný softvér, ktorý automatizoval naše úlohy z predošlých fáz. K natrénovanému modelu sme s cieľom ukončiť celý životný cyklus objavovania znalostí vytvorili serverovú aplikáciu, ktorá dokáže používať tento model. V závere nasadenia sme zhodnotili náš projekt a jeho výstup.

Plánovanie nasadenia

Plán nasadenia (Obrázok 30) spočíval v štyroch fázach, ktoré bolo potrebné automatizovať pre modelovanie v budúcnosti. Na automatizáciu sme použili Github Actions, ktoré nám dovolilo rozdeliť jednotlivé úlohy (Job). Prvé tri Joby korešpondovali s úlohami, ktoré sme riešili v predošlých fázach. Posledný Job mal za úlohu natrénovaný model spolu s aplikáciou nahrať na produkčný server.



Obrázok 30 Diagram plánu nasadenia

Automatizácia úloh

Prvým krokom automatizácie úloh bolo vytvorenie pracovného toku (Workflow). Táto akcia je uložená v ceste .github/workflows/main.yml.

Ako je vidieť v kóde 8, náš Workflow sme obmedzili, aby sa spúšťal iba počas udalosti (Event), ktorá spočívala z pull requestu do branchu master (nášho hlavného branchu). Následne sme tento Workflow rozdelili na 4 Joby:

- data_preparation – načíta neupravené dátové súbory, vykoná prípravu dát z týchto súborov a v poslednom kroku tieto údaje spojí do jednotlivých súborov pre trénovanie a testovanie modelu,
- training – vytvorí model neurónovej siete a potom použije súbory pripravené v predošlom Jobe na natrénovanie viacerých verzií neurónovej siete, ktoré sú uložené ako artefakty,
- evaluation – vykoná evalváciu modelov s vyhodnotením vhodnej hodnoty pre hranicu každého modelu, vykoná testovanie natrénovaných modelov pre určenie funkčnosti a dodržania požiadaviek klienta,
- deployment – použije SSH prístup na pripojenie k serveru, z ktorého následne vykoná nasadenie natrénovaného modelu a aplikácie na náš tento server.

```yaml
name: CI

on:
  pull_request:
    types: [closed]
    branches:
      - master

jobs:
  data_preparation:
    ...

  training:
    ...

  evaluation:
    ...

  deployment:
    ...
```

Kód 8 - Github akcia main.yml

Keďže súbory vygenerované v Joboch sa neprenášajú medzi sebou, tak sme použili kód 9, v ktorom sme vygenerovali artefakty (Artifact) a následne sme ich použili v nasledujúcich Joboch.

```yaml
- uses: actions/download-artifact@v2
  with:
    name: data
    path: python/csv/output

...

- uses: actions/upload-artifact@v2
  with:
    name: model
    path: python/models
```

Kód 9 Sťahovanie a nahranie artefaktov v Jobe

Úlohou prvých 3 Jobov bolo nainštalovať Python s potrebnými knižnicami a spustiť skripty, ktoré vykonali naše doterajšie akcie z predošlých fáz. Na to sme použili jednoduchý príkazy *run* v našich Joboch, ako je vidieť v kóde 10.

```
- run: |
    python -m pip install --upgrade pip
    pip install pandas tensorflow==2.14.0

- run: python3 python/scripts/build_model.py
```

Kód 10 Inštalácia nástrojov a spustenie skriptu

Job deployment (Kód 11) mal za úlohu nasadiť nami vygenerovaný model s aplikáciou na produkčný server. Na nahranie súborov sme použili SSH prístup, ktorý nám umožnil vzdialene sa pripojiť na server cez bežca (Runner) a naklonovať repozitár na server.

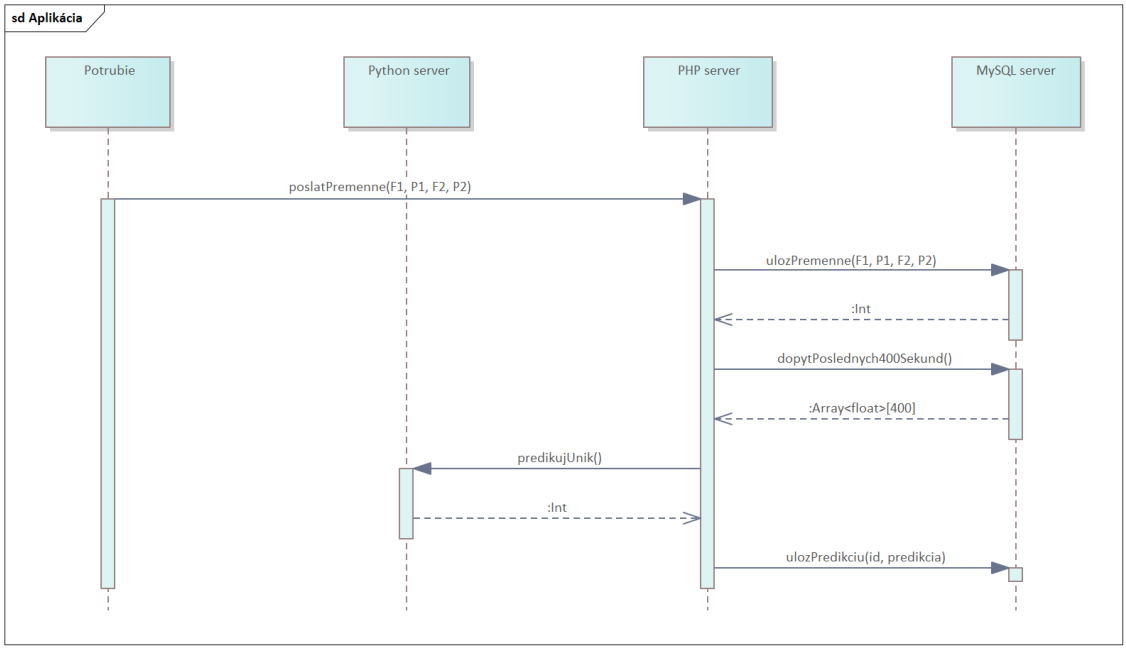
```
deployment:
  runs-on: ubuntu-latest
  name: Deployment
  needs: evaluation
  steps:
    - uses: actions/checkout@v2

    - env:
        PRIVATE_KEY: ${{ config.SSH_PRIVATE_KEY }}
        SERVER_IP: ${{ config.SERVER_IP }}
        GITHUB_REPO: ${{ config.GITHUB_REPO }}
      run: |
        mkdir -p ~/.ssh
        echo "$PRIVATE_KEY" > ~/.ssh/id_rsa
        chmod 600 ~/.ssh/id_rsa
        ssh-keyscan -H $SERVER_IP >> ~/.ssh/known_hosts
        git clone $GITHUB_REPO
```

Kód 11 Deployment Job

Aplikácia

Aplikácia, ktorá bude používať model, ktorý sa skladá z troch samostatných zložiek. Na sekvenčnom diagrame (Obrázok 31) môžeme vidieť, ako prebieha komunikácia jednotlivých častí aplikácie. Aplikáciu vyvíjame v prostredí Docker, aby sme zabezpečili konzistentné nasadenie na rôznych platformách a vyhli sa potrebnému zapuzdreniu konfigurácií s projektom.

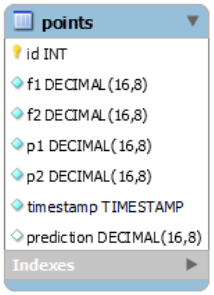


Obrázok 31 Sekvenčný diagram komunikácie aplikácie

Hlavná časť aplikácie bude server používajúci PHP. Jeho hlavnou úlohou je prijímať dáta z potrubia, komunikovať so serverom, ktorý bude hostiť natrénovaný model a všetky dáta, ktoré príjme, odosielať na server s MySQL databázou.

Model bude umiestnený na serveri, ktorý používa Python. Jedinú úlohu, ktorú tento server má, je vykonať predikciu na dátach, ktoré príjme a vrátiť rekonštrukčnú chybu danej predikcie.

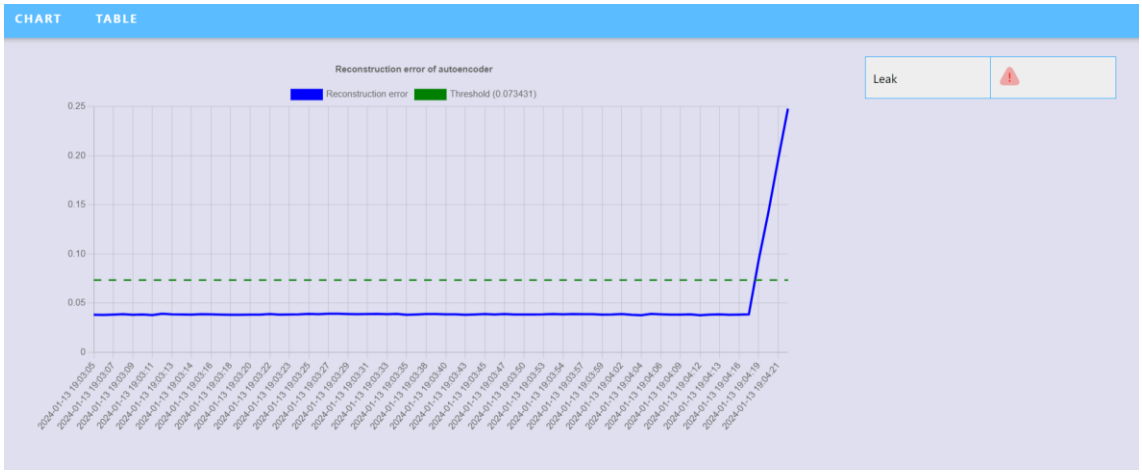
MySQL databáza bude obsahovať iba 1 tabuľku s názvom points (body), ktorej štruktúru môžeme vidieť v databázovej schéme (Obrázok 32)



Obrázok 32 Schéma MySQL databázy

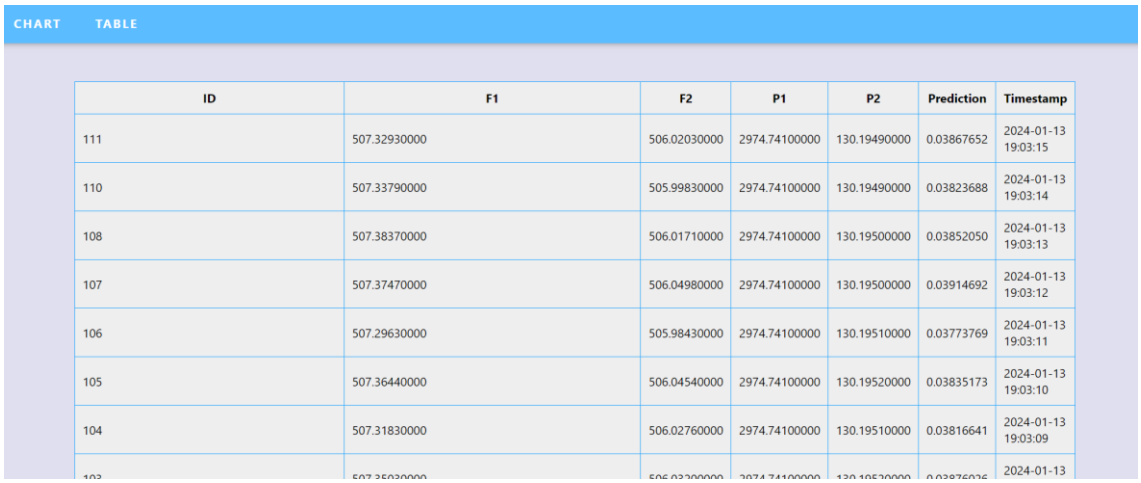
Keďže aplikácia potrebuje na správne fungovanie prijímať údaje v definovanom intervale, tak sme vytvorili skript, ktorý nám tieto testovacie dáta v našej aplikácii dokáže falšovať. Skript pošle údaje, ktoré sú prijaté serverom každú sekundu, následne je na nich vytvorená predikcia úniku, a potom sú uložené do databázy.

Po otvorení aplikácie máme k dispozícii dve zobrazenia. Prvé zobrazenie zobrazuje graf (Obrázok 33), ktorý sa aktualizuje v skoro reálnom čase na základe údajov z databázy. Pri načítaní tohto grafu sa spustí funkcia, ktorá každých 5 sekúnd vytvorí žiadosť na server a v odpovedi dostane príslušné údaje.



Obrázok 33 Aplikácia – Graf údajov v reálnom čase

Druhé zobrazenie aplikácie, ktoré môžeme vidieť na obrázku 34, nám zobrazí tabuľku údajov. Táto tabuľka je taktiež aktualizovaná v skoro reálnom čase každých 5 sekúnd. Na rozdiel od grafu, v tomto zobrazení máme väčšiu úroveň detailu samotných údajov a vidíme teda celú tabuľku z databázy so všetkými stĺpcami.



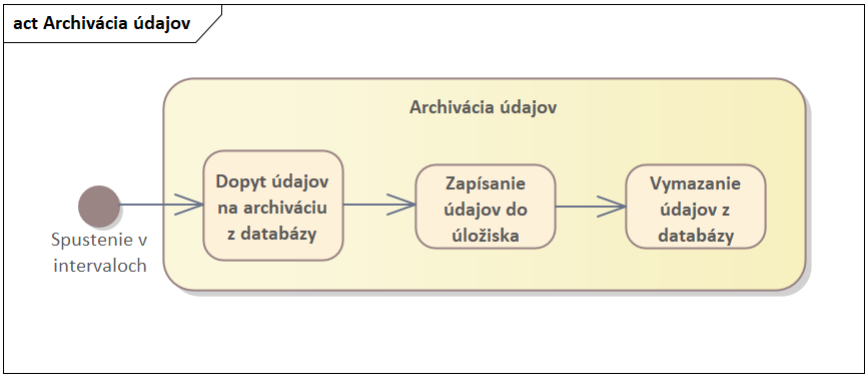
ID	F1	F2	P1	P2	Prediction	Timestamp
111	507.32930000	506.02030000	2974.74100000	130.19490000	0.03867652	2024-01-13 19:03:15
110	507.33790000	505.99830000	2974.74100000	130.19490000	0.03823688	2024-01-13 19:03:14
108	507.38370000	506.01710000	2974.74100000	130.19500000	0.03852050	2024-01-13 19:03:13
107	507.37470000	506.04980000	2974.74100000	130.19500000	0.03914692	2024-01-13 19:03:12
106	507.29630000	505.98430000	2974.74100000	130.19510000	0.03773769	2024-01-13 19:03:11
105	507.36440000	506.04540000	2974.74100000	130.19520000	0.03835173	2024-01-13 19:03:10
104	507.31830000	506.02760000	2974.74100000	130.19510000	0.03816641	2024-01-13 19:03:09

Obrázok 34 Aplikácia – Tabuľka údajov v reálnom čase

Problémy pri nasadení

Reálne produkčné prostredie môže obsahovať faktory, ktoré by mohli obmedziť chod aplikácie. Jedným z týchto faktorov je objem dát. Keďže senzorový systém potrubia je naprogramovaný, aby odosielal hodnoty premenných zo senzorov na server každú sekundu, mohol by nastať problém s ukladaním týchto hodnôt do databázy.

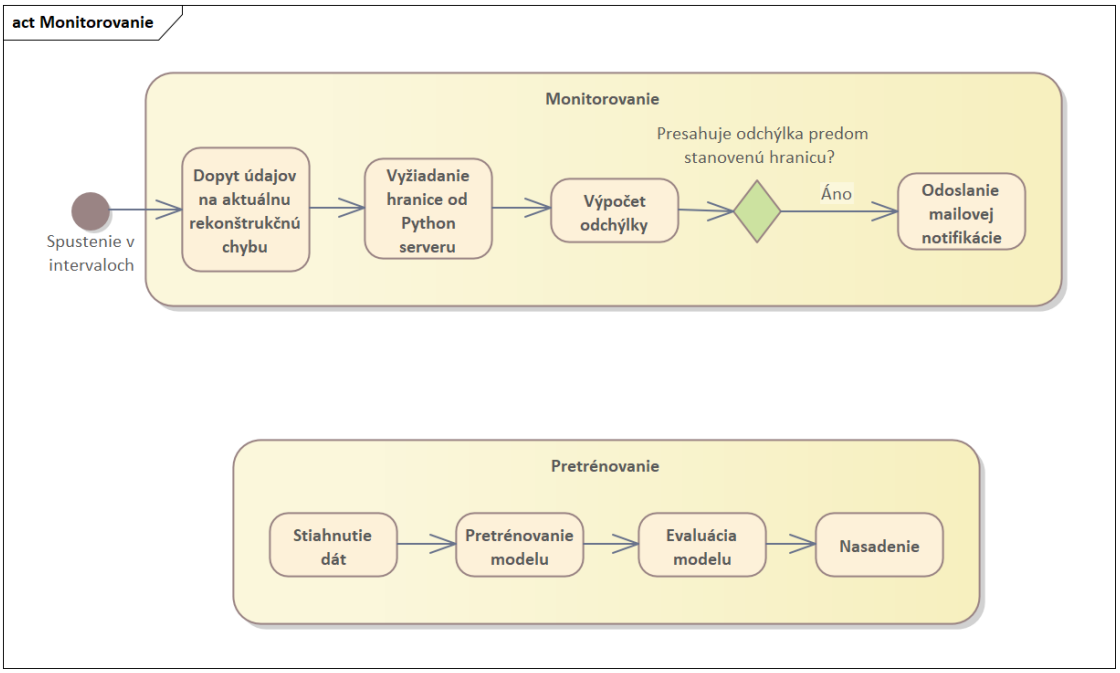
Navrhované riešenie (Obrázok 35) pre takýto stav môže byť použitie systému archivácie údajov, v ktorom by sa údaje z databázy exportovali, uložili a následne vymazali z databázy vo vopred určenom intervale. Aplikácia by následne bola obmedzená z hľadiska množstva dát, ktoré si používateľ dokáže zobraziť, ale zamedzilo by sa spomaleniu alebo potenciálnemu spadnutiu aplikácie.



Obrázok 35 Diagram procesu archivácie údajov

Monitorovanie a pretrénovanie

Jeden z hlavných problémov nasadenia modelu na server bude odklon predikcií modelu. Keďže model bude prijímať údaje po dlhú dobu, je potrebné monitorovať odchýlku modelu a na základe tejto odchýlky pretrénovať neurónovú sieť na nových hodnotách premenných. Diagram týchto dvoch procesov je možné vidieť na diagrame aktivít (Obrázok 36).



Obrázok 36 Diagram procesov Monitorovanie a Pretrénovanie

Prvým krokom tejto úlohy je monitorovanie modelu a jeho odchýlky. V kóde 12 môžeme pozorovať skript, ktorý toto monitorovanie vykonáva. Prvou časťou úlohy je zistenie priemernej rekonštrukčnej chyby modelu za predošlý deň. Keďže tieto údaje sú držané v databáze, tak na výpočet tejto hodnoty je použitý SQL príkaz. Druhý krok spočíva v zavolaní žiadosti na Python server, ktorý nám vráti hranicu používaného modelu. V poslednom kroku sa porovná vypočítaná odchýlka z týchto dvoch hodnôt a v prípade, že daná odchýlka presahuje predom definovanú hranicu, tak sa odošle notifikačný mail o tomto probléme.

```
$today = date('Y-m-d H:i:s', strtotime('today'));
$yesterday = date('Y-m-d H:i:s', strtotime('yesterday'));

$sql = "SELECT AVG(points.prediction) AS 'average_reconstruction_error'
        FROM points WHERE points.timestamp BETWEEN '$yesterday' AND
        '$today' AND points.prediction IS NOT null";

$avgRE = selectSQL($sql)[0]['average_reconstruction_error'];

$expRE = curlGet("http://host.docker.internal:5000/threshold")['threshold'];
$expRE = $expRE / 2;

$deviation = ($avgRE - $expectedRE) / $expectedRE;

if ($deviation < MONITOR_THRESHOLD) return;

sendMail(MONITOR_THRESHOLD, $expectedRE, $avgRE);
```

Kód 12 Skript na monitorovanie odchýlky rekonštrukčnej chyby

Prípravu údajov v úlohe pretrénovania nie je možné automatizovať, nakoľko táto príprava si vyžaduje výber údajov z potrubia, ktoré neobsahujú zaznamenaný únik a vyváženosť hodnôt pre používanie čerpadla a nepoužívanie čerpadla. Z tohto dôvodu je v predošlom kroku odoslaný notifikačný mail, ktorý vývojára upozorní na potrebu pretrénovania modelu. Vývojár po vygenerovaní nových dátových súborov tieto súbory nahrá na server, ktorý následne spustí automatizované pretrénovanie modelu.

Pretrénovanie modelu je definované v novom Workflowe s názvom drift.yml, ktorého Joby sú vypísané v kóde 13.

```
jobs:
  drift-download-data:
    ...

  drift-retraining:
    ...

  drift-evaluation:
    ...

  deployment:
    ...
```

Kód 13 Workflow pretrénovania modelu

Job s názvom drift-download-data má za úlohu stiahnuť súbory potrebné na pretrénovanie modelu. Na stiahnutie sme použili command download-artifact (Kód 14). Pomocou neho sa stiahne jeden súbor trénovacích dát, dva súbory testovacích dát a v poslednom kroku sa stiahne samotný model. Následne sa tieto stiahnuté súbory nahrajú

ako artefakty, aby sme ich mohli v ďalších Joboch použiť na pretrénovanie a evalváciu nového modelu.

```
- uses: actions/download-artifact@v2
  with:
    name: train-file
    url: ${{ config.FILE_DOWNLOAD_URL }}/drift/data_train.csv
    path: python/csv/drift
```

Kód 14 Stiahnutie súborov v Jobe deployment-download-data

Joby drift-retraining a drift-evaluation spustia Python skript, ktorý vykoná pretrénovanie a evalváciu modelu. Tieto Python skripty sú podobné skriptom z hlavného Workflowu. Pretrénovací skript nevytvorí model, ale načíta stiahnutý model z prvého kroku. Job deployment je totožný s Jobom z hlavného Workflowu.

4 DISKUSIA

Počas prípravy dát sme vykonávali úlohu integrácie dát, v ktorej sme použili očistené údaje na vygenerovanie časových rádov, ktoré sme používali pri trénovaní modelu neurónovej siete. Tieto časové rády obsahovali 100 sekúnd spojitých údajov, ktoré predstavovali vývoj hodnôt premenných tekutiny v potrubí. V budúcnosti by sa mohla osvedčiť zmena v počte údajov, ktoré tieto časové rády obsahujú. Pri zvýšení tohto počtu by neurónová sieť mala možnosť získať viac informácií na výstupe a potenciálne by dokázala rozpoznať rozdiel medzi únikom tekutiny a činnosťou čerpadla. V takomto prípade by bolo potrebné zvýšiť počet vrstiev a neurónov neurónovej siete.

Jeden z faktorov, ktoré mohli negatívne ovplyvniť trénovanie neurónovej siete, bol objem dát použitých vo fáze modelovania. V nasledujúcich výskumoch by sme odporúčali vygenerovať väčšie množstvo údajov, aby bol model schopný lepšie pochopiť štruktúru údajov a potenciálne vytvoriť lepšie výsledky pre problematiku detekcie únikov v potrubí.

V našej práci sme použili typ neurónovej siete Autoencoder s doprednou väzbou. Vytvorili sme aj rekurentný model Autoencoder, ktorý používal vrstvy LSTM, no nakoľko nám technické faktory nášho prostredia neumožnili natrénovať model, tak sme nedosiahli merateľný výsledok. Tento model mohol byť užitočný pre zistenie nových informácií alebo faktorov, ktoré určujú, či model dokáže byť natrénovaný na údaje z potrubia počas činnosti čerpadla.

Užitočné informácie by sme získali aj v prípade použitia metód, ktoré nepoužívajú modely neurónových sietí, ale iné modely strojového učenia. Tieto modely by mohli užitočne prispieť k výskumu a mohli by sa osvedčiť lepšími výsledkami.

Nakoľko údaje generované simulátorom obsahovali iba údaje o premenných tekutiny v potrubí a neexistoval údaj, ktorý by určil, či v danom momente prebiehal únik tekutiny, tak sme použili metódy učenia bez učiteľa.

Počas opätovného trénovania neurónovej siete bolo potrebné pripraviť aktuálne údaje o tekutine v potrubí, ktoré sme použili na trénovanie modelu. Keďže tieto údaje musia byť vyvážené z hľadiska použitia, či nepoužitia čerpadla a je nutná neprítomnosť únikov v pripravených dátach, nedokázali sme automatizovať túto časť procesu, a preto sme sa rozhodli, že v tejto fáze je potrebné použitie ľudského faktoru na pripravenie súborov, ktoré sa použijú na trénovanie neurónovej siete.

Použili sme CRISP-DM ako metodiku pre vývoj modelu neurónovej siete. Vývojový model ASUM-DM obsahuje dodatočné kroky, procesy a šablóny pre vývoj modelu, ale nie je veľa dostupných informácií ohľadom fungovania a práce s týmto vývojovým modelom. Z tohto dôvodu sme sa rozhodli použiť CRISP-DM, ktorý sme prispôsobili potrebám nášho projektu a v prípade nedostatkov informácií sme boli schopní dodatočne dohľadať potrebné manuály.

V našej práci sme použili nástroje GitHub a GitHub Actions. Tieto nástroje nám slúžili na verziovanie aplikácie v Git repozitári a automatizáciu úloh. Keďže tieto služby sú open-source a majú prehľadnú dokumentáciu ako aj veľkú komunitnú podporu (community support), tak boli správnou voľbou pre náš projekt. Pre celý proces by bolo vhodné využiť cloudové riešenie ako napríklad platformu AWS, ktorá ponúka široký sortiment služieb vhodných pre naše použitie. Namiesto spomenutých nástrojov by sme využili AWS služby CodeCommit a CodeBuild.

ZÁVER

V práci sme analyzovali technológie a metódy používané pri vývoji modelov strojového učenia na detekciu anomálií. Bližšie sme popísali technológie neurónových sietí a jednotlivé druhy týchto sietí. Vysvetlili sme ako funguje detekcia anomálií a aké metódy sa v dnešnej dobe používajú na rozpoznanie týchto anomálií. Poukázali sme na rôzne modely vývoja, ktoré sa používajú pri dátovej vede na trénovanie prediktívnych alebo analytickým modelov. Vysvetlili sme pojmy MLOps a Github Actions a ich využitie v modeloch vývoja.

V praktickej časti sme použili model vývoja CRISP-DM, aby sme vytvorili a natrénovali funkčný model neurónovej siete, ktorá dokáže rozoznať anomálie v premenných potrubia, ktoré sú spôsobené únikmi tekutiny. Postupovali sme podľa jednotlivých fáz modelu vývoja a v každej fáze sme vykonali jednotlivé úlohy, ktoré do danej fázy patria.

Prvá fáza bola obchodné pochopenie, v ktorej sme sa zamerali na porozumenie problematike klienta a aké výstupy projektu od nás klient vyžaduje. Po zhodnotení obchodných cieľov a požiadaviek od klienta sme vytvorili vhodný a uskutočniteľný návrh plánu projektu

Fáza porozumenia dát spočívala vo vygenerovaní menšej vzorky údajov použitím simulátora. Vzorky sme následne preskúmali, aby sme pochopili ich štruktúru a určili problémy, ktoré sa v dátach nachádzajú a korektne sme skontrolovali validitu a kvalitu vygenerovaných údajov.

Príprava dát bola fáza, v ktorej sme využili znalosti o dátach na vykonanie potrebného čistenia dát. Následne sme nad týmito údajmi vykonali agregačné operácie, aby sme ich transformovali do formy, ktorú sme využili na trénovanie, testovanie a evalváciu neurónovej siete.

Vo fáze modelovania sme sa pokúsili vytvoriť tri modely doprednej neurónovej siete Autoencoder a jeden model LSTM neurónovej siete Autoencoder. Tieto modely sme následne posúdili a určili, či tieto modely dokážu vykonávať funkciu, na ktorú sme ich natrénovali. Zvolili sme vhodnú architektúru neurónovej siete, Autoencoder, nakoľko je tento druh siete vytvorený tak, aby sa naučil rozpoznávať vzory v údajoch.

Vo fáze evalvácia sme preskúmali modely, ktoré sme natrénovali a určili sme, či tieto modely spĺňajú požiadavky od klienta. Po posúdení týchto modelov sme určili ďalšie kroky, ktoré mali nasledovať.

Vo fáze nasadenia sme tento model nahrali na produkčný server, kde spolu s vytvorenou aplikáciou tento model dokáže vykonávať svoj účel. Pri nasadení modelu sme automatizovali jednotlivé úlohy, ktoré sme vykonávali a vytvorili sme proces, ktorý by model dokázal pretrénovať v prípade, kedy bude odhalená prítomnosť odchýlky rekonštrukčnej chyby modelu.

V tomto projekte sa nám podarilo čiastočne dosiahnuť náš cieľ natrénovania neurónovej siete, ktorá dokáže rozoznať únik tekutiny v potrubí. Aj keď model nebol schopný rozoznať únik tekutiny od zapnutia čerpadla v potrubí, bol schopný jasne rozoznať, či v potrubí nastane únik pokiaľ čerpadlo nie je používané. Model by sa tak dal použiť v praxi za pomoci ľudského faktoru, ktorý by dokázal rozoznať, či predikcia modelu, že v potrubí sa nachádza únik, nebola spôsobená iba zapnutím čerpadla.

Zdrojový kód aplikácie, využitých Python skriptov a dátových súborov je dostupný na verejnom GitHub repozitári, ktorý sa nachádza v odkaze https://github.com/ukf-lukasgrofcik/pipeline-leak-prediction-system.

ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV

AGGARWAL, Charu, 2023. An Introduction to Neural Networks. Dostupné na doi: 10.1007/978-3-031-29642-0_1

ALLA, Sridhar, 2021. Beginning MLOps with MLFlow. Dostupné na doi: 10.1007/978-1-4842-6549-9

ARAFA, Ahmed, Nawal El-Fishawy, Mohammed Badawy, Marwa Radad 2023. RN-Autoencoder: Reduced Noise Autoencoder for classifying imbalanced cancer genomic data. Dostupné na: doi: 10.1186/s13036-022-00319-3

CHANDOLA, Varun, Arindam BANERJEE a Vipin KUMAR, 2017. Encyclopedia of Machine Learning and Data Mining, Second Edition. ISBN 978-1-4899-7685-7

CHAPMAN, Pete, Julian CLINTON, Randy KERBER, Thomas KHABAZA, Thomas REINARTZ, Colin SHEARER, Rüdiger WIRTH, 2000. CRISP-DM 1.0. Step-by-step data mining guide

FENG, Yong, Jiang ZHONG, Zhong-yang XIONG, Chun-xiao YE, Kai-gui WU, 2007. Network Anomaly Detection Based on DSOM and ACO Clustering. Dostupné na doi: 10.1007/978-3-540-72393-6_113

FOIS, Guiliana, Agüero CROVELLA, Gustavo Andrés BRITOS, Paola VERÓNICA, 2020. Investigación Formativa en Ingeniería. Dostupné na: doi: 10.5281/zenodo.4031253

GAUTAM, Naveenta, Amol CHOUDHARY, Brejesh LALL, 2021. Comparative study of neural network architectures for modelling nonlinear optical pulse propagation. Dostupné na doi: 10.1016/j.yofte.2021.102540

GITHUB, 2024. GitHub Docs. Dostupné na internete: < https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions>

IBM, 2015. Delivery Process: ASUM-DM. Dostupné na internete: <http://gforge.icesi.edu.co/ASUM-DM_External/index.htm#cognos.external.asum-DM_Teaser/deliveryprocesses/ASUM-DM_8A5C87D5.html_desc.html?proc=_0eKIHlt6EeW_y7k3h2HTng&path=_0eKIHlt6EeW_y7k3h2HTng>

KREUZBERGER, Dominik, Niklas KÜHL, 2023. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. Dostupné na doi: 10.1109/ACCESS.2023.3262138

KUMAR, Rajnish, 2022. The Principles of Clean Code: DRY, KISS, and YAGNI. Dostupné na internete: <https://medium.com/@curiousraj/the-principles-of-clean-code-dry-kiss-and-yagni-f973aa95fc4d>

MEHROTRA, Kishan, Chilukuri K. MOHAN, HuaMing HUANG, 2017. Anomaly Detection Principles and Algorithms. Dostupné na doi: 10.1007/978-3-319-67526-8

MUKHERJEE, Sudipta, 2020. ML.NET Revealed. Dostupné na doi: 10.1007/978-1-4842-6543-7_9

NABATI, Elaheh GHOLAMZADEH, Klaus-Dieter THOBEN, 2017. On Applicability of Big Data Analytics in the Closed-Loop Product Lifecycle: Integration of CRISP-DM Standard. Dostupné na: doi: 10.1007/978-3-319-54660-5_41

NASIMUL, Noman, 2020. A Shallow Introduction to Deep Neural Networks. Dostupné na: doi: 10.1007/978-981-15-3685-4

RAO, N. Thirupathi, Debnath BHATTACHARYYA, Hye-jin KIM, 2022. Anomaly Detection in Solar Radiation Forecasting Using LSTM Autoencoder Architecture. Dostupné na doi: 10.1007/978-981-16-8364-0_14

REBALA, Gopinath, Ajay RAVI, Sanjay CHURIWALA, 2019. An Introduction to Machine Learning. Dostupné na doi: 10.1007/978-3-030-15729-6_13

SALVARIS, Mathew, Danielle Dean, Wee Hyong, 2018. Recurrent Neural Networks. Dostupné na doi: 10.1007/978-1-4842-3679-6_7

SCHNEIDER, Patrick, Fatos XHAFA, 2022. Anomaly detection: Concepts and methods. Dostupné na doi: 10.1016/B978-0-12-823818-9.00013-4

SCHRÖER, Christoph, Felix KRUSE, Jorge MARX GOMÉZ, 2021. A Systematic Literature Review on Applying CRISP-DM Process Model. Dostupné na: doi: 10.1016/j.procs.2021.01.199

SHAFIQUE, Umair, Haseeb QAISER, 2014. A Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA). ISSN 2351-8014 Vol. 12 No. 1 Nov. 2014, pp. 217-222

SIEGEL, Barry, 2020. Industrial Anomaly Detection: A Comparison of Unsupervised Neural Network Architectures. Dostupné na: doi: 10.1109/LSENS.2020.3007880

SORVISTO, Dayne, 2023. MLOps Lifecycle Toolkit. Dostupné na doi: 10.1007/978-1-4842-9642-4_4

SUNITHA, Basodi, Chunyan JI, Haiping ZHANG, Yi PAN, 2020. Gradient amplification: An efficient way to train deep neural networks. Dostupné na: doi: 10.26599/BDMA.2020.9020004

VISHWAKARMA, Gajendra, Chinmoy PAUL, A.M. ELSAWAH, 2020. An algorithm for outlier detection in a time series model using backpropagation neural network. Dostupné na: doi: 10.1016/j.jksus.2020.09.018

USHIKU, Yoshitaka, 2021. Long Short-Term Memory. Dostupné na doi: 10.1007/978-3-030-63416-2_300359

WENG, Yu, Lei LIU, 2018. A Sequence Anomaly Detection Approach Based on Isolation Forest Algorithm for Time-Series. Dostupné na doi: 10.1007/978-3-540-72393-6_113

WESLEY, Addison, 1994. Simulation Neuronaler Netze. ISBN 3-89319-554-8

ZOZNAM PRÍLOH

Príloha A – Enterprise Architect projekt vložený cez AIS

Príloha B – Zdrojový kód dostupný na GitHub repozitári https://github.com/ukf-lukasgrofcik/pipeline-leak-prediction-system