Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Analýza aktivit skladu pomocí Process Miningu a Machine Learning technologií

Diplomová práce

Autor: Bc. Vladislav Poverin

Studijní obor: Informační management im2-p

Vedoucí práce: Ing. Barbora Tesařová, Ph.D.

Odborný konzultant: Ing. Jakub Vancl MBA

ŠKODA AUTO a. s.

Hradec Králové Srpen 2022

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 20. 04. 2022 Jméno a příjmení studenta

Poděkování

Děkuji vedoucí diplomové práce Ing. Barboře Tesařové, Ph.D. a odbornému konzultantovi Ing. Jakubovi Vanclovi MBA a Ing. Janu Suchému ze společnosti ŠKODA AUTO, a. s. za metodické vedení práce, cenné rady a trpělivost.

Anotace

POVERIN, V. Analýza aktivit skladu pomocí Process Miningu a Machine Learning technologií. Hradec Králové, 2021/22. Diplomová práce na fakultě Informatiky a managementu Univerzity Hradec Králové. Vedoucí diplomové práce Ing. Barbora Tesařová, Ph.D.

Diplomová práce „Analýza aktivit skladu pomocí Process Miningu a Machine Learning technologií“ se zabývá zkoumáním, analýzou a extrakcí dat prostřednictvím software Splunk, konkrétně logových zpráv ve společnosti ŠKODA AUTO, a. s. Teoretická část práce vysvětluje aktuální principy a koncepty, kterých se využívá při práci s velkými objemy dat. Praktická část je věnována zpracování nestrukturovaných dat získaných z centrálního skladu společnosti, jejich analýze a následnému použití statistických modelů pro predikci. Cílem diplomové práce je poukázat jak na důležitost a výhody, které analýza nestrukturovaných dat přináší, tak na konkrétní východiska.

Klíčová slova: Splunk, log, Process mining, Data mining, Power BI, Python, Big Data, databáze, indexace, vizualizace, machine learning, statistické modely

Annotation

POVERIN, V. Analysis of warehouse activity using Process Mining and Machine Learning technologies. Hradec Králové, 2021/22. Master's thesis at the Faculty of Information Technologies and Management University Hradec Králové. Supervisor Ing. Barbora Tesařová, Ph.D.

This Diploma Thesis examines the different types of analysis, extraction and transformation techniques which are currently used when working with big data. The theoretical part of this thesis introduces and explains the methodology of knowledge discovery of databases and importance of data mining process of discovering rules and patterns in unstructured data. There are presented tools that are widely used for data visualization, process mining and machine learning. Practical part of the thesis is focused on implementation of the methods and techniques into the analysis of activities in the warehouse of ŠKODA AUTO, a. s. company. The goal of this work is to refer to importance and advantages of data analysis of unstructured data as demonstrate the mentioned practices of working with data.

Key words: Splunk, log, syslog, logging, Process mining, Data mining, Python, , Database, visualization, machine learning

Obsah

Úvod ... 1

Cíl práce ... 2

1 Dobývání znalostí z databází ... 3

1.1 Data Mining ... 5

1.1.1 Metodologie data miningu ... 5

1.2 Techniky a algoritmy data miningu ... 10

1.2.1 Klasifikace a diskriminace ... 10

1.2.2 Shlukování ... 11

1.2.3 Predikce (prediktivní analýza, regresní analýza) ... 11

1.2.4 Asociační pravidla / Asociační analýza ... 12

1.2.5 Neuronové sítě ... 12

1.3 Analýza časových řad ... 13

1.3.1 Dekompozice časových řad ... 13

2 Nástroje pro analýzu a vizualizaci dat ... 15

2.1 Získání dat ze zdroje ... 15

2.2 Syntaktická analýza dat ... 15

2.3 Indexace a vizualizace dat ... 16

2.3.1 Relační databáze ... 16

2.3.2 NoSQL databáze ... 17

2.4 Splunk ... 18

2.4.1 Architektura Splunk ... 19

2.5 Microsoft Power BI ... 24

2.5.1 Power BI architektura ... 24

2.5.2 Hlavní funkce Power BI ... 25

2.5.3 Digital Analysis Expressions (DAX) ... 26

3 Process Mining ... 27

3.1 Typy Process Miningu ... 28
3.2 Procesní log ... 29
4 Machine Learning ... 31
4.1 Typy Machine Learning systémů ... 32
4.1.1 Učení s učitelem ... 33
4.1.2 Učení bez učitele ... 34
4.1.3 Zpětnovazební učení ... 35
4.1.4 Online vs Offline učení ... 36
4.1.5 Instance vs Model-Based Learning ... 36
4.2 Nástroje pro programování ML ... 37
4.2.1 Python ... 37
4.3 Statistické modely používané v úlohách Machine Learning ... 38
4.3.1 Lineární a logistická regrese ... 38
4.3.2 Rozhodovací stromy ... 40
4.3.3 ARIMA ... 41
4.3.4 LSTM ... 44
5 Praktická část ... 45
5.1 Extrakce a transformace dat ... 45
5.1.1 Reprezentace transformovaných dat ... 48
5.2 Analýza dat v Power BI ... 49
5.2.1 Obecný pohled na zpracovaná data ... 49
5.2.2 Detailní pohled na zpracovaná data ... 50
5.3 Process Mining ... 53
5.3.1 Příprava dat pro Process Mining ... 53
5.3.2 Zpracování procesních logů ... 56
5.3.3 Procesní mapa ... 58
5.4 Machine Learning ... 62

5.4.1 Data 62
5.4.2 Predikce počtu chyb 63
5.4.3 Evaluace modelů 70
5.4.4 Predikce počtu chyb s ohledem na typ chyby 73
6 Shrnutí výsledků 76
7 Závěry a doporučení 77
8 Seznam použité literatury 78
9 Seznam obrázků 82
10 Seznam tabulek 84

Úvod

S neustálým technologickým pokrokem a vývojem společnosti, roste také množství generovaných a ukládaných dat. Data jako taková, pro nás však nemají žádnou hodnotu, dokud jim nedokážeme dát podobu, které jsme schopni porozumět. Proces extrakce znalostí z dat se nazývá data mining. Data mining v sobě zahrnuje metody a techniky explorace a analýzy datových souborů. To nám pomáhá z dat extrahovat užitečné informace a znalosti. Jsou to právě informace a znalosti, které nám dávají konkurenční výhodu a pomocí kterých jsme schopni přijímat řešení a předvídat následující vývoj.

Každý subjekt, ať už se jedná o podnik či jednice, nepřetržitě generuje a přijímá informace. V této informační době je čím dál složitější třídit, validovat a rozumět informacím, jež nám jsou neustále předkládány. Pro porozumění informacím je zapotřebí znalostí a při vysokém objemu dat, nástrojů a technologií se jedná o klíčovou vlastnost. Významnou roli v tomto procesu rozpoznávání dat hraje čas, a pokud chceme datům a informacím z nich získaných rozumět, musíme využívané metody neustále aktualizovat. S narůstajícím množstvím dat a potřebou jim rozumět a využívat je k dosažení námi zamýšlených cílů roste také množství nástrojů, které je možné využít.

Ve společnosti ŠKODA AUTO, a. s. se využívá stále většího množství moderních nástrojů a technik pro analýzu dat. Tato diplomová práce věnuje pozornost těm, kterých se ve firmě k analýze a práci s daty používá.

Práce se skládá ze dvou částí. První část je teoretická a tvoří ji čtyři kapitoly. První kapitola popisuje dobývání znalostí z databází, data mining a jeho nástroje a techniky. Druhá kapitola se týká nástrojů pro analýzu dat, které ŠKODA AUTO, a. s. používá. Třetí kapitola teoretické části je věnována metodám process miningu a čtvrtá kapitola popisuje principy a techniky strojového učení. Praktická část je zaměřená na aplikaci metod a postupů představených v teoretické části na strojová data z autonomního centrálního skladu ŠKODA AUTO, a. s. Nad daty je nejdříve provedená transformace, analýza a vizualizace. Následně dochází k implementaci process miningu za účelem pochopení procesů probíhajících ve skladu. Závěr praktické části je věnován predikci výskytu chyb v probíhajících procesech pomocí machine learning modelu.

Cíl práce

Cílem této diplomové práce je představení metod a postupů využívaných k transformaci, vizualizaci a analýze strojových dat, jejich následná aplikace a zapojení machine learningových algoritmů na podporu vybraných procesů.

1 Dobývání znalostí z databází

Roku 1936 Alan Turing publikoval článek „On Computable Numbers, with an Application to the Entscheidungsproblem“ [1], ve kterém přichází s myšlenkou počítače, který by mohl vykonávat složité výpočty a zpracovávat velké množství dat. Turing tak položil základy počítačové éry. Od té doby se toho mnoho změnilo. Nyní jsou počítače a data součástí každodenního života člověka, který sám, často nevědomky, přispívá ke zvyšování jejich objemu.

Data uchováváme v různých podobách, různými způsoby a na různých místech. V roce 2020 došlo k výraznému navýšení vytvořených a uložených dat. [2] Je to mimo jiné způsobeno vlivy pandemie COVID-19, kdy jsou lidé často nuceni pracovat, učit se a trávit čas v prostředí svých domovů. Dle výzkumu datové korporace IDC z března 2021 v roce 2020 dosáhlo množství vytvořených či replikovaných dat 64,2 ZB. [2] Společnost odhaduje, že v následujících 5 letech dojde ke dvojnásobně většímu vytvoření dat od doby vynalezení digitálního úložiště. K růstu dat dochází především díky neustálému technologickému rozvoji, růstu ekonomik a nových trendů v různých odvětvích, jako jsou například autonomní automobily, chytré domácnosti a průmysl 4.0. Většina společností se tato nashromážděná data snaží využít ve svůj prospěch a získat tak konkurenční výhodu ve svém odvětví, lépe porozumět svému zákazníkovi či optimalizovat a předvídat výdaje.

S rozvojem informačních technologií a jejich dostupností je nyní možné velké objemy dat zpracovávat, dále je analyzovat a vytvářet statistické modely, které dokážou s určitou přesností predikovat další vývoj zpracovávaných dat. V 80. letech 20. století se tak začíná rozvíjet komplexní přístup (Knowledge Discovery from Databases – KDD), neboli dobývání znalostí z databází. Hana Skalská ve své knize „Data Mining a klasifikační modely“ popisuje KDD jako oblast managementu znalostí, která se zaměřuje na možnosti automatizace vyhledávání znalostí z databází, formalizaci analýzy dat, modelování dat a prezentací výsledků ve formě, usnadňující výběr rozhodovací strategie. [3] Cílem KDD je umožnit automatizovaný přístup k relevantním informacím z dat a podpora dalšího rozhodování na základě výstupů, kterých je dosaženo pomocí metod Data Miningu, který se pokládá za samostatný obor a součást procesu vyhledávání znalostí z dat.

Autoři Oded Maimonl a Lior Rokach považují data mining za jeden z klíčových kroků v procesu KDD. Proces KDD se nejčastěji skládá z devíti kroků, které nejsou pevně definovány a mohou se měnit a upravovat dle potřeby řešeného problému: [4]

1. *Vymezení aplikační oblasti a porozumění cílům* – při vedení procesu vyhledávání znalostí z dat je potřeba porozumět prostředí, ze kterého data pochází a otázkám, které mají být s pomocí dat zodpovězeny.
2. *Výběr a vytvoření data setu, nad kterým bude vyhledávání prováděno* – pokud jsou definované cíle, následujícím krokem je zjistit, která data jsou k dispozici, navrhnout proměnné, či způsob jakým by bylo možné získat doplňující data. Tento krok je z hlediska data miningu velmi důležitý.
3. *Předběžné zpracování a čištění dat* – tato část se zaměřuje na čištění dat, alokaci odlehlých bodů, přijímá se rozhodnutí o způsobu nakládání s chybějícími hodnotami a dochází také k odstranění šumu.
4. *Transformace dat* – v této fázi KDD je kladen důraz na přípravu dat pro data mining. Hledají se znaky, které je možné využít pro predikci. Součástí toho mohou být transformace proměnných nebo metody snížení dimenze (vynecháním či sloučením specifických atributů)
5. *Data mining* – výběr vhodné DM metody se odvíjí zejména od typu dat a cílů KDD, kterých je má být dosaženo. Mezi základní typy data miningu patří například klasifikace, shlukování, diskriminační nebo regresní analýza.
6. *Výběr algoritmu pro data* mining – veškeré předešlé kroky určují strategii. Tento krok je definicí taktiky. Výběr konkrétního algoritmu pro vyhledávání vzorů v datech. Cílem je výběr nejvhodnějšího algoritmu a jeho parametrů.
7. *Implementace algoritmu DM* – tento krok se může opakovat několikrát, než je dosaženo uspokojivých výsledků. Takovému procesu se říká „tuning“ algoritmu, tj. úprava parametrů, např. minimální množství instancí jednotlivých uzlů v rozhodovacím stromu.
8. *Vyhodnocení* – vyhodnocení a interpretace nalezených vzorů získaných z dat a porovnání s definovanými cíli v prvním kroku.
9. *Implementace/využití nalezených znalostí* – využití získaných znalostí a jejich zahrnutí do dalších systémů. Znalosti mohou sloužit jako spoušť pro změnu určitých procesů, kdy je dále možné měřit a analyzovat vliv uplatněných znalostí na změny.

1.1 Data Mining

Data mining se považuje za samostatný obor, který v sobě zahrnuje speciální statistické metody využívané pro důkladnou analýzu velkých datových souborů. Tento pojem lze definovat jako činnost, která využívá metody explorace dat. Tukey aplikaci exploračních metod popsal později spolu s návrhy dalších intenzivních statistických analýz, jejichž cílem bylo vyhledávání nových souvislostí v datových souborech. [3]V oblasti informačních technologií, se termín data mining používá zejména v souvislosti s intenzivní analýzou datových souborů velkého rozsahu.

1.1.1 Metodologie data miningu

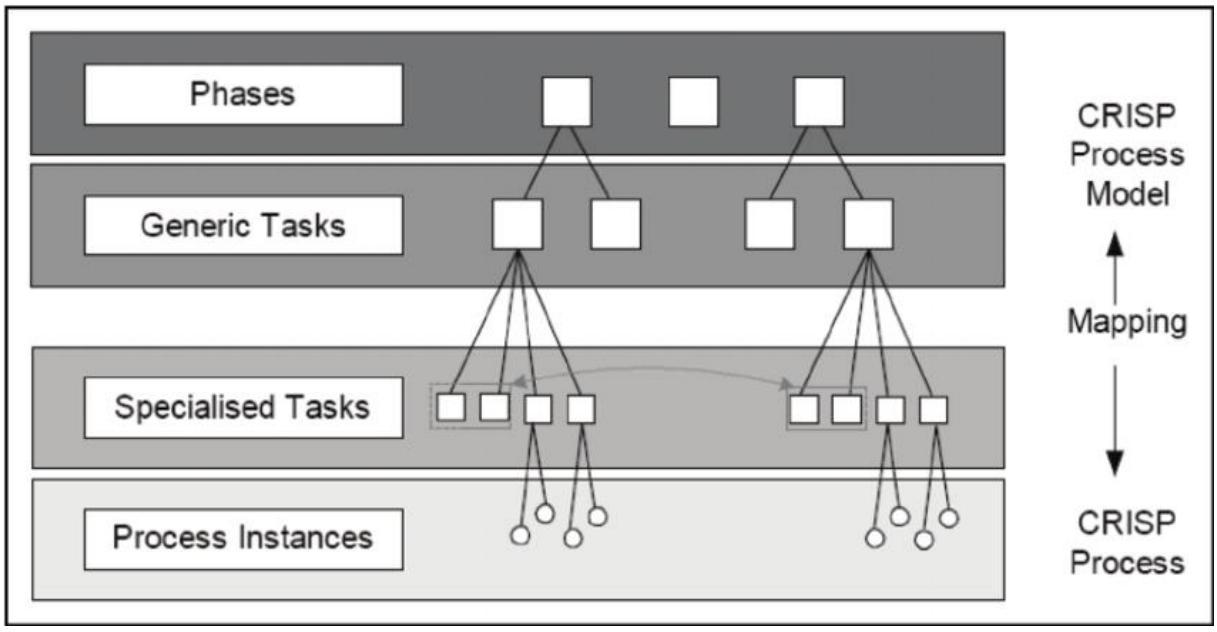
S rozvojem data miningu a zvyšujícím se zájmem firem ho využívat na analýzu dat a následnou podporu při rozhodování, začaly vznikat metodické postupy, které měly za úkol data mining zefektivnit. Tyto postupy se snaží proces data miningu rozdělit do několika fází. Není pravidlem, že kroky daného procesu na sebe navazují. Pořadí, ve kterém jsou představeny se může měnit, nebo opakovat dle potřeb dané úlohy.

Mezi nejznámější a nejpoužívanější metodologie stále patří CRISP-DM (CRoss-Industry Standard Process for Data Mining). Dalším velmi známým postupem je tzv. SEMMA (Sample, Explore, Modify, Model, Assess). SEMMA je používaným přístupem díky rozšířenému používání informačního software SAS Enterprise Miner od společnosti SAS Institute.

Metodologie CRISP-DM vznikla v rámci Evropského výzkumného projektu, jehož cíl spočíval v hledání univerzálního postupu dobývání znalostí z databází, který bude použitelný v nejrůznějších komerčních aplikacích. Takový postup neboli metodologie, by měla nejenom usnadnit a zefektivnit data mining, ale zároveň snížit náklady a čas vynaložené na jednotlivé úlohy.

Metodologii CRISP-DM je možné popsat na základě hierarchické struktury, která v sobě zahrnuje popis úkolů ve čtyřech úrovních:

- Fáze projektu
- Obecné úlohy
- Specializované úlohy
- Aplikace v procesu

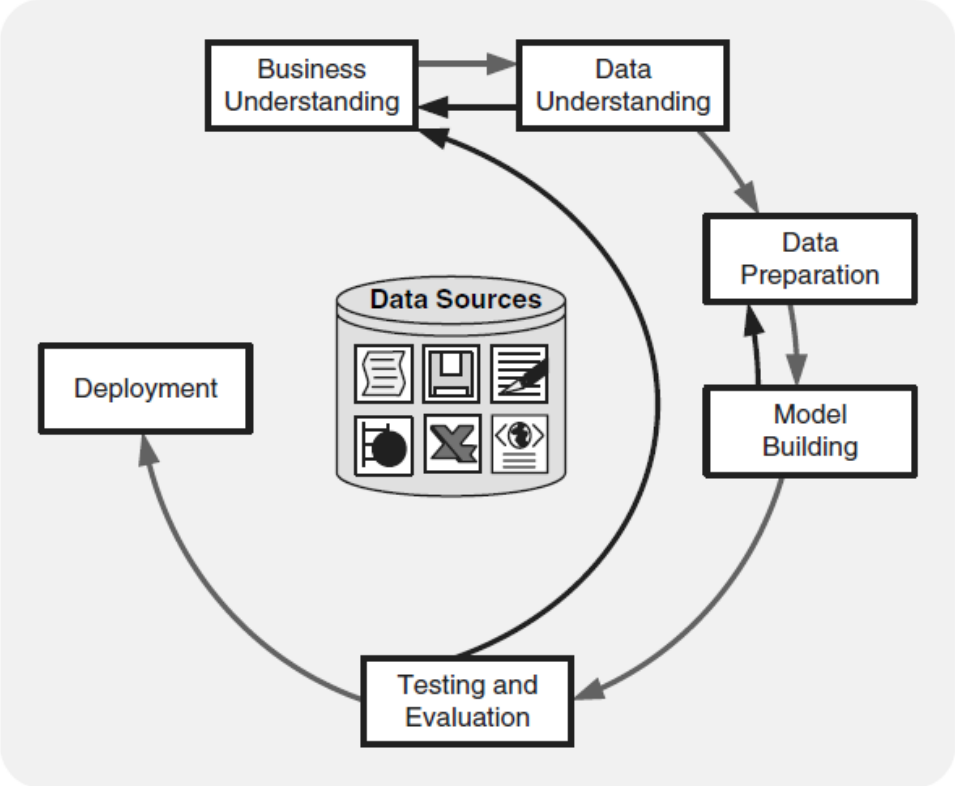


***Obrázek 1**: Hierarchická struktura metody CRISP-DM, převzato z [5]*

Jak je z obrázku 1 patrné, nejvyšší úroveň hierarchické struktury se skládá z několika úloh na druhé úrovni, která popisuje tzv. obecné úlohy projektu. Kroky obsaženy v obecných úlohách jsou zobrazeny na obrázku 1 v dolní části. Tato úroveň musí být obecná na tolik, aby pokryla všechny možné situace data miningu nezávisle na typu projektu.

Na třetí úrovni hierarchické struktury jsou specializované úlohy, které vznikají převodem obecných úloh z druhé úrovně na konkrétní akce podle řešeného problému. *Například na druhé úrovni by mohla být obecná úloha nazvaná očistit data. Třetí úroveň popisuje, v čem se bude tato úloha lišit v různých situacích, například čištění číselné hodnoty versus čištění kategoriální hodnoty nebo zda řešený problém je typu shlukování nebo prediktivního modelu* [6].

Čtvrtá úroveň aplikace v procesu popisuje technickou realizaci konkrétních úloh, evidenci všech činností, rozhodnutí a jejich implementaci. Dokládá, jaké úkony byly vykonány a předkládá dosažené výsledky.



Obrázek 2: *Životní cyklus data miningu, převzato z [7]*

Životní cyklus data mining projektu je v metodologii tvořen následujícími fázemi [4]:

1. Porozumění problému – hlavním výstupem této fáze by měla být definice problému a stanovení cílů, jak z pohledu podniku, tak z pohledu data miningu.

2. Porozumění datům – tato fáze zahrnuje sběr a popis vlastností dat. Důležitou roli zde hraje kvalita dostupných dat.

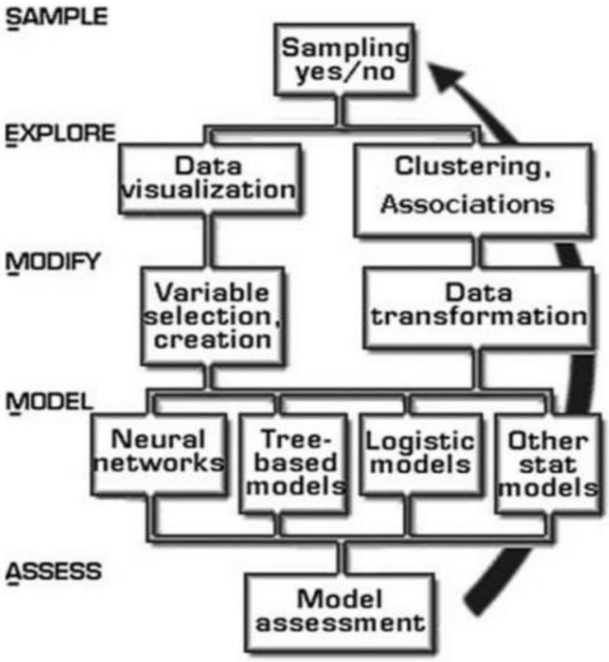
3. Příprava dat – ve fázi přípravy dat dochází k jejich transformaci. Jedná se často o časově nejnáročnější část při práci s daty. Na této úrovni probíhá čištění dat, filtrování, řešení prázdných hodnot, výběr atributů a další aktivity úpravy dat pro usnadnění následného modelování.

4. Modelování – je proces extrakce vzorů z dat a rozhodnutí o vhodném modelu, který bude nejvíce efektivní při práci s danými daty. Využívá se zde matematických a logických modelů za použití algoritmů asociace, rozhodovacích stromů, logické regrese či shlukování.

5. Hodnocení – cílem této fáze je vyhodnocení výstupu modelu. Hodnocení probíhá na základě stanovených kritérií podniku. Je také důležité, aby se pohlíželo na zadané cíle a jejich splnění. Pokud požadavky nebudou naplněné, vrací se celý proces do některého z předchozích kroků. Metody vyhodnocení se odvíjí od typu zvoleného modelu.

6. Využití v praxi – plánování implementace a monitoring. Celý projekt je dokumentován a shrnut v komplexním reportu.

Další zmíněnou metodologií je SEMMA. Ta usnadňuje aplikaci statistických a vizualizačních technik. Klade důraz na výběr a transformaci nejvýznamnějších proměnných, zvolení odpovídajícího modelu pro následnou predikci a přesnost k tomu využívaných modelů. [7]

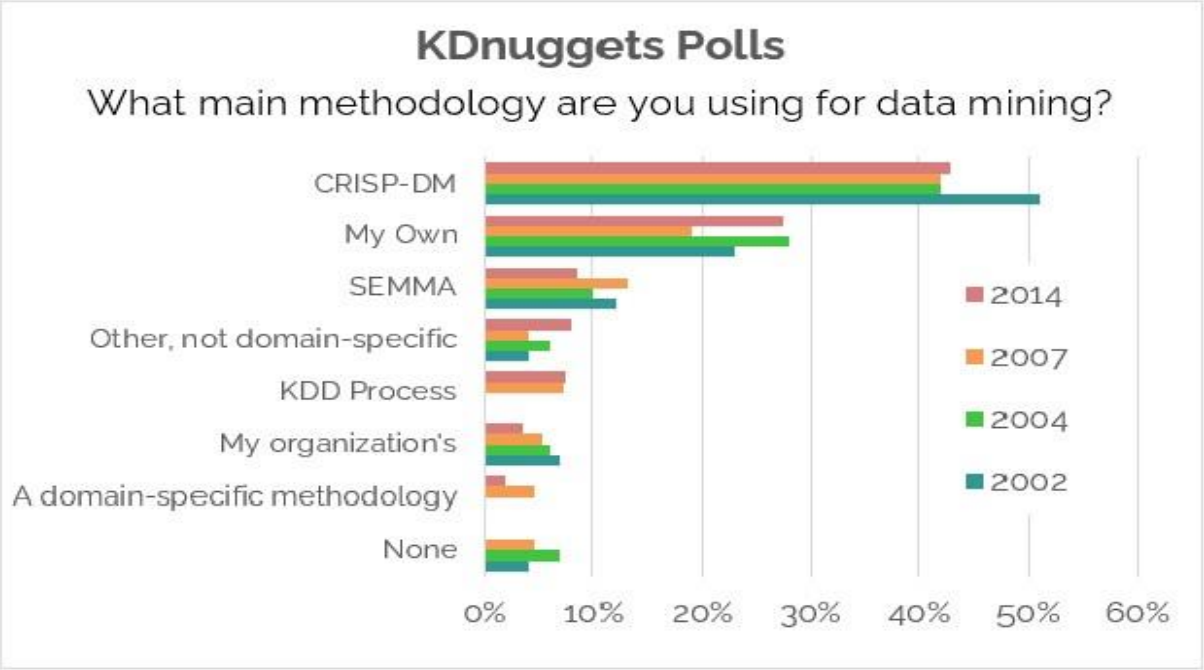


Obrázek 3: *Fáze metodologie SEMMA, převzato z [7]*

Na obrázku výše jsou vyobrazeny jednotlivé fáze metodologie SEMMA. Následuje jejich popis:

1. Sample (výběr vzorku dat) – zde dochází k výběru vzorku dat, který bude dostatečně velký na to, aby se dal považovat za reprezentativní, ale zároveň bylo možné s ním rychle manipulovat. Zde je také možné rozdělit data do patřičných skupin, dle účelu využití, například na data pro validaci, testování či trénování.

2. Explore (explorace) – cílem explorace dat je najít nové, dosud neznámé informace a znalosti. Při exploraci je využíváno popisné statistiky a vizualizace dat. Výstupem této fáze je popis/odhalení trendů nebo závislostí v datech.
3. Modify (transformace) – na základě výsledků explorace probíhá transformace dat. Často se jedná o úpravu počtu proměnných či vytvoření skupin. V průběhu data mining procesu není neobvyklé jednotlivé kroky opakovat, protože může docházet k novým objevům v průběhu procesu.
4. Model (modelování) – tvorba modelu dle dostupných informací a charakteristice dat.
5. Assess (porovnání) – poslední krok data mining procesu dle SEMMA metodologie se zaměřuje na vyhodnocení modelu, jeho přesnosti a kvalitu jim nalezených znalostí a informací. [7]



Obrázek 4: *Popularita jednotlivých metodologií data miningu, převzato z [8]*

Na obrázku 4 je možné vidět výsledky dotazníku prováděného společností KDnuggets, která se dotázala 200 respondentů na jimi využívaný druh metodologie při data miningu. Z výsledků je vidět, že obě zmíněné metodologie se stále těší velkému zájmu.

1.2 Techniky a algoritmy data miningu

Při vyhledávání znalostí z dat, se ve fázi data miningu využívá různých technik a metod. Obecně se tyto metody dělí na deskriptivní a prediktivní. *Deskriptivní metody* popisují vztahy mezi daty a využívají se k analýze chování proměnných a jejich závislostí mezi sebou. *Prediktivní metody* mají za cíl, na základě dostupných dat, odhalit trend a poskytnout odhad pravděpodobného budoucího vývoje. V této kapitole budou představeny techniky a metody, které jsou při úlohách data miningu nejpoužívanější.

1.2.1 Klasifikace a diskriminace

Skalská ve své knize [3] definuje úlohy klasifikace jako „*obecný proces vycházející ze systematické analýzy a syntézy vlastností, které vedou k rozdělení množiny prvků do podmnožin prvků s největší podobností, se společnými znaky, se společnými příčinami vlastností, nebo se shodným vývojem.“*

V případě, kdy nejsou skupiny pro zařazení nových prvků známé, mluvíme o tzv. klasifikaci bez učitele – shlukové analýze. Cílem takové úlohy je nalezení podobných prvků a charakterizace skupin. Hledají se tedy společné vlastnosti a odlišnosti prvků.

Pokud jsou skupiny pro klasifikaci známé, jedná se o tzv. diskriminaci, respektive klasifikaci s učícím souborem. Cílem je najít model/množinu pravidel, na základě kterého, bude možné zařadit nový prvek do již známé skupiny. Předpokladem je existence proměnných, díky kterým je možné od sebe skupiny rozlišit.

Nejpoužívanější klasifikační modely:

- Lineární diskriminační analýza LDA (nelineární nebo neparametrická DA)
- Rozhodovací strom
- Bayesovská klasifikace
- Metoda SVM (Super Vector Machines – Metoda podpůrných vektorů) nelineární separace
- Metody ensemble (několik klasifikátorů)

1.2.2 Shlukování

Shlukování se využívá k identifikaci skupin, které z pohledu datového souboru, obsahují podobné prvky. Shluková analýza si klade za cíl roztřídit prvky do několika, pokud možno nejvíce homogenních skupin – shluků. Prvky uvnitř shluku by si měly být podobné co nejvíce, naopak prvky z různých skupin, co nejméně. Pomocí technik hledání shluků je možné definovat strukturu dat a získat tak informaci o rozdělení a korelaci jednotlivých atributů v datech. K těmto účelům je také možné využít klasifikační metody, které však bývají nákladnější, a proto se techniky hledání shluků často uplatňují k přípravě dat pro výběr skupin atributů a klasifikaci. [9]

Hierarchické aglomerativní metody

- Metoda nejbližšího souseda
- Metoda nejvzdálenějšího souseda
- Centroidní metoda

Hierarchické divizní metody

- MacNaughton–Smithova metoda

Nehierarchické metody

- Forgyova a Janceyova metoda
- MacQueenova metoda (K-means)
- Fuzzy C-means

1.2.3 Predikce (prediktivní analýza, regresní analýza)

Nejpoužívanější technikou pro predikci jsou metody regresní analýzy. Cílem těchto statistických metod je odhadnout hodnotu závisle proměnné pomocí již známých hodnot proměnných nezávislých. Předvídání budoucího vývoje dat u komplexních problémů není jednoduché. Pro takové případy se využívají složitější techniky, jako jsou například logistická regrese nebo neuronové sítě. Některé algoritmy provádí jak klasifikační, tak i regresní model. Jako příklad je možné zmínit CART (Classification and Regression Trees) algoritmus, jehož výsledkem jsou klasifikační a regresní stromy. [4]

Typy regresních metod:

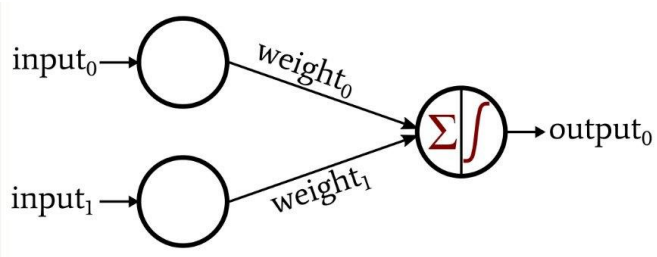
- Lineární regrese
- Mnohonásobná lineární regrese
- Nelineární regrese

1.2.4 Asociační pravidla / Asociační analýza

Cílem asociační analýzy je nalezení asociačních pravidel v datovém souboru. Asociační pravidlo je definováno výrazem X → Y, kde X je skupinou prvků a Y je jeden prvek, se kterým jsou ve vztahu. [7] Nalezení asociací a korelací mezi jednotlivými prvky je možné získat cenné informace o zpracovávaných datech, které je dále možné využít pro podporu rozhodování. Nejvhodnějším příkladem využití asociační analýzy, je analýza nákupního košíku, kterou popularizátor této metody Rakesh Agrawal, rozebírá ve své publikaci [10]. Cílem analýzy nákupního košíku je nalézt vzájemné vztahy (asociace) mezi jednotlivými produkty, které zákazníci nakupují v obchodě. Nejznámějšími algoritmy pro vyhledávání asociačních pravidel jsou algoritmy Apriori a CARMA.

1.2.5 Neuronové sítě

Neuronové sítě jsou v dnešní době jedny z nejpoužívanějších klasifikačních modelů napříč různými odvětvími. Jejich architektura je inspirovaná lidským mozkem, který k přenosu informací využívá sítě nervových buněk (neuronů) propojené axony.



Obrázek 5: *Perceptron, převzato z [11]*

Základním typem umělého neuronu je perceptron, který přijímá několik binárních vstupů $\{x_1, x_{2...}\}$ a produkuje jediný binární výstup, kterého je dosaženo pomocí sumy součinu vstupů s množinou vah $\{w1, w2, ..., wi\}$. Síť se pak učí úpravou vah tak, aby byla schopna předpovídat správné zařazení prvků do tříd. [12] Neuronové sítě jsou vhodné pro řešení komplexních úloh, při kterých mohou být aplikovány za účelem odhalení trendů v datech a následné predikci.

1.3 Analýza časových řad

Analýza časových řad je součástí praktické části této práce, a proto bude tato kapitola rozebrána podrobněji. Modely využívané k predikci časových řad jsou popsány v kapitole o strojovém učení.

Časovou řadu můžeme definovat jako posloupnost hodnot náhodné veličiny, v časových intervalech. Tato posloupnost je uspořádaná chronologicky, s rovnoměrnými časovými intervaly mezi pozorováními. Časovou řadu je možné zapsat následujícím způsobem:

$$y_{t1}, y_{t2} \ldots . y_{tn} \quad \text{kde } y_t \text{ je hodnota v čase } t = 1 \ldots . n$$

Analýza časových řad spadá do studií stochastických procesů, kdy se kromě jevů náhodných veličin zkoumá i jejich časový vývoj. S reprezentací dat formou časových řad se můžeme setkat v přírodních vědách v podobě údajů o výši naměřených teplot za roční období, sociologii, kdy sledujeme počty narozených dětí v daném roce, nebo ekonomice a finančnictví, při získávání informací o vývoji kurzu měn, ceny akcií, růstu HDP apod.

Cílem analýzy časových řad je porozumění mechanismu generování hodnot časové řady a vytvoření modelu, který s určitou přesností poskytne vhled do pozorování historických hodnot. Takový model je pak použit pro popis a analýzu struktury časové řady a predikci budoucího vývoje hodnot pozorování. [13]

1.3.1 Dekompozice časových řad

Časové řady jsou obecně považovány za směs rušivých a užitečných komponent. Při modelování je nutné separovat užitečnou složku od náhodného rušení, kdy je užitečná složka tvořená ještě trendovou, sezónní a cyklickou složkami. Takový proces se nazývá dekompozicí časové řady.

Trend

Obecná tendence časových řad k stoupání, klesání nebo stagnaci v nějakém konkrétním intervalu je považována za trendovou složku, a proto je možné říci, že trend představuje tzv. proces dlouhodobých změn. Pokud hodnota časové řady delší dobu kolísá kolem určité hodnoty, jedná se o konstantní trend.

Sezónnost

Je typickou vlastností časových řad z oblasti ekonomie a vyjadřuje pravidelně se opakující odchýlení od trendové složky v rámci nějakého časového období. Za období se nejčastěji volí rok, čtvrtletí, týden apod.

Cykličnost

Cyklická složka je definována jako výkyv trendu způsobený dlouhodobými cyklickými událostmi s délkou delší než jeden rok. Cyklus je ve statistice definován jako dlouhodobý výkyv s neurčitou periodou. Cyklická složka časové řady se někdy spojuje s trendovou složkou jako součást tzv. střednědobého trendu nebo střednědobé tendence vývoje, která má obvykle oscilační formu a neznámou proměnlivou periodu.

Náhodná složka

Náhodná, nebo také reziduální složka v časových řadách je tvořená náhodnými jevy, které se neopakují podle definovaného pravidla či vzoru. Takové výkyvy v datech mohou být způsobeny nečekanými událostmi jako například válkou, demonstrací, povodní apod. [14]

2 Nástroje pro analýzu a vizualizaci dat

Software pro vizualizaci a analýzu dat by měli zahrnovat veškeré funkce, které umožní získat potřebné informace z nestrukturovaného datového souboru.

Tyto funkce můžeme rozdělit na:

- Získání dat ze zdroje
- Syntaktickou analýzu dat (parsing dat nebo zpracování dat)
- Indexace zpracovaných dat
- Vizualizace dat

2.1 Získání dat ze zdroje

Získávání strojových dat ze zdroje zajišťují „programy“, které jsou schopny data z jejich zdroje dopravit do prostředí, kde dohází k jejich dalšímu zpracování. Odlišné systémy mají odlišný způsob, jakým zachází s daty, která generují. Dělí se na tři základní kategorie dle použité architektury:

Push-based – znamená, že zdroj je schopen ukládat zprávy na lokální disk, nebo je samostatně dopravit po síti tam kam je potřeba. Je také možné využít tzv. forwarderů, které jsou instalovány/umístěný co možná nejblíž ke zdroji dat a starají se o transport generovaných dat.

Pull-based – jsou software, které dokážou získat data přímo ze zdroje.

Search in place – jedná se o způsob, kdy data není potřeba nikam přepravovat. Příkladem takového přístupu je model map-reduce. Výhodou takového přístupu je možnost nahlížet na data v reálném čase, obzvlášť pokud se jedná o analýzu pouze části dat. Nevýhodou takového modelu pak můžou být zvýšené nároky na výkon datového zdroje.

2.2 Syntaktická analýza dat

Strojová data jsou generována v nestrukturované podobě, a proto jim není vždy snadné porozumět. V případě, kdy je zapotřebí s daty dále pracovat a získat z nich užitečné informace, je nutné provést patřičnou transformaci. Syntaktickou analýzu neboli parsing, je možné

provádět před, nebo po indexaci dat. Parsing je nezbytnou součástí procesu analýzy strojových zpráv. Jedná se o techniku, při které jsou textové řetězce uspořádány dle předem definovaných vzorů. Příchozí text je porovnáván se vzory a v případě shody je možné s textem manipulovat různými způsoby. Tyto vzory nám dovolují mimo jiné například odstranit nehodící se znaky z obdržené zprávy či přeměnit strukturu textu.

Nejznámějšími nástroji pro parsing a analýzu strojových dat jsou:

- Programovací jazyky – Programovací jazyky jako např. Python či C/C++ disponují knihovnami, které jsou učené pro transformaci dat a poskytují možnost data rychle upravovat.
- Splunk – Placený softwarový produkt, který poskytuje nástroje pro sběr a analýzu dat. Splunk parsuje data před jejich indexací.
- Fluentd – Open source kolektor dat, který v sobě zahrnuje i určité filtry a pluginy, kterých se dá využít k lepší strukturaci příchozích dat.
- Graylog – Open source robustní řešení pro management logových zpráv s možností parsingu a úpravy vstupních logových zpráv.
- Logstash – Logstash je open source modul, který dokáže sbírat data z více zdrojů a následně je odesílat tam, kam je potřeba. Mimo přepravy logových zpráv nabízí také jejich úpravu na vstupu. Je tak možné změnit příchozí zprávu do potřebného formátu, který bude vhodný pro uložení do databáze a další analýzu.

2.3 Indexace a vizualizace dat

Ať už data prošla procesem parsingu, nebo zůstávají v nestrukturované podobě a budou upravená až v případě potřeby, jsou ukládána do databází. Nejpoužívanějšími databázovými systémy jsou stále relační databáze. Princip relačních databází bude v této práci stručně popsán, stejně jako další typ databázových systémů pro ukládání dat, tzv. NoSQL databáze, které jsou často využívány pro indexaci velkého množství nestrukturovaných dat.

2.3.1 Relační databáze

Relační databáze (relational database management systems, RDBMS) ukládají data v podobě n-tic organizovaných do relací. Vyhledávání je realizováno prostřednictvím operací relační algebry. Základní reprezentací n-tice v relační databázi je řádek v tabulce. Důvodem

je nepřehlednost zápisu pomocí množin. Jeden řádek v tabulce představuje jeden záznam v databázi. Řádky v tabulce se skládají z několika polí neboli sloupců, které mají pevně stanovené datové typy. Nad daty uloženými v relační databázi jsou prováděny dotazy pomocí Structured Query Language (SQL). Díky tomuto standardizovanému vyhledávacímu jazyku pro relační databáze je možné data do databáze vkládat a následně se na ně dotazovat.

Jedním z nejdůležitějších pojmů týkajících se relačních databází jsou transakce a jejich tzv. ACID architektura. Transakce se může skládat z několika procesů. Ty jsou provedeny všechny, nebo žádný. Výsledkem vykonání transakce je změna v databázi například v podobě vložení nových dat do databáze.

2.3.2 NoSQL databáze

S neustálým nárůstem množství dat a rychlostí s jakou k takovému generování dochází se mění i přístupy, kterých je využíváno pro nakládání s daty. Změny nastávají také na straně uživatele a společností, které chtějí z dat vytěžit co nejvíce informací, v co nejkratším čase. Systémy pro ukládání dat jiným než klasickým relačním způsobem, se obecně označují jako NoSQL databázové systémy. Většina těchto databázových systémů v dnešní době vzniká pro podporu efektivního zpracování Big Data. Tento pojem definuje například společnost IBM takto: „V závislosti na odvětví a organizaci zahrnují Big Data informace z interních a externích zdrojů, jako jsou transakce, sociální média, podniková data, senzory a mobilní zařízení. Firmy mohou tato data využívat, aby lépe přizpůsobily své výrobky a služby potřebám zákazníka, dále optimalizovaly provoz a infrastrukturu a/nebo nalezly zcela nové zdroje příjmů“. [15]

Většina NoSQL databází oproti relačním databázím nevyužívají pro své transakce ACID pravidel, nýbrž přístup BASE. Jedná se opět o zkratku, která je složená ze začátečních písmen následujících pojmů Basically available, Soft-state, Eventually consistent.

- Basically available – Většina dat je dostupná po většinu času, nemůže se stát, že systém přestane fungovat úplně.
- Soft-state – Oproti ACID commitu, kdy jsou všechny úpravy v databázi úspěšně potvrzeny a uloženy v BASE principu tomu tam nemusí být vždy. Může se tak stát, že různým uživatelům ve stejnou dobu můžou být zobrazena jiná data.
- Eventually consistent – Databáze se převážně nachází v konzistentním stavu, avšak dovoluje i částečnou nekonzistenci dat za účelem zvýšit dostupnosti dat a rychlosti

dotazování. Může se tak stát, že různí uživatelé v určitém čase (inconsistency window) vidí odlišná data. [16]

Výhody NoSQL databází:

1. Flexibilní škálovatelnost: relační databázové systémy využívají tzv. vertikální škálovatelnost, což znamená nasazení výkonnějšího hardware ve snaze navýšit kapacitu databáze. NoSQL databáze škálují horizontálně. Zpracování každé úlohy může být distribuováno v rámci množiny serverů, které je možné rozšiřovat přidáváním dalších serverů v rámci clusteru.
2. Flexibilní datový model: NoSQL databáze nevyžadují určení žádného databázového modelu anebo je schéma dat volné.
3. Efektivní čtení: NoSQL databáze nevyžadují pevné datové schéma, což nám dovoluje vytvořit datovou strukturu tak, aby nám dovolovala co nejlépe odpovídat na často kladené dotazy.
4. Ekonomický aspekt: Značnou výhodou je samozřejmě i finanční náročnost správy databázového systému. Díky horizontální škálovatelnosti NoSQL databází není od serverů, na které ukládáme data, požadováno velkého výpočetního výkonu. Je tedy možné v clusteru využít relativně levných počítačů.

Nevýhody NoSQL databází:

1. Standardizace přístupů k datům: Relační databáze mají jasně standardizovaný přístup k datům prostřednictvím jazyka SQL. S každou nově implementovanou NoSQL databází je potřeba se učit novou syntaxi.
2. Robustnost: NoSQL databáze mezi námi nejsou tak dlouho jako relační databáze a na rozdíl od relačních databází nedosahují léty prověřené robustnosti.
3. Uživatelská podpora: Myšlenka open source nám poskytuje neustálý vývoj v odvětví NoSQL databází. Bohužel ne všechny projekty nabízí dostačující uživatelskou podporu ve srovnání s relačními databázemi.

2.4 Splunk

Splunk je komerční produkt, [17] který je schopen indexovat příchozí data a následně v nich vyhledávat. Využívá se zejména pro indexaci strojových dat a jejich analýzu. K dotazování nad uloženými daty Splunk využívá vlastní Splunk Processing Language (SPL) dotazovací jazyk.

SPL dovoluje uživateli vytvářet dotazy dle formátu dat, hledat souvislost mezi daty a následně prezentovat nalezené informace pomocí tabulek a grafů různého druhu.

Splunk umožňuje sběr dat z lokálního, nebo vzdáleného zdroje, který disponuje Splunk klientem. Indexovat je možné data různého typu, ale nejlépe pracuje s daty ve formátu csv, logy operačního systému syslog a Apache web-server errors. SPL pracuje pouze s textovými daty. Není tedy možné číst binární soubory, ve kterých ukládají své logy například Wireshark nebo Microsoft Event Viewer.

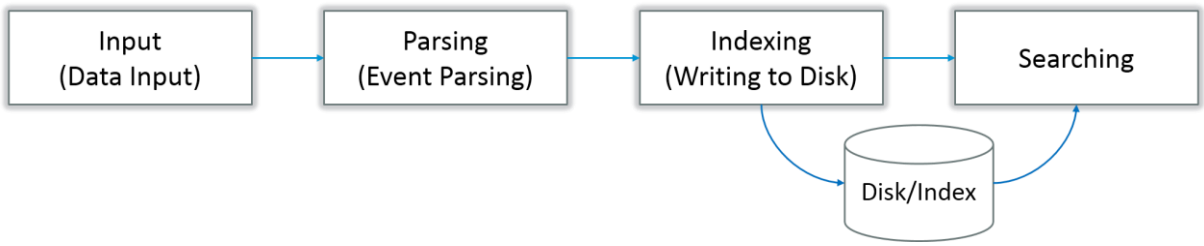
I přes to, že je tento Splunk soustředěn na indexaci dat a vyhledávání v nich, jedná se o OLAP (Online Analytical Processing). OLAP je považován za jeden z nástrojů Business Intelligence a slouží k rychlé analýze dat v téměř reálném čase. Většina nástrojů pro analýzu logů nebo strojových dat, jako je např. Splunk, Logstash či Fluentd jsou považovány za OLAP systémy. [18]

2.4.1 Architektura Splunk

Abychom lépe pochopili, jak Splunk funguje, je potřeba se podívat na jednotlivé fáze zpracování dat a jejich provedení.

Jak již bylo zmíněno v předchozí kapitole o nástrojích pro analýzu a vizualizaci dat, funkce, které daný nástroj musí být schopen vykonat, jsou: získání dat ze zdroje, parsing dat, indexace zpracovaných dat a následná vizualizace. Z pohledu Splunk software mluvíme o třech fázích, které v sobě tyto funkce zahrnují:

- Fáze získání dat
- Fáze indexace dat
- Fáze vyhledávání v datech



Obrázek 6: *Splunk, fáze zpracování dat, převzato z [19]*

Fáze získání dat

V této fázi Splunk přijímá data ze zdroje v hrubé podobě a rozděluje je do bloků, kdy každý blok opatří metadaty. Metadata obsahují informace jako například hostname, zdroj nebo typ zdroje dat. Do informací o blocích můžeme zařadit i typ kódování znaků či hodnoty, díky kterým je ve fázi indexace rozhodnuto, do jakého indexu se daný blok dat uloží.

Fáze indexace dat

Tato fáze se skládá ze dvou částí: parsingu a indexace:

1. Ve fázi parsingu Splunk transformuje data do podoby, ve které bude vyhledávání v datech zjednodušené. V terminologii Splunku se jedná o event processing, jelikož během tohoto procesu Splunk rozdělí data do tzv. individuálních eventů.
   Tato fáze se skládá z několika následujících kroků
   a. Rozdělení datového proudu do jednotlivých řádků
   b. Identifikace, parsing a vytvoření timestamp
   c. Vytvoření metadat jednotlivých eventů
   d. Transformace eventů a metadat prostřednictvím regex pavidel
2. Ve fázi indexace Splunk uloží již zpracovaná data do indexu na disku. Výhodou indexace je rychlé vyhledávání jednotlivých částí dat.

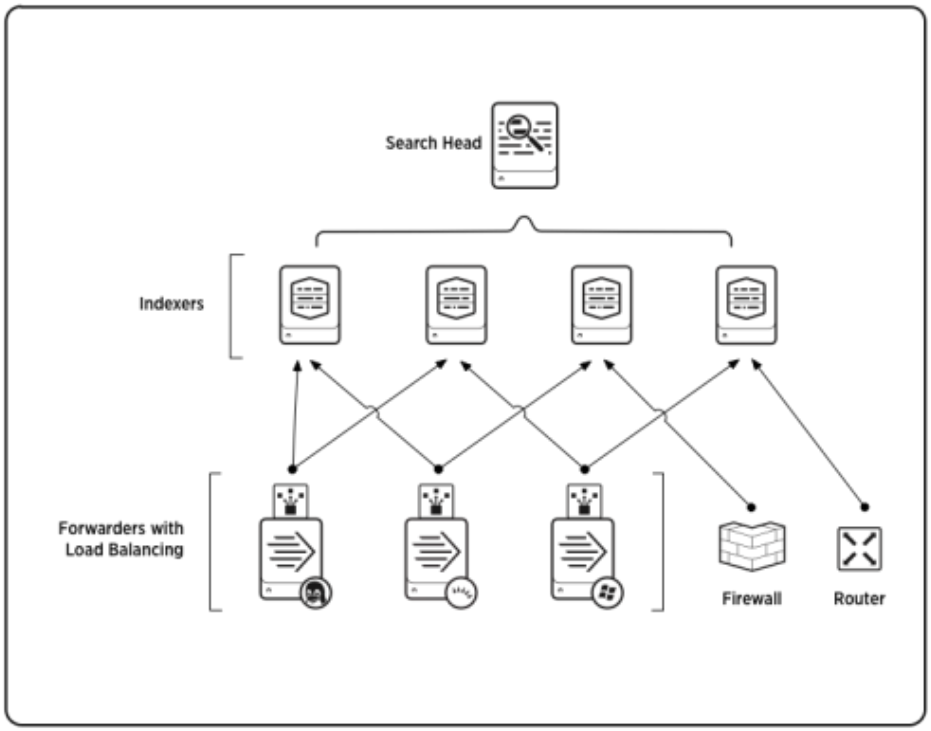
Fáze vyhledávání v datech

Tato fáze se zabývá uživatelským rozhraním a tím, jak uživatel může v datech vyhledávat, získávat informace a vytvářet z nich grafy, dashboardy, reporty a upozornění na definované situace.

Komponenty

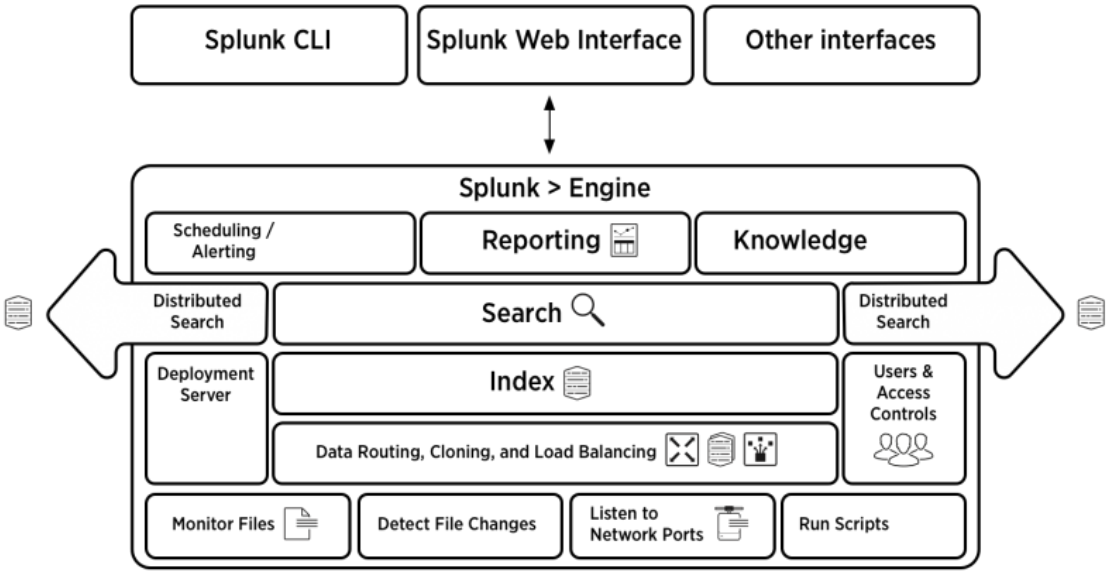
Zmíněných fází, ve kterých Splunk poskytuje uživateli pohodlí při práci s daty a získávání užitečných výstupů je dosaženo pomocí tří základních komponent:

- Forwarder – jedná se o nezávislou instanci Splunk software, která zajišťuje transport dat ze zdroje do vzdáleného úložiště, kde následně dochází k parsingu a indexaci dat. Tímto způsobem je možné sledovat dění na několika zařízeních, na která se forwarder nainstaluje. Díky velmi nízkým výpočetním nárokům je možné forwardery umístit na tisíce vzdálených zdrojů v reálném čase. Společnost Splunk ve svých testech dokázala zpracovat data až z 5 tisíc zdrojů. [20]
    - Universal Forwarder – tento typ forwarderu neprovádí žádnou úpravu dat před jejich indexací pomocí indexeru. Místo toho posílá data v podobě, ve které je získá od zdroje.
    - Heavy Forwarder – pomocí tohoto typu forwaredu je možné data vyfiltrovat, provést základní parsing a až poté data transportovat do indexeru. Sníží to čas i rychlost uložení dat, ale bude to mít větší nároky na zdroj.
- Indexer – je komponenta, která indexuje a ukládá data přicházející od forwarderu. Právě indexer dělá z příchozích dat tzv. eventy. Pokud se jedná o universal forwarder, indexer nejdříve provede parsing příchozích dat a až poté je uloží. Pokud data přichází od heavy fowraderu, indexer je pouze zaindexuje. Při indexaci se vytvoří několik souborů, které se umístí do jednotlivých adresářů a rozdělí na tzv. buckety. Soubory vznikající při indexaci:
    - Soubor hrubých dat po kompresi
    - Indexy, které odkazují na hrubá data (zvyšují rychlost vyhledávání)
    - Metadata – informace o formátu, obsahu, velikosti apod.
- Search Head – představuje uživatelské rozhraní pro komunikaci se Splunkem. Pomocí této komponenty je možné v datech vyhledávat, vytvářet dotazy, dashboardy, grafy a reporty. Tuto komponentu je možné instalovat zvlášť na vzdálený server pomocí povolení splunkweb služby. Instance Splunku může být jak search head tak search peer, kdy první ze zmíněných odpovídá pouze za vyhledávání v datech, kdežto search peer vykonává indexaci a komunikuje s dalšími instancemi typu search heads.



Obrázek 7: *komponenty, převzato z [21]*

Na obrázku 7 je vyobrazeno základní nastavení, které získáme při instalaci Splunk Enterprise na jeden stroj. Prostřednictvím webového rozhraní je možné nastavit zdroje dat, v tomto případě jsou jimi servery s operačními systémy Linux, Solaris a Windows, firewall a router. Data jsou indexována zvlášť dle nastavených pravidel. Veškerá vstupní data je možné analyzovat, vytvářet grafy a reporty na stejné instanci. [19]



Obrázek 8: *Splunk, vlastnosti, převzato z [22]*

Obrázek 8 popisuje veškeré vlastnosti, které Splunk uživateli nabízí. V horní části je možné vidět možnosti komunikace se Splunk instancí, dále rozdělení jednotlivých funkcí analýzy a vyhledávání v datech, od indexace přes vizualizaci a reporting, až k využití programovacích jazyků pro vytvoření statistických modelů či predikce z dat.

Výhody software Splunk: [18]

- Velmi rozsáhlé funkce pro analýzu a monitorování strojových dat
- Kvalitní a přívětivé uživatelské rozhraní
- Rychlost a šetrné požadavky na výpočetní výkon
- Možnost vytváření a ukládání dashboardů s grafy a funkcemi
- Podpora programovacích jazyků a Machine Learning
- Množství rozšiřujících pluginů
- Možnost vytváření alarmů na definované situace.

Nevýhody Software Splunk:

- Vysoké náklady v případě využívání Splunku v jeho plné implementaci
- Vysoká komplexita nástroje
- Složitost debuggingu a dotazovacího jazyka
- Potřeba vytvoření dobré architektury a znalosti na správu software
- Omezení na množství zpracovávaných dat

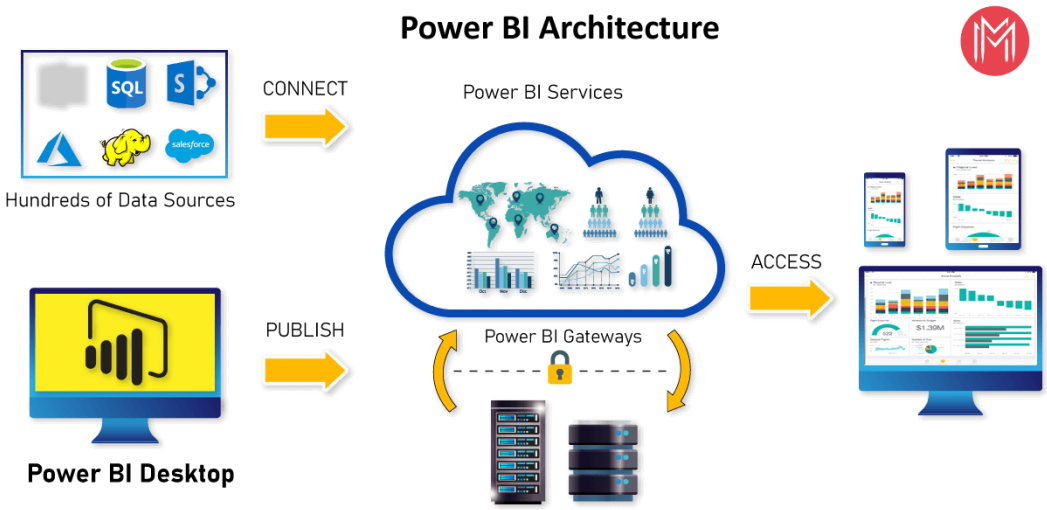
2.5 Microsoft Power BI

Power BI je sada nástrojů a služeb, které umožňuji jejich uživatelům pohodlněji pracovat s daty a získávat tak cenné informace. Jak již z názvu vyplývá, jedná se o Business Intelligence platformu, dostupnou široké veřejnosti od roku 2015. Software poskytuje široké množství různých analytických metod, od vytváření jednoduchých grafických vizualizací, přes sestavení komplexních dashboardů či reportů, až po možnosti implementace predikcí a programování. Power BI dokáže sbírat velké množství dat z více než sta různých datových zdrojů, jako jsou databázové systémy, nebo soubory odlišných datových typů. Je také možné jako vstup definovat skripty programovacích jazyků Python, nebo R. Po definici zdroje jsou data reprezentovaná v podobě tabulek jejichž řádky představují jednotlivé záznamy, podobně jako je tomu v případě relačních databází, se kterými je možné následně pracovat.

2.5.1 Power BI architektura

Power BI je dostupný ve třech následujících podobách: [23]

- Power BI Desktop – jedná se o implementaci software, která poskytuje uživateli nejvíce funkcí, a proto se využívá zejména v prvním kontaktu se zdrojem dat, kdy je potřeba data nahrát, transformovat a připravit na vytvoření vizualizací v reportu. Report představuje v terminologii Power BI kolekci vizualizací, jako jsou grafy, tabulky, interaktivní panely apod.
- Power BI Mobile – tato mobilní aplikace neposkytuje uživateli modifikaci dat a dovoluje uživateli pouze sledování již vypracovaných vizualizací.
- Power BI Service – webová aplikace Power BI, která se využívá zejména ke sdílení, vytváření dashboardů a úpravu vizualizací. V případě, kdy společnosti mají obavy ze sdílení svých dat na cloudu, Microsoft nabízí zabezpečenou možnost Power BI Report Server. Tato platforma splňuje požadavky firem na bezpečnost sdílení dat.



Obrázek 9: *Architektura Power BI, převzato z [24]*

Na obrázku výše je stručně popsána možnost využití Microsoft Power BI, kdy se komplexní analýza dat provádí prostřednictvím Power BI Desktop, následně se výstupy sdílí prostřednictvím cloudového řešení Power BI Service, kde je také možnost využití tzv. Power BI Gateway. To slouží k propojení cloudu s on-premise datových zdrojů typu DirectQuery, Import, Live Query. Poté jsou data zpřístupněny dalším uživatelům.

2.5.2 Hlavní funkce Power BI

Jak již z předchozí kapitoly vyplývá, Microsoft Power BI má mnoho nástrojů, kterých může uživatel využít pro práci s daty, vytváření analýzy a její následnou prezentaci a sdílení. Všechny tyto funkce je možné rozdělit do tří kategorií:

- Transformace a vizualizace dat – primární funkce/smysl software je získávání informací z dat a jejich různých zdrojů. Velmi vhodným nástrojem pro práci s daty je tzv. query editor, který nabízí různé, již předem definované, funkce pro předběžnou úpravu dat. Po transformaci dat je možné přistoupit k jejich vizualizaci a prezentaci pomocí dashboardů. Díky dashboardům je možné upravovat práva uživatelů, kteří mají přístup k reportům a přehled o provedených změnách v jednotlivých reportech.
- Sdílení výstupů – v případě sdílení výstupů je třeba klást velký důraz na bezpečnost. Jednou ze zajímavých funkcí pro sdílení, je již výše zmíněný Power BI On-Premise Data Gateway. Pomocí této funkce je možné vytvořit permanentní zabezpečené spojení mezi zdrojem dat a Power BI reporty. To zajistí aktualizaci reportů v případě nových, nebo pozměněných zdrojových dat. Kompatibilita s Azure dává uživatelům možnost

využít nástrojů jako např. vytvoření datových skladů či využití machine learning technologií pro analýzu dat.

- Pokročilé nástroje pro práci s daty a vyhledávání – Power BI nabízí také pokročilé funkce práce a analýzy dat, jako jsou využití knihoven programovacích jazyků Python a R. Díky integraci těchto nástrojů je možné využívat statistických modelů pro predikci budoucího vývoje dat. [25]

2.5.3 Digital Analysis Expressions (DAX)

Microsoft SQL Server Analysis Services (SSAS) a Microsoft Power Pivot používají programovací jazyk DAX. Vznikl v roce 2010, kdy byl původně vydán PowerPivot pro Excel 2010, kdy se ještě název skládající ze dvou slov psal dohromady. Komunita Excelu, která používá DAX k vývoji datových modelů Power Pivot v Excelu, a komunita Business Intelligence (BI), která používá DAX k vytváření modelů pomocí SSAS, postupem času rostly a tento jazyk využívaný k vyjadřování vzorců nabral na populárnosti. [26]

DAX se využívá v Power BI, pracuje na základě datového modelu, kdy se data reprezentují jako vzájemně propojené tabulky. Součástí vzorců tohoto jazyku jsou funkce, operátory a hodnoty, které dovolují nad daty provádět pokročilé výpočty a dotazovat se na data v datovém modelu. [27]

Jedním ze zajímavých konceptů jazyka DAX jsou tzv. míry. Jedná se o vzorce dynamických výpočtů, které je možné aplikovat na data při vytváření reportů. Míry jsou vhodné pro úpravu prezentovaných dat, kdy je potřeba využít filtrování určitých atributů mezi jednotlivými tabulkami. Ve vzorci míry se využívá standardních funkcí agregace, jako je tomu například při vytváření funkcí typu COUNT nebo SUM, nebo je možné definovat vlastní. Jednotlivé míry mohou být použity ve vzorcích jiných měr.

Kromě měr DAX disponuje mnoha dalšími zajímavými komponenty a funkcemi. Ty však nejsou předmětem této práce a nebude jim, proto zde věnována pozornost.

3 Process Mining

Process mining je množina metod, které se zaměřují na získávání užitečných poznatků z dat generovaných prováděnými procesy. Process mining spojuje procesní (řízení obchodních procesů a operační výzkum) a datovou (dolování a analýza dat) vědu, díky čemuž je možné provádět důkladnou analýzu procesů z dostupných dat. Metod process miningu je možné využít v jakémkoliv odvětví, které disponuje daty reprezentujícími probíhající procesy, ať už je řeč o logistice, financích nebo zdravotnictví. [28]

Jedním z důvodů rostoucího zájmu o implementaci process miningu je neustálý růst ukládaných dat v podobě tzv. event logů, které v sobě obsahují informace o určité činnosti ve výpočetním systému a mohou tak vést k informacím o již provedených procesech. Druhým významným důvodem zvýšené pozornosti process miningu je potřeba pochopení a optimalizace probíhajících procesů v rychle se měnícím konkurenčním prostředí.

Následující tabulka popisuje případy a s nimi spojené dotazy, které je možné řešit s pomocí process miningu. [29]

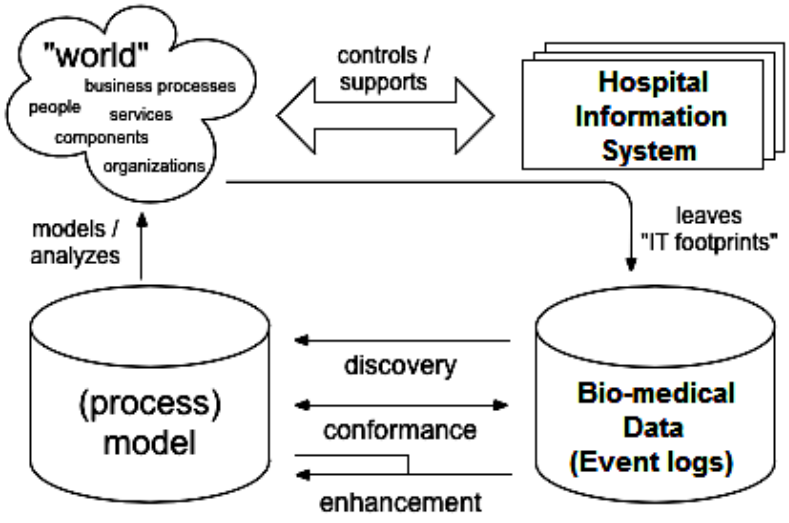
Use Case	Otázky	Typ
Detekce business procesů	Který proces popisuje danou aktivitu?	Koherence
Hledání slabých míst v procesu	Místa v procesu, která ho zpomaluji a co taková místa způsobuje?	Produktivita
Nalezení odchylek v procesu	Kde se skutečný proces odchyluje od očekávaného procesu? Proč k těmto odchylkám dochází?	Soudržnost
Optimalizace procesů	Jak urychlit proces? Jak ho provést na co nejnižší počet kroků?	Produktivita
Předvídání problémů v procesu	Je možné předvídat výskyt odchylek, zpoždění a rizik v procesu?	Produktivita/Koherence

***Tabulka 1**: Process Mining Use Cases [vlastní zpracování]*

3.1 Typy Process Miningu

Wil van der Aalst ve své knize Process Mining Data Science in Action [29] popisuje tři hlavní typy process miningu, které za pomocí event logů a použitých technik umožňuji lépe porozumět probíhajícím procesům v informačním i reálném světě.

1. Discovery – Tato technika vytvoří model bez použití předchozích informací. Model je tedy vytvořen pouze na základě dostupných event logů neboli záznamů o určité události, popisujících aktivitu v systému. Příkladem takového modelu může být vytvoření sociální sítě, která může reprezentovat spolupráci lidí ve společnosti.
2. Conformance (shoda) – Při tomto typu process miningu dochází k porovnání již existujícího modelu s event logy stejného procesu. Probíhá také ověřování shody modelu s informacemi, které v sobě obsahuje event log. Jako příklad je možné uvést tzv. kontrolu čtyř očí, kdy je v modelu definováno pravidlo, že prováděná činnost je kontrolována druhou osobou. Díky zkoumání událostních logů pomocí pravidel definovaných v modelu je možné odhalit případné porušení těchto pravidel.
3. Enhancement (vylepšení) – Myšlenkou tohoto typu process miningu je zlepšení, nebo úprava existujícího procesního modelu pomocí informací z event logu o aktuálním procesu. Protože je hranice mezi realitou a modelem měřená kontrolou shody, třetí typ process miningu se soustředí na změnu modelu za účelem lepší reprezentace reality. Například se může jednat o úpravu modelu, pokud se posloupnost probíhajících aktivit neshoduje s realitou.



Obrázek 10: *Aktivity process miningu, převzato z [29]*

3.2 Procesní log

Na obrázku níže je zobrazena tabulka s procesními logy. Každý řádek v tabulce představuje jednu konkrétní událost. Procesní log musí obsahovat název procesní instance a vykonané aktivity v chronologickém pořadí. Časový údaj procesního logu poskytuje informaci o době trvání mezi jednotlivými aktivitami. Často je možné se setkat s informací o začátku a konci jedné aktivity. Atributy v procesním logu je možné rozdělit na dva typy – event level atributy a case level atributy. Event level atributy jsou variabilní a mohou se měnit (člověk vykonávající aktivitu, čas nebo cena). Case level atributy jsou pro celou instanci neměnné (místo, produkt nebo systém ve kterém procesy probíhají). [29]

Case id	Event id	Properties				
		Timestamp	Activity	Resource	Cost	...
1	35654423	30-12-2010:11.02	register request	Pete	50	...
	35654424	31-12-2010:10.06	examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	check ticket	Mike	100	...
	35654426	06-01-2011:11.18	decide	Sara	200	...
	35654427	07-01-2011:14.24	reject request	Pete	200	...
2	35654483	30-12-2010:11.32	register request	Mike	50	...
	35654485	30-12-2010:12.12	check ticket	Mike	100	...
	35654487	30-12-2010:14.16	examine casually	Pete	400	...
	35654488	05-01-2011:11.22	decide	Sara	200	...
	35654489	08-01-2011:12.05	pay compensation	Ellen	200	...

Obrázek 11: *Procesní log, převzato z [29]*

Tabulka na obrázku je seřazená podle tzv. Case id, kdy každé case id reprezentuje jednu instanci procesu a je neměnné. Procesní log dále disponuje unikátním identifikačním číslem každé události (Event id), časem, ve kterém daná událost proběhla, popisem aktivity, jménem osoby, kterou byla aktivita provedena a cenou související s danou událostí.

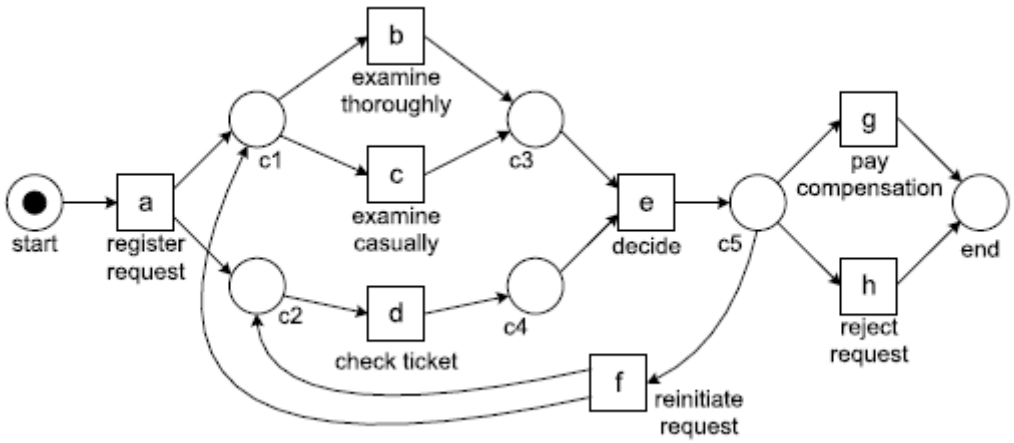
Pro účely sestavení modelu použitím process mining algoritmů je v tomto příkladu zapotřebí, aby case id a aktivita byly mezi sebou navzájem propojené a seřazené dle Case id. Díky tomu pak může dojít k sestavení tabulky z obrázku 12 níže, která obsahuje posloupnost aktivit provedených v souvislosti s jednotlivými Case id. Aktivity v tabulce jsou reprezentovány pomocí písmen (a = register request, b = examine thoroughly, c = examine, d = check ticket atd.)

Pomocí process mining algoritmů pak můžou být informace z tabulky na obrázku 12 transformovány do procesního modelu. [29]

Case id	Trace
1	$\langle a, b, d, e, h \rangle$
2	$\langle a, d, c, e, g \rangle$
3	$\langle a, c, d, e, f, b, d, e, g \rangle$
4	$\langle a, d, b, e, h \rangle$
5	$\langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle$
6	$\langle a, c, d, e, g \rangle$
...	...

Obrázek 12: *Posloupnost aktivit procesu, převzato z [29]*

Model vyobrazen na obrázku 13 je vytvořen pomocí tzv. α-algoritmu, který umožňuje z poskytnutých dat z obrázku 12 pomocí Petriho sítě vytvořit procesní model. Jeho správnost je možné ověřit provedením kontroly kroků v případě case id č. 1. Díky kroku *a* jsou otevřená místa *c1* a *c2* a je možné pokračovat k *b* a *d*. Po provedení událostí (*a, b*) pak zůstává provedení *d*, kdy je dále možné po krocích *c3* a *c4* pokračovat aktivitám (*e, h*).

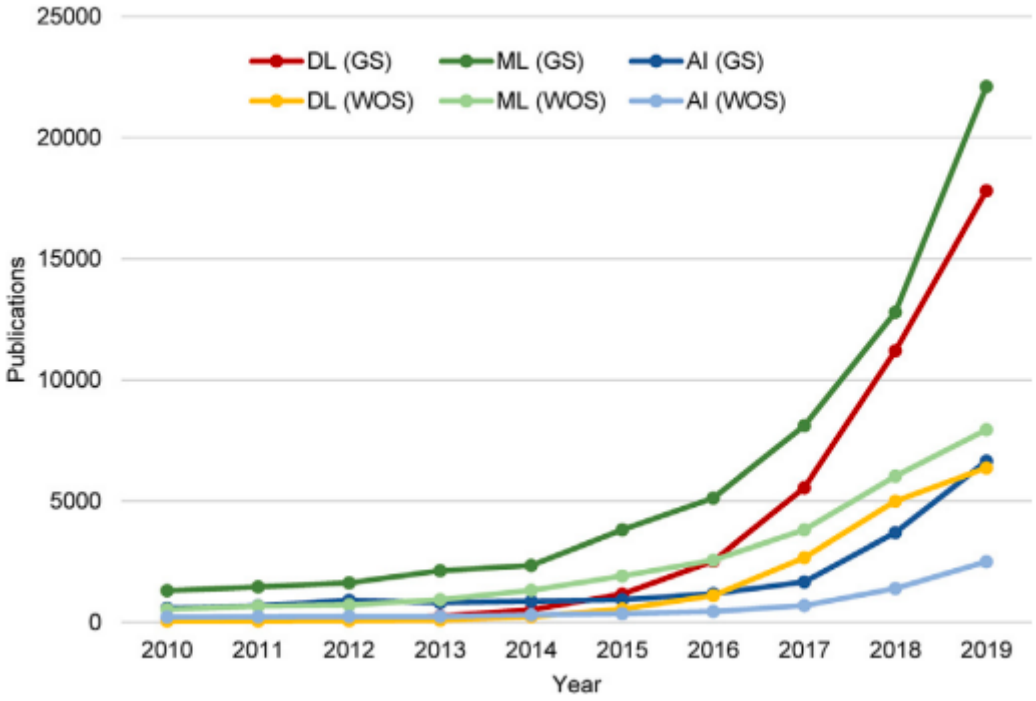


Obrázek 13: *Model process miningu, převzato z [29]*

Obrázek 13 poskytuje příklad jednoduchého modelu procesu vyřízení požadavku na reklamaci vstupenky, který byl vytvořen pomocí tzv. α-algoritmu. V praktické části je představen reálný příklad modelů, které popisují reálné procesy, odehrávající se v autonomním centrálním skladu.

4 Machine Learning

Cílem Machine leraningu, dále také pod zkratkou ML, je naprogramovat počítač tak, aby byl schopen se učit z poskytnutých dat. Za jednu z prvních definic ML je považován výrok Arthura Samuela z roku 1959 „Machine learning je obor, který dává počítači schopnost se učit, aniž by k tomu byl explicitně naprogramován“. V dnešní době se s aplikacemi ML setkáváme dnes a denně, ať už se jedná o detektor spamů v e-mail klientovi, hlasového asistenta, napovídání při vyhledávání nebo samořídící automobily.



Obrázek 14: *Vědecké publikace související s Machine Learning, převzato z [30]*

Na obrázku 14 je zobrazen graf, který ukazuje rapidní růst počtu vědeckých publikací na serverech Google Scholar a Web of Science za období 2010 až 2019. Jedná se o publikace na témata jako jsou Deep Learning (DL), Machine Learning (ML) a Artificial Intelligence (AI).

Využití ML metod je možné v širokém spektru odvětví a oborů. Jeho nespornou výhodou je jednoduchost, s jakou je možné řešit komplexní problémy. Příkladem může být již zmíněný detektor spamů v elektronické poště. V případě, kdy by probíhalo programování takového software bez použití ML, jednalo by se s nejvyšší pravděpodobností o kód, který by v sobě zahrnoval příklady často se vyskytujících slov či souvětí, které se používají v nevyžádaných e-mailech. Hned jak by si odesílatelé takových zpráv uvědomili, že jsou jejich e-maily

přesměrované do nevyžádané pošty, změnili by způsob komunikace a bylo by zapotřebí aktualizovat již napsaný program, který by v sobě zahrnoval nové výrazy. Tento proces by tak musel probíhat neustále dokola a nikdy by neskončil.

Díky ML je možné rozpoznat změnu ve vzorcích chování odesílatelů nechtěných zpráv a tím přeorientovat chování detektoru na nový typ e-mailů bez jakéhokoliv zásahu ze strany vývojářů. V případě ML totiž dochází k neustálému učení se ze zpráv obdržených v minulosti, které dovolí modelu detekovat opakující se vzorec v nevyžádané poště. Následně je možné využít ML inspekci nového trendu v nevyžádaných zprávách tak, že po nějakém čase, kdy modelem prošlo větší množství dat, dojde k analýze zpráv, které model považoval za spam a zjistit tak spolehlivost modelu, nebo odhalit nové vzorce chování odesílatelů.

Obecně je možné říct, že je Machine learning vhodné aplikovat v následujících případech:

- Řešení problému vyžaduje mnoho úprav v průběhu času či velkého množství pravidel
- Složitost problému neumožňuje nalézt tradiční řešení
- Prostředí, ve kterém je potřeba řešit problém je velmi proměnlivé
- Řešení vyžaduje získávání důležitých informací z velkého množství dat [31]

4.1 Typy Machine Learning systémů

Momentálně existuje velké množství typů ML systémů, a proto je vhodné tyto typy rozdělit do jednotlivých, obecnějších kategorií. Takové kategorie ML algoritmů/modelů mohou být vytvořeny na základě následujících kritérií/úvah/otázek:

- Je model trénován s pomocí učitele či nikoliv? (učení s učitelem, učení bez učitele, kombinace učení s i bez učitele, nebo zpětnovazební učení)
- Dokáže se model učit postupně za běhu? (online versus batch learning)
- Algoritmus se učí pouze pomocí porovnávání nových dat s daty předchozími, nebo je schopen odhalit vzorce v trénovacích datech, vytvořit model a předvídat další vývoj? (instance-based learning versus model-based learning)

Je možné si klást jiné otázky, nebo je mezi sebou různě kombinovat. Nejedná se tedy o rozdělení, které je nutné za každou cenu sledovat. Nicméně je tato kategorizace jasná a zahrnuje v sobě podstatu rozdělení typů algoritmů strojového učení. V následujících podkapitolách budou proto zmíněná kritéria podrobněji rozebrána. [31]

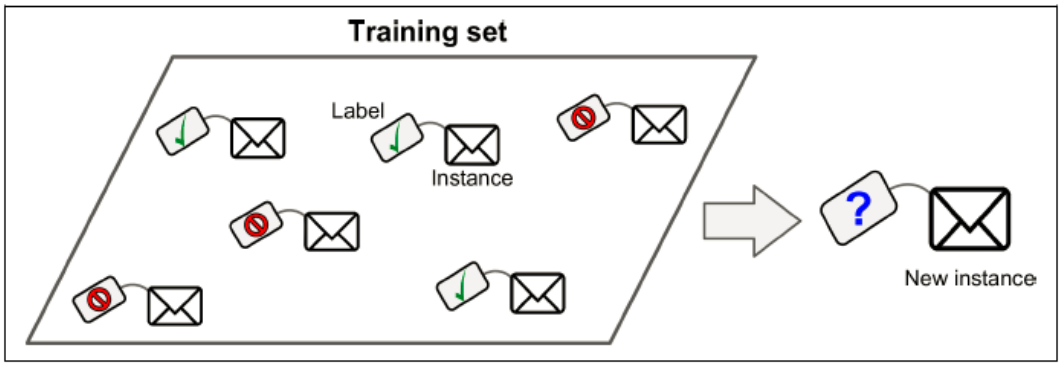
4.1.1 Učení s učitelem

Při učení s učitelem se k učení modelu využívá množiny tzv. označených vzorů (labeled dataset), jehož součástí je i očekávané řešení. Datový soubor se dále dělí na:

- Trénovací data – tato část dat je použita v procesu učení modelu
- Validační data – se využívá pro kontrolu průběhu učení. V případech, kdy dochází k růstu přesnosti predikce u trénovací množiny dat, a naopak poklesu přesnosti predikce u validačního souboru, jedná se o tzv. přeučení modelu. Model si pamatuje pouze trénovací data a nic se nenaučil.
- Testovací data – slouží ke kontrole přesnosti modelu po jeho naučení. Testovací data musí být odlišná od trénovacích, aby bylo možné správně hodnotit průběh učení modelu a jeho přesnost.

Proces učení modelu začíná vytvořením předpovědi na základě vstupů a ta je následně porovnána s předpokládaným výsledkem. Model je pak upravován za účelem snížení chyby predikce. Tento proces se opakuje, dokud se hodnota celkové chyby nesníží na požadovanou úroveň.

Učení s učitelem řeší dva typy příkladů. Prvním příkladem učení s učitelem je klasifikace. Již zmiňovaný spamový filtr patří právě do této kategorie. Model je trénován pomocí souboru, který obsahuje příklady spam e-mailů a tzv. ham e-mailů. Po procesu učení by měl být model schopen určit, zda se jedná o vyžádaný e-mail či nikoliv. [31]



Obrázek 15: *Trénování klasifikačního modelu, převzato z [31]*

Na obrázku 15 je vyobrazen trénovací soubor s vyžádanými i nevyžádanými zprávami na základě kterého je trénován klasifikační model. Následně probíhá zařazení příchozí pošty.

Druhým příkladem jsou úlohy předvídání cílové numerické hodnoty, například cenu automobilu (vysvětlovaná proměnná) v závislosti na proměnných vysvětlujících (ujeté kilometry, rok výroby, značka). Pro natrénování takového modelu je zapotřebí dát algoritmu mnoho trénovacích dat s vlastnostmi auta i jeho cenou.

Obrázek 16: Regresní mode, převzato z [31]

Na obrázku 16 je příklad regresního modelu. Graf na ose y znázorňuje hodnoty vysvětlované proměnné, která je závislá na hodnotách nezávislých proměnných na ose x. Při nové vstupní hodnotě by měl model být schopen odhadnout hodnotu závislé proměnné.

Nejpoužívanější algoritmy při učení s učitelem

- Algoritmy K-Nearest neighbor
- Lineární regrese
- Logistická regrese
- Metoda podpůrných vektorů (SVMs)
- Algoritmy náhodného lesa a rozhodovací stromy

4.1.2 Učení bez učitele

Při učení bez učitele není v průběhu procesu učení využito množiny tzv. označených vzorů (labeled dataset). Trénovací data tak neobsahují očekávané řešení. Algoritmy tohoto typu učení označí data uspořádáním, nebo popisem jejich struktury. Technika učení bez učitele je vhodná v případech, kdy není zřejmé, jak by měl výsledek vypadat.

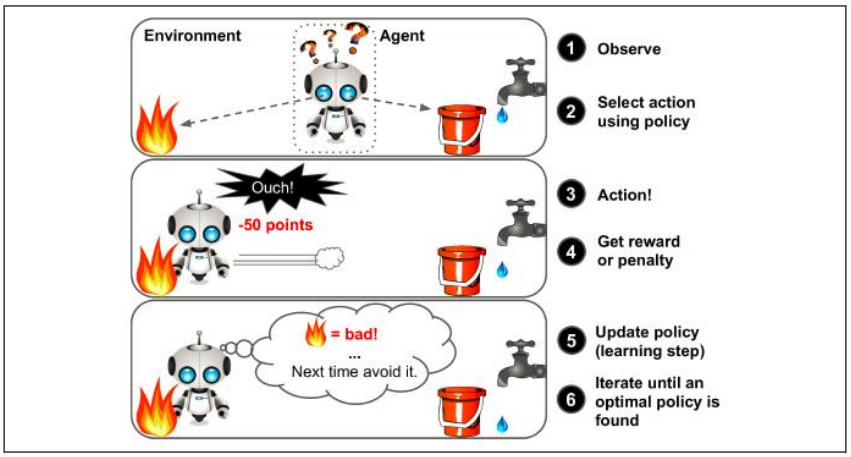
Klasická úloha pro tento typ strojového učení je segmentace zákazníků dle jejich nákupních preferencí. V tomto případě se předpokládá náležitost velkého množství dat o zákaznících a produktech, které nakoupili. Označení dat probíhá během procesu učení zjišťováním vztahů mezi jednotlivými prvky v datech. Například je možné z dat odhalit, že určité procento uživatelů je mužského pohlaví a nakupuje produkt A večer, kdežto další procento uživatelů je pohlaví ženského a nakupuje výrobek B ráno.

Nejpoužívanější algoritmy při učení bez učitele

- **Shlukování (Clustering)**
  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
- **Asociace (Association rule learning)**
  - Apriori
  - Eclat

4.1.3 Zpětnovazební učení

V případě zpětnovazebního učení se učící systém, tzv. agent přijímat rozhodnutí v prostředí na základě odměn, nebo pokut, které za každé rozhodnutí obdrží. Takový systém se musí naučit, která strategie je pro něj nejvýhodnější z hlediska odměny.



Obrázek 17: *Zpětnovazební učení, převzato z [31]*

Příkladem použití tohoto typu učení jsou roboti, kteří se učí chodit, nebo DeepMind's AlphaGo program, který v roce 2017 porazil mistra světa Ke Jie ve hře Go.

4.1.4 Online vs Offline učení

Další kritérium pro dělení typu systémů strojového učení je, zda se model dokáže učit z proudu příchozích dat či nikoliv.

V případě tzv. Batch Learningu není systém schopen se učit z přicházejícího toku dat a musí mu být poskytnuta veškerá data najednou. Takové učení vyžaduje větší množství času a výpočetní kapacity a probíhá offline. Model je po dokončení procesu učení nasazen do produkce a v případě jakýchkoli změn je třeba učení opakovat na novém souboru dat a následně opět nasadit.

Algoritmy systémy strojového učení mohou být trénované postupně z nových příchozích dat. Jedná se o tzv. online způsob procesu učení, které posílá data do systému nepřerušovaně, nebo v malých dávkách. Tento způsob učení je vhodný například při predikci finančních trhů, kdy je potřeba sledovat rychlý vývoj cen aktiv. Výhodou tohoto typu učení je úspora úložiště, protože data po procesu učení mohou být smazána. Naopak nevýhodou může být proměnlivá kvalita trénovacích dat, kdy dochází ke snížení přesnosti modelu a je zapotřebí následné přeučení.

4.1.5 Instance vs Model-Based Learning

Strojové učení je také možné dělit podle toho, zda učení probíhá pouze pomocí instancí, nebo je pro potřeby učení vytvořen konkrétní model. Učení podle instancí je nejjednodušším přístupem. V případě učení detektoru spamu by bylo využito příkladů spam e-mailů a v případě podobnosti mezi trénovacími daty a novými příchozími zprávami by e-mail byl zařazen do nevyžádané pošty. Takový model by však mohl zařadit do nevyžádané pošty i e-mail, který by obsahoval pouze nějakou frázi podobnou e-mailům, které jsou definované jako spam.

Mnohem sofistikovanější způsob učení je pomocí vytvoření modelu. Například v případě, kdy je potřeba zjistit, zda peníze dělají lidi šťastnější, je možné získat data o HDP a indexu životní úrovně a datové soubory spojit. Následně probíhá hledání trendu. V případě lineárního trendu je vytvořen model lineární regrese s definicí optimálních parametrů modelu dle nejnižší chyby, který je pak použit pro trénování. [31]

4.2 Nástroje pro programování ML

Nejpoužívanějšími programovacími jazyky pro implementaci machine learning technik jsou Python, Java a C/C++. Tyto programovací jazyky disponují řadou knihoven, které jsou navrženy pro práci s daty, jejich transformaci, vizualizaci a implementaci machine learning algoritmů. Mezi další populární jazyky využívané při strojovém učení se řadí Julia či LISP. [32] Jelikož bylo v této práci využito programovacího jazyku Python, bude mu věnována pozornost v následujících kapitolách i praktické části.

4.2.1 Python

Programovací jazyk Python je mezi vývojáři velmi populární a je mnohými považován za jeden z nejlepších v souvislosti s práci s daty a strojovým učením. Python je dynamicky sémantický, interpretovaný, objektově orientovaný programovací jazyk vysoké úrovně. Jeho vysokoúrovňové vestavěné datové struktury spolu s dynamickým typováním a dynamickými vazbami z něj dělají ideální jazyk pro psaní skriptů a rychlý vývoj aplikací. Stručná a snadno naučitelná syntaxe jazyka Python upřednostňuje čitelnost, což snižuje náklady na údržbu softwaru. Jazyk Python disponuje řadou modulů a balíčků, což podporuje modularitu programů a opakované použití kódu. Interpret jazyka Python a jeho rozsáhlá standardní knihovna jsou zdarma ke stažení a distribuci ve zdrojové nebo binární podobě pro všechny hlavní platformy. [33]

Nejpoužívanější knihovny

- **Scikit-Learn –** Open source knihovna, která se hodí pro implementaci algoritmů pro učení s učitelem i bez učitele. Poskytuje také mnoho nástrojů pro práci s daty, modely, úpravou dat apod. Je velmi efektivní a srozumitelná. [34]
- **Keras –** Je open source knihovna vhodná pro vývoj umělých neuronových sítí. Součástí je také vysokoúrovňové rozhraní usnadňující zacházení s knihovnou. Využívá se často ve spolupráci s dalšími knihovnami jako např. TensorFlow. [35]
- **TensorFlow –** Velmi komplexní knihovna pro distribuované numerické operace. Dovoluje vytvářen a implementovat velmi rozsáhlé neuronové sítě s distribuovaným výpočetním výkonem až na několika desítkách tzv. multi-GPU serverů. TensorFlow byl vytvořen společností Google a podporuje mnoho jejich machine learning aplikací. Nyní se jedná o open source projekt, na jehož vývoji se podílí řada vývojářů. [36]

4.3 Statistické modely používané v úlohách Machine Learning

Modelů pro využití strojového učení existuje celá řada. Několik z nich bylo již zmíněno výše a v této kapitole bude pozornost věnována několika nejznámějším modelům. Tato práce je zaměřená na tzv. time-series data neboli data v podobě časových řad, a proto jsou v této kapitole modely pro analýzu časových řad rozebrány podrobněji.

4.3.1 Lineární a logistická regrese

Jedná se o jedny z nejpoužívanějších úloh. Algoritmy pro řešení těchto úloh spadají do typu učení s učitelem. V případě učení s učitelem jsou řešeny dva druhy problémů – klasifikační a regresní. Při řešení klasifikačního problému (klasifikace s učícím souborem – kdy skupiny jsou předem známé, tzv. úloha diskriminace) dochází k zařazení hodnot v datovém souboru do skupin, které vznikají na základě určité podobnosti. Následně probíhá predikce třídy pro nové vstupní hodnoty. U regresního problému jde o identifikaci dat spojitých hodnot o předpovědi této veličiny. Na rozdíl od klasifikace se tedy nepredikuje třída, ale konkrétní hodnota. [3]

Lineární regrese

Lineární regrese předpokládá lineární závislost mezi dvěma veličinami závislé proměnné (vysvětlované) a nezávislé proměnné (vysvětlující). Prvním krokem bývá zpravidla nanesení dat do bodového grafu a ověření přítomnosti předpokládané závislosti. Následuje provedení odhadu regresní přímky, jejíž rovnice je ve tvaru

$$Y_i = \beta_0 + \beta_1 \times x_i + \varepsilon_i$$

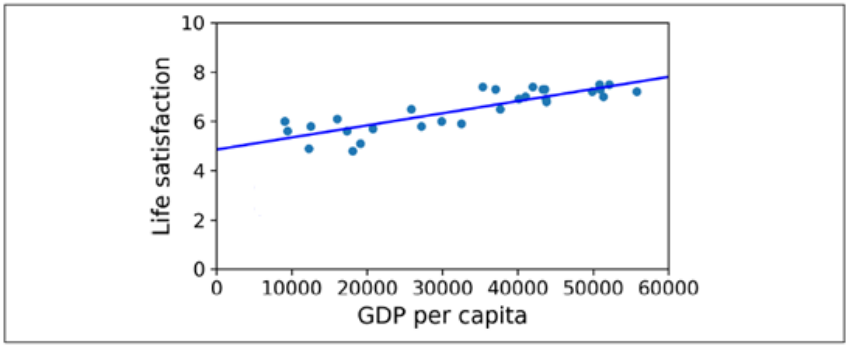
Rovnice popisuje lineární statistickou závislost mezi veličinou X (nezávislou) a závislou veličinou Y neboli přímku + chybu, kdy nenáhodná část modelu je přímka a ε_i je náhodná odchylka.

- Absolutní člen β_0, geometricky průsečík přímky se svislou osou.
- Regresní koeficient, koeficient v regresi, geometricky směrnice β_1 přímky.

Cílem regrese je proložit naměřenými body přímku, která nejlépe vystihuje vztah mezi proměnnými. K proložení přímky se často využívá metody nejmenších čtverců. Lineární

regrese se využívá například pro predikce vývoje cen, hospodářských ukazatelů, množství prodaného zboží a obecně tehdy, kdy mezi oběma veličinami platí lineární vztah.

Na obrázku níže, je ukázka modelu lineární regrese pro predikci životní úrovně státu, dle HDP per capita. Jednotlivé body na grafu představují HDP jednotlivé země a úroveň životní spokojenosti dle dat poskytovaných OECD. Po natrénování modelu, je možné predikovat životní spokojenost občanů určité země, podle velikosti HDP per capita. [31]



Obrázek 18: *Model lineární regrese, převzato z [31]*

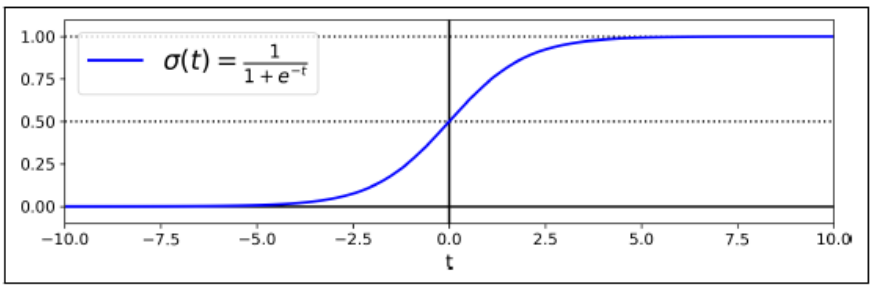
Logistická regrese

Logistická regrese se využívá k řešení klasifikačních problémů, kdy vysvětlovaná proměnná není spojitá. Cílem logistické regrese je predikovat pravděpodobnost výskytu určitého jevu při změně hodnot vysvětlujících (nezávislých proměnných). Typickým příkladem je predikce pravděpodobnosti výskytu spam e-mailu (hodnoty v rozsahu 0-1). Logistická regrese převádí regresi lineární na klasifikaci pomocí logistické (sigmoidální) funkce, která má tvar

$$\sigma(t) = \frac{\exp(t)}{\exp(t) + 1} = \frac{1}{1 + \exp(-t)}$$

Za předpokladu, že t je lineární funkcí vysvětlující proměnné x, je možné vyjádřit t jako

$$t = \beta_0 + \beta_1 x$$



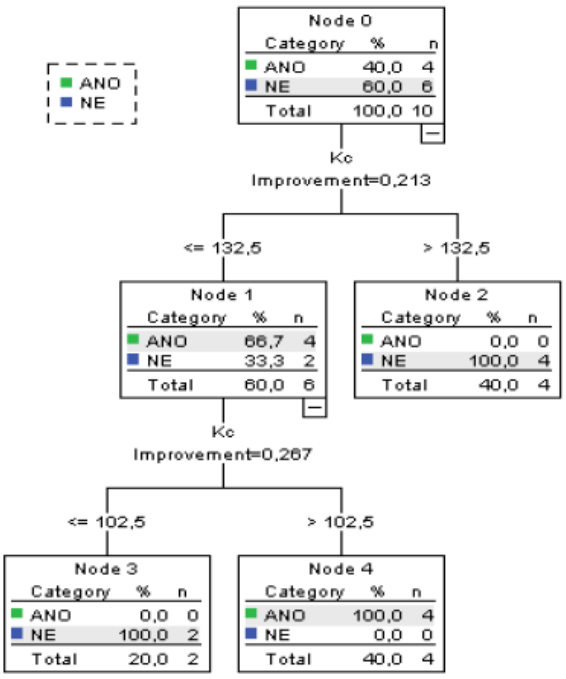
Obrázek 19: *Funkce logistické regrese, převzato z [31]*

4.3.2 Rozhodovací stromy

Rozhodovací stromy se využívají pro řešení klasifikačních úloh s učitelem. Princip modelu rozhodovacího stromu je rozdělení souboru do předem známých tříd, které zajistí tzv. minimální nečistotu neboli homogenitu tříd. Spočívá ve výběru atributů, které umožní optimální klasifikaci a sestavení rozhodovacích, klasifikačních pravidel – hierarchického rozhodovacího stromu. Růst stromu probíhá na základě rekurzivního binárního dělení.

Pro zajištění atributu, který zajistí nejlepší rozdělení uzlů se využívá tzv. kriteriální statistiky (spliting criterium). Kritérium optimalizuje dělení dat tak, aby v koncových uzlech byla docílena co nejvyšší homogenita. Nejčastějšími mírami homogenity v případě rozhodovacích stromů je kritérium minima kvadratické chyby – Gini index a Entropie.

Následující model na obrázku 20 popisuje pravděpodobnost zakoupení výrobků stejného typu na základě ceny. Vstupní datový soubor obsahuje určitý počet výrobků, jejich cenu a informaci o zakoupení (Ano/Ne). Algoritmus si nejdříve seřadí data podle ceny výrobků, vytvoří dělící body podle střední hodnoty mezi atributy, vypočítá Gini index a podle jeho nejnižší hodnoty určí maximální heterogenitu neboli hodnotu rozdělení při prvním dělení (cena 132,5 Kč). Závěrem takového triviálního modelu může být doporučení pro nastavení ceny produktu mezi 102,5 Kč a 132,5 Kč. [3]



Obrázek 20: *Rozhodovací strom, převzato z [3]*

4.3.3 ARIMA

ARIMA je model využívaný pro analýzu a predikci vývoje časových řad. Jedná se o nejobecnější model tohoto typu, který je založen na Box-Jenkinsové metodologii. Model ARIMA je v této práci použit, a proto bude v této kapitole rozebrán a vysvětlen důkladněji.

Box-Jenkinsova metodologie

Jedná se o metody dekompozice časové řady, jež na rozdíl od klasických dekompozičních metod, které kladou důraz na systémové složky a s jednotlivými pozorováními zachází jako s navzájem nekorelovanými, uvažuje náhodnou složku časové řady jako základní prvek, který může být tvořen korelovanými náhodnými veličinami. Korelační analýza v Box-Jenkinsonově zkoumá závislosti mezi veličinami časové řady.

Obecně je možné zapsat jakýkoliv lineární model v následujícím tvaru:

$$y = G \cdot u + H \cdot e$$

kde jsou y výstupní veličinou a u vstupní veličinou, G transformační funkcí (vyjadřuje dynamické vlastnosti systému – závislost výstupu na vstupu), e reprezentuje chybovou složku – vyjadřuje skutečnost, že se stejným vstupem dostaneme jiný výstup a H je tzv. „noise model“ (popisuje, jak působí rušení na výstup ze standardizovaného zdroje rušení e(t)).

Při implementaci metody hrají důležitou roli tzv. autokorelační vlastnosti časových řad. Stacionarita časové řady znamená, že chování časové řady je stochasticky ustálené. Za stacionární proces je považován proces rovnoměrně vyvážený (konstantní rozptyl), kdy závislost mezi dvěma libovolnými pozorováními závisí pouze na jejich vzájemné časové vzdálenosti, a nikoliv na jejich umístění v časové řadě.

Autokovarianční a autokorelační funkce odhaduje sílu závislosti mezi veličinami. Průběh a vlastnosti autokorelační funkce jsou v rámci Boxovy-Jenkinsovy metodologie důležitým ukazatelem, neboť napovídají, jaký typ modelu je vhodné pro danou časovou řadu použít. [37] [38]

Autoregresní model AR

Patří spolu s modelem klouzavých průměrů mezi hlavní představitele Box-Jankinsonovy metodologie. Autoregresní model AR(p) vychází ze vztahu závislosti hodnot a tudíž předpokládá, že každá hodnota v časové řadě, je v závislosti s předchozími hodnotami v dané řadě. Model je definován následujícím vztahem:

$$Y_t = \alpha + \rho Y_{t-1} + \varepsilon_t$$

kde nová hodnota řady Y_t je vypočítána na základě předchozích hodnot α násobenou autokorelační vlastností řady (autoregresním parametrem) a současné hodnoty bílého šumu ε_t s nulovou střední hodnotou a rozptylem σ^2.

Y je stacionární, pokud |ρ| < 1 a nestacionární pro ρ = 1 (jednotkový kořen) a |ρ| > 1 (nestacionární explozivní vývoj). [39]

Model klouzavých průměrů MA

Tento model popisuje časovou řadu jako lineární kombinaci minulých hodnot bílého šumu ε_t. Proces klouzavých průměrů MA(q) je možné zapsat následujícím vztahem:

$$Y_t = \theta_0 + \varepsilon_t - \theta_1 \varepsilon_{t-1}$$

$$\mu = \theta_0$$

Proces MA(q) je stacionární pro libovolnou volbu jeho parametru a jeho střední hodnota je nulová. U modelů klouzavých průměrů je kladen důraz na podmínku „invertibility“. Proces AR(p) lze vždy napsat jako proces MA(∞). Zatímco proces MA(q) je invertibilní, všechny kořeny rovnice θ(L) = 0 musí ležet mimo jednotkovou kružnici. Tato podmínka je známá jako podmínka „invertibility“ pro proces MA(q). [38] [40]

Smíšený model ARMA

Kombinací procesu AR(p) a MA(q) dostaneme smíšený proces ARMA(p,q). Smíšený proces ARMA je často zapisován pomocí operátoru zpětného posunutí:

$b(B) \cdot y = \varepsilon_t$ pro AR(p) s autoregresním parametrem b(B)

$y_t = w(B)\varepsilon_t$ pro MA(q) s parametry modelu w(B)

$b(B)y_t = w(B)\varepsilon_t$ pro ARMA kombinaci

Podmínka stacionarity procesu ARMA je shodná s podmínkou stacionarity procesu AR(p) a podmínka invertibility procesu je shodná s podmínkou invertibility procesu MA(q). Střední hodnota procesu ARMA je nulová a jeho autokorelační funkce vyhovuje podobné soustavě diferenčních rovnic jako v případě AR procesu.

Model ARIMA(p,d,q) je nestacionární smíšený model který vznikl integrací dvou, výše popsaných, modelů AR a MA. ARIMA modely umožnují popis procesu, u kterých nejenže dochází ke změnám úrovně, ale tyto změny mohou mít nesystematický, náhodný charakter (což je pro časové řady z praxe běžné). Tento model modeluje stochasticky vedle náhodných fluktuací i trendovou složku. Modely ARMA se konstruují až pro řadu prvních nebo vyšších diferencí časové řady a při konstrukci ARIMA modelu není požadována stacionarita analyzované řady.

ARIMA model je definován následujícím vztahem:

$$b(B) \cdot v_t = w(B) \cdot \varepsilon_t$$

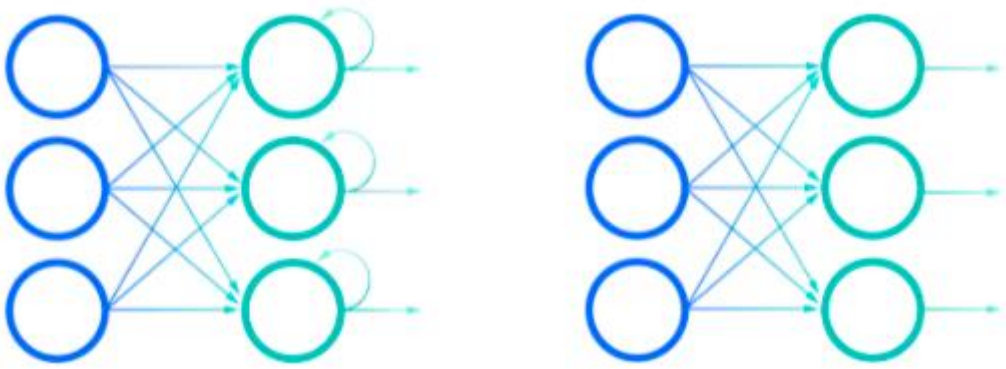
$$v_t = \Delta^d y_t$$

v_t je *d*-tá diference modelovaného procesu *y* pro proces *v*, kde proces *v* je proces ARMA(p,q) s nenulovou střední hodnotou. [41]

4.3.4 LSTM

Long Short Term Memory (LSTM), je speciální druh rekurentních neuronových sítí a svoji popularitu získává zejména díky schopnosti řešení problému mizejícího gradientu v případě použití metod zpětné propagace chyby. Struktura LSTM v sobě zahrnuje prvky, které umožňují dlouhodobé uchování informací.

Rekurentní neuronové sítě (RNN), jsou typem umělých neuronových sítí, které ke svému učení využívají zejména sekvenční data v podobě časových řad. K učení vyžadují trénovací sadu dat a od klasických neuronových sítí se liší schopností pamatovat si předchozí vstupy, pomocí kterých mohou být ovlivněny budoucí vstupy i výstupy. RNN vytváří tzv. vnitřní cykly a tvoří tak efektivní model pro rozpoznávání určitých vzorců chování v dlouhé sekvenci dat.



Obrázek 21: *Rekurentní neuronové sítě vs umělé neuronové sítě, převzato z [42]*

Na obrázku 21 je vyobrazen rozdíl mezi klasickými umělými neuronovými sítěmi (vlevo) a rekurentními (vpravo).

Díky schopnosti hledání možných korelací mezi hodnotami a sdílení parametrů napříč každou vrstvou RNN jsou modely tohoto typu hojně používány při řešení problémů jako jsou např. systémy rozpoznávání řeči, autonomní řízení na dálnicích, rozpoznávání obrázků a využívají je software jako jsou Siri nebo Google Translate. [42]

5 Praktická část

Tato část diplomové práce je zaměřená na aplikaci metod a postupů představených v teoretické části na strojová data z autonomního centrálního skladu ŠKODA AUTO, a. s. Data jsou transformována pomocí programovacího jazyku Python do podoby, která usnadní jejich následnou analýzu a vizualizaci v Microsoft Power BI. Následuje implementace process miningu za účelem pochopení procesů probíhajících ve skladu. Závěr této části je věnován využití machine learning modelu ARIMA s cílem predikce výskytu chyb v probíhajících procesech.

5.1 Extrakce a transformace dat

Logy ze strojů obsluhujících sklad se v reálném čase ukládají do Splunku. Pro potřeby analýzy, process miningu a machine learningu bylo rozhodnuto, že data ze Splunku v seemi– strukturované podobě budou exportována a transformována dle potřeby. Nejvýhodnějším formátem pro reprezentaci dat je textový formát .csv, ve kterém byla data exportována.

Zpracování .csv souborů s daty probíhalo prostřednictvím programovacího jazyku Python a bylo rozděleno na dvě části. První část python kódu měla na starost parsing logu a transformaci dat do čitelné podoby, pro usnadnění další práce s daty. Druhá část se zabývala vytvořením rozdílných metrik, které zjednodušily následnou vizualizaci dat.

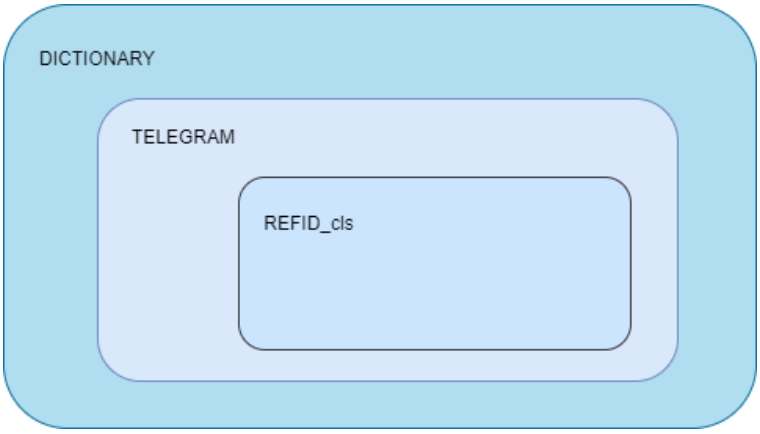
V první části byl za pomocí python knihovny Pandas proveden parsing logové zprávy, díky kterému je možné z logu získat potřebné informace. Pandas je knihovna, která poskytuje veškeré nástroje potřebné pro nakládání a manipulaci s daty. Jednou z nejdůležitějších vlastností, je možnost vytvoření tzv. dataframe objektu pro rychlou a efektivní práci s daty. Dataframe je možné si představit jako tabulku, která se skládá z tzv. Pandas Series, což je jednodimenzionální datová struktura v podobě seznamu prvků. Tyto datové struktury poskytují lepší manipulaci a orientaci v datech. Pro extrakci potřebných částí z logové zprávy bylo využito regulárních výrazů neboli regexu a dalších nástrojů, které Pandas nabízí, jako například možnost podmíněného vyhledávání a filtrace v datech, aplikace datetime formátu pro práci s časovými údaji, odstranění nepotřebných řádků v datovém souboru apod.

Po rozboru všech řádků .csv souboru pomocí Python knihovny Pandas byly získány z logových zpráv požadované údaje a uloženy do nového, výstupního .csv souboru v následujícím formátu:

time	REFID	type	zdroj	destinace	cilovadest	kodchyby
2021-10-04 00:20:41	3121436523350	LAM	B31304600512	LAM01.P1****	B1-R401*****	6401
2021-10-04 00:21:16	3121431518356	LAM	LAM01.P1****	B4AP07110744	B1-R401*****	6401
2021-10-04 00:22:45	3121436318350	1	B2SC01116248	B2SC02116252	B1-R401*****	6401
2021-10-04 00:22:50	3122236518312	MPR	WA_PRN01	nan	nan	nan
2021-10-04 00:22:50	3125536518350	1	B2SC02116252	B2SC02216253	B1-R401*****	6401
2021-10-04 00:22:55	3121436523360	1	B2SC02216253	B2SC05216258	B1-R401*****	0
2021-10-04 00:22:58	3129536518320	1	B2SC05216258	B1SC06215431	B1VP26414540	0
2021-10-04 00:23:46	3112436518350	1	B1SC06215431	B1SC07215531	B1VP26414540	0
2021-10-04 00:24:00	3121436518331	1	B1SC07215531	B1VP26414540	B1VP26414540	0
2021-10-04 00:24:08	3121436518234	10	nan	nan	nan	nan
2021-10-04 00:34:18	3121436511122	1	B1SC26424547	R40200100300	R40200100300	0
2021-10-04 00:49:09	3121436514455	11	nan	nan	nan	nan

***Tabulka 2**: Získané hodnoty z semi– estrukturované logové zprávy*

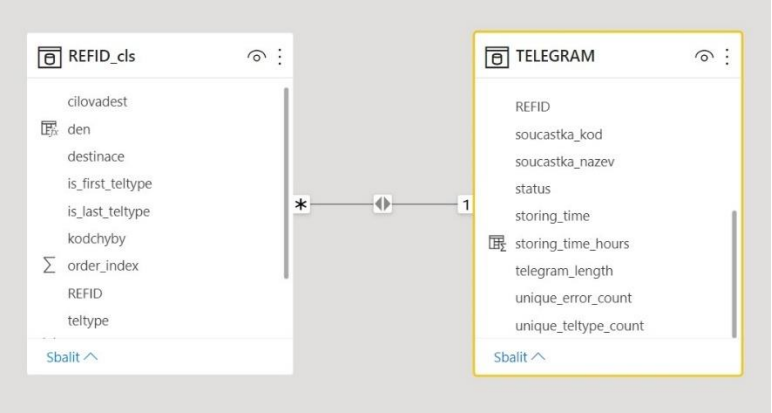
Druhá část zpracování dat byla zaměřena na vytvoření lepší struktury datového souboru. První krok této fáze vede k uspořádání rozparsovaných dat do tzv. dictionary struktury. Dictionaries v Pythonu jsou pole formátu key-value. Dictionary byla naplněná prostřednictvím for cyklu, který procházel .csv soubor a dle definovaných podmínek ukládal data v potřebném tvaru. Dictionary bylo využito k ukládání jednotlivých telegramů podle referenčního ID, které vystupuje jako klíč a hodnotami jsou objekty veškerých telegramů, jež mají stejné referenční ID, dále pod pojmem REFID. Schéma takového ukládání je zobrazeno na následujícím obrázku:



***Obrázek 22:** Schéma struktury transformace dat [vlastní zpracování]*

Do dictionary s názvem REFID_dictionary jsou ukládány instance třídy TELEGRAM. Atributy této třídy byly v průběhu procesu zpracování dat dopočítávány (duration, status, str_time, cont_loop, completed apod.). Nutno zmínit, že mezi atributy objektu TELEGRAM patří také

list telegrams[], do kterého byly ukládány instance třídy REFID_cls se stejným REFID. Atributy třídy REFID_cls byly tvořeny údaji získánými z parsingu hrubé logové zprávy. Mezi atributy patří: čas, teltype, REFID, kód chyby, destinace, cílová destinace, zdroj, viz. tabulka 2.



Obrázek 23: *Relace mezi třídami REFID_cls a TELEGRAM [vlastní zpracování]*

Na obrázku 23 je zobrazen vztah 1: * mezi třídami TELEGRAM a REFID_cls, který je iniciován pomocí společného REFID.

Před ukládáním dat do dictionary bylo zapotřebí převést časový údaj v sloupci time na datový typ datetime, aby bylo možné v další části zpracování dat pracovat s časem a vypočítávat dobu trvání jednotlivých pohybů strojů. V rámci plnění dictionary tak docházelo k seřazení zpráv se stejným REFID dle času a zároveň k jejich indexaci.

Aby bylo možné určit správnost pohybů strojů ve skladu, bylo zapotřebí definovat a označit kompletní telegram neboli KLT přepravku, která prošla nezbytnými procesy. K označení kompletního telegramu (completed) bylo třeba nalézt první telegram dle času a odpovídajícího typu (type 0017) a posledního telegramu v listu s typem (type 0011). Další informace, která je z pohledu analýzy zajímavá, je doba trvání této cesty. Tu určíme rozdílem časů mezi telegramy typu 0011 a 0017. Takto byl vypočten atribut duration. Počty výskytů chyb a jednotlivých typů telegramů ve skladovacím procesu byly ukládány do jednotlivých atributů.

Zajímavou informací je mimo jiné čas, ve kterém se podaří KLT přepravku uložit na místo ve skladu, v našem případě byl tento údaj označen termínem „str_time“ neboli storing time. Storing time se opět vypočítával na základě času mezi jednotlivými místy ve skladu. Tentokrát bylo třeba najít první zmínku o přepravce, pro získání času vstupu do skladovacího procesu, a telegram s totožným REFID typu „LAM“. Čas uložení, „str_time“ přepravky, byl následně vypočten rozdílem časů prvního telegramu a telegramu typu LAM.

Součástí skriptu bylo i sledování jednotlivých destinací, kterými přepravky se součástkami procházely. Díky tomu bylo možné na první pohled vidět, kolika pozicemi daná KLT přepravka prošla a zda nedošlo k opakování jednotlivé pozice napříč putováním skladem. Na základě této myšlenky bylyv průběhu transformace dat vytvořeny dva nové atributy: „len“ a „cont_loop“.

5.1.1 Reprezentace transformovaných dat

Výstupem transformace dat za použití programovacího jazyku Python byly dva textové soubory formátu .csv:

REFIDtel_export.csv

Tento soubor reprezentuje každý zaznamenaný pohyb přepravky. Záznamy jsou seřazeny dle času, ze kterého vychází order a referenčního ID. Dále je z tohoto souboru možné získat informaci o zdroji, ze kterého je pohyb realizován, následující destinaci a konečnou destinaci, či záznam o chybě, vzniklé mezi pozicemi.

order	time	REFID	type	zdroj	Destinace	cilovadest	chyba
1	2021-11-01 23:26:23	03121438333669	17	nan	Nan	nan	nan
2	2021-11-24 21:07:32	03121438333669	LAM	B3040540	LAM01.P1**	B2R301****	6401
3	2021-11-24 21:07:54	03121438333669	LAM	LAM01.P	B4AP021102	B2R301****	6401
4	2021-11-24 21:15:04	03121438333669	11	nan	Nan	nan	nan

Tabulka 3: *Soubor REFIDtel_export.csv*

REFID_export.csv

Jak již z názvu sloupců vyplývá, jedná se o soubor obsahující získané informace z transformovaných dat, které se vztahují k jedinečnému referenčnímu ID (REFID). Z tohoto souboru je možné vyčíst, jaká součástka vešla do skladu, jak dlouho se tam nacházela, v jakém čase byla uložená na konkrétní pozici ve skladu a následně vyskladněná, nebo například kolik má unikátních chyb. Tato entita tedy reprezentuje jednu konkrétní KLT přepravku.

REFID	completed	cont_loop	telegram_len	status	len	kod	nazev	str_time	duration
03121435189786	TRUE	TRUE	LongTel	stored	9	None	None	0,368	117,984
03121434066214	TRUE	TRUE	ShortTel	completed	9	None	None	0,425	117,980
03121436713561	TRUE	FALSE	ShortTel	stored	9	None	None	0,682	117,882
03121436294235	TRUE	FALSE	ShortTel	completed	9	None	None	0,664	117,688

Tabulka 4: *Soubor REFID_export.csv*

Finální výstup Python algoritmu obsahuje informace jako jsou: referenční id přepravky s komponenty, výše vysvětlené atributy completed a cont_loop. Dalším údajem je vzdálenost, kterou přepravka ve skladu urazila. Tento údaj byl získán pomocí výpočtu mediánu kroků, kterými KLT přepravka prošla. KLT, které za svůj pobyt ve skladu udělaly více kroků, než byl zjištěný medián byly označeny termínem LongTel. Atribut status rozlišuje pouze uložené přepravky, tzn. že nemáme údaj o jejich vyskladnění, které může proběhnout až v dalším měsíci, nebo kompletní, kdy víme, že dané komponenty sklad již opustily. Zbytek atributů ve výsledném .csv souboru byl vysvětlen v předchozí kapitole. Příklad z tabulky 3 obsahuje telegramy, které nedisponují údajem o kódu, nebo názvu součástky. Tyto údaje má pouze konkrétní typ události (type 0017).

Jelikož bylo v průběhu procesu analýzy dat zjištěno, že strojové logy typu 0017 obsahují ještě informaci o dodavateli, čísle dodávky, datu dodání a kódu obalu, byl Python algoritmus pro parsing událostí upraven tak, aby z tohoto typu zpráv dokázal zmíněné informace získat. Finální soubor REFID_export.csv tedy obsahoval navíc ještě 4 sloupce (VendorNumber, DeliveryNumber, InboundDate, Cover).

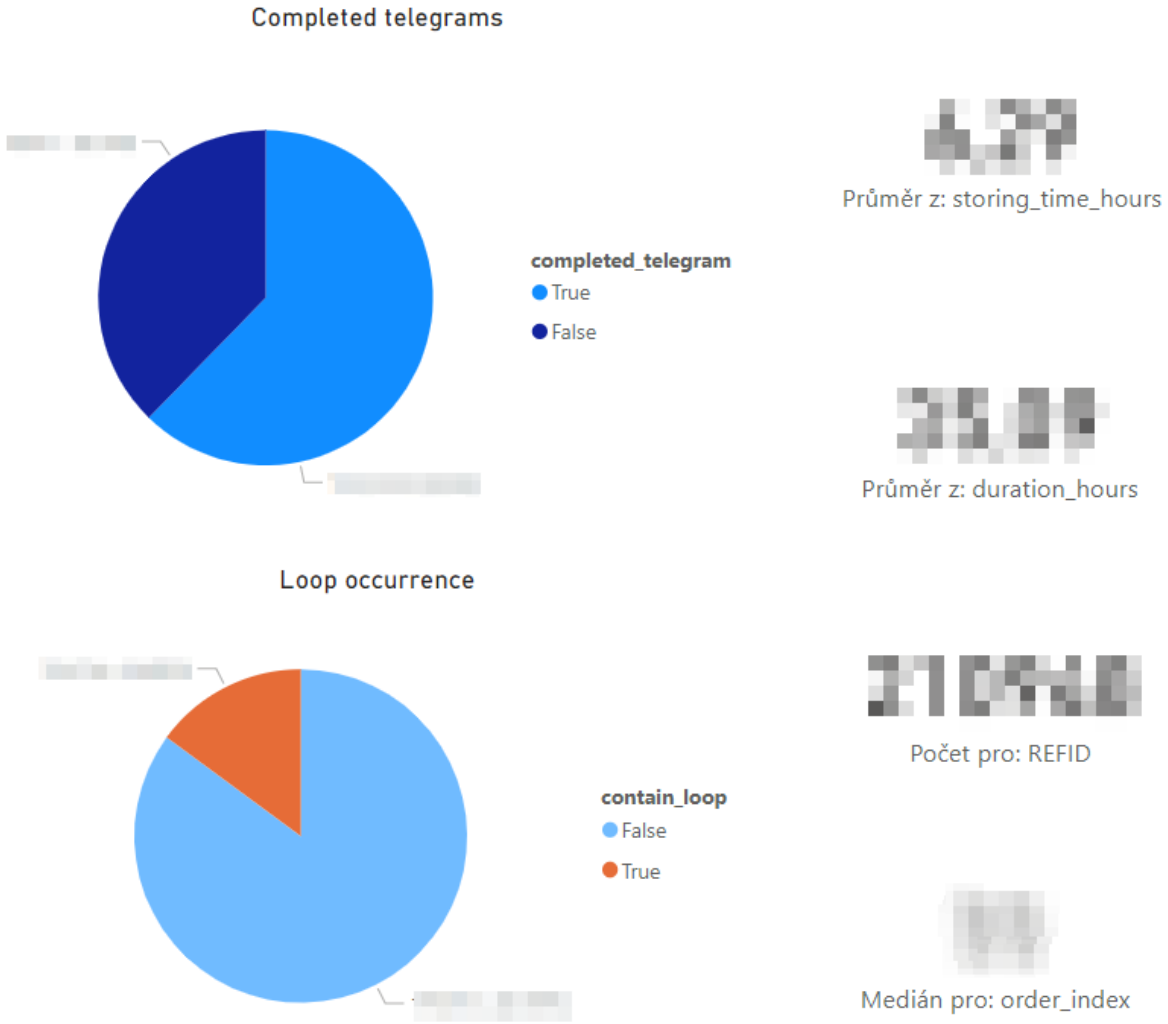
5.2 Analýza dat v Power BI

Po úpravě semi-strukturovaných dat bylo možné přistoupit k jejich analýze a vizualizaci. Detailnější pohled na zpracovaná data umožňuje získat informace, které na první pohled nejsou zřejmé. K této analýze bylo využito softwaru Microsoft Power BI. Vstupem pro analýzu byly dva výše vytvořené soubory s více než 2,5 milionů zaznamenaných událostí ve skladu, které byly získány za období jednoho měsíce. Je nutno dodat, že kvůli omezením ve výrobě, zapříčiněnými koronavirovou pandemií a nedostatky polovodičů, aktivita ve skladu byla značně omezena. Tento fakt však nemá vliv na proces analýzy dat, ani logiku algoritmů při jejich zpracování.

5.2.1 Obecný pohled na zpracovaná data

Na obrázku 23 je zobrazen vztah mezi oběma .csv soubory, které byli mezi sebou propojené pomocí stejného klíče neboli REFID údaje. Díky tomu jsme mohli v Microsoft Power BI zobrazit základní popisné statistické údaje, jako například velikost pozorovaného vzorku, počty kompletních telegramů (KLT přepravky, které úspěšně opustily sklad), počet přepravek, které

opakovaly určité kroky ve skladu a objevovaly se na některých pozicích vícekrát, průměr času naskladnění, nebo průměrnou dobu KLT přepravky strávenou ve skladu.



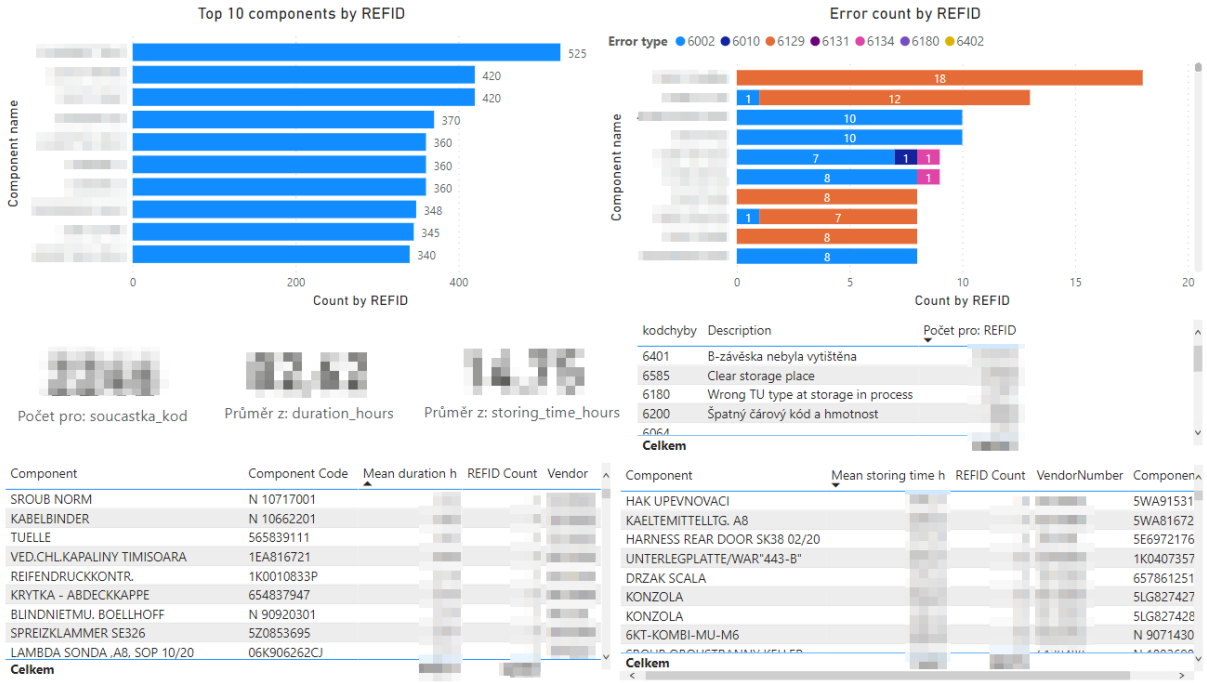
Obrázek 24: *Popisné statistiky Power BI*

Obrázek 24 ukazuje jeden z dashboardů, vytvořených v prostředí Power BI, které poskytují vizuální prezentaci zpracovaných dat. Na grafech můžeme vidět podíl kompletních telegramů, průměrný čas strávený ve skladu, nebo medián kroků, kterými KLT přepravka projde. Tyto informace nabízí obecný pohled na transformovaná data a jejich základní atributy.

5.2.2 Detailní pohled na zpracovaná data

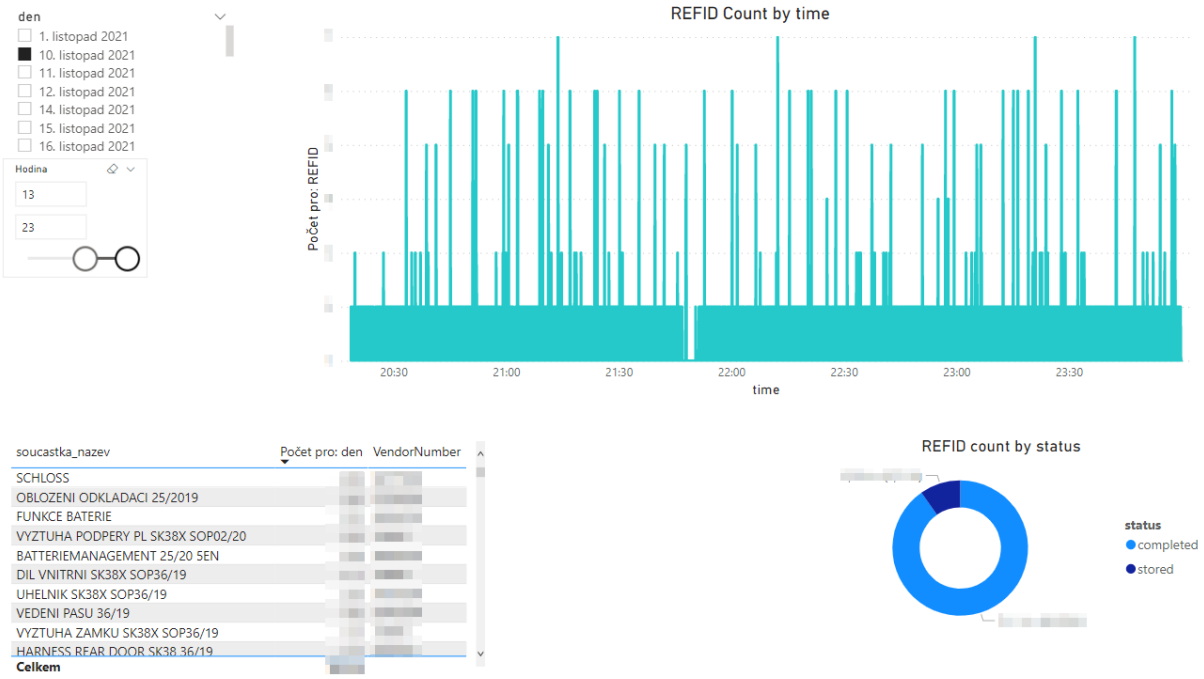
Ze zpracovaných dat bylo vytvořeno několik detailnějších dashboardů. Každý z těchto dashboardů poskytuje podrobnější pohled na dění ve skladu a umožňuje tak, podle aktuálních potřeb, upravovat parametry vyhledávání. Tyto dashboardy, na rozdíl od obecného dashboardu,

který sloužil k vytvoření základního, zjednodušeného pohledu na vstupní data, byly rozděleny do několika částí. Každá část prezentuje data z určitého úhlu pohledu, ať už se jednalo o míru chybovosti napříč putováním KLT přepravky ve skladu dle určitého obsahu dané přepravky, výrobce, nebo času. Vzhledem k citlivosti dat jsou hodnoty v dashboardech záměrně rozostřené.



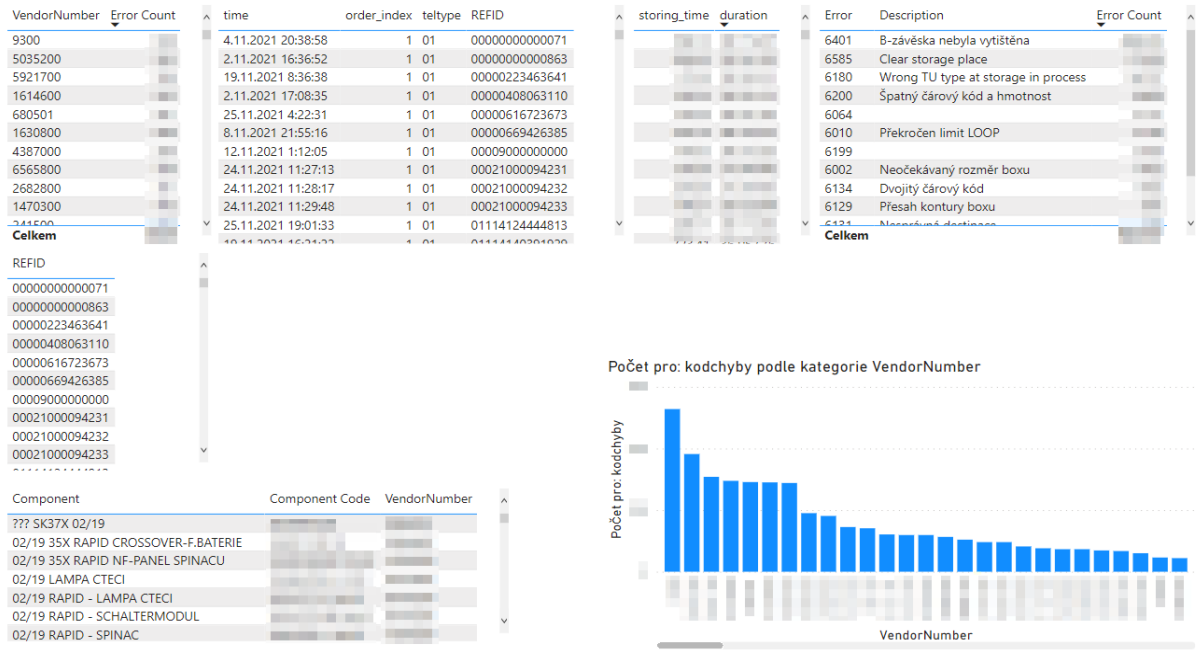
Obrázek 25: *Dashboard Power BI dle přepravovaných komponent*

Na obrázku 25 je zobrazen dashboard, který byl vytvořen za účelem zobrazení dat a jednoduchého vyhledávání v nich dle komponent, které byly přepravovány v jednotlivých KLT přepravkách. Z grafů je možné identifikovat součástky, které byly nejčastěji přepravovány, jejich počet, druh a počet chyb, které se v průběhu přepravy dané komponenty vyskytly. Grafy disponují informací o průměrném čase, který KLT přepravka s konkrétní součástkou strávila ve skladu, nebo dobu za kterou byla KLT s komponentou uložená na určité místo ve skladu. Dashboard disponuje informací o dodavateli, který danou součástku dodává. Všechny tyto údaje umožnily analyzovat data jak z pohledu součástky, průměrného času stráveného ve skladu, dodavatele, nebo chyb, které se vyskytly v průběhu pobytu KLT přepravky ve skladu. Díky tomu, že jsou grafy v Power BI interaktivní, bylo možné jednoduše filtrovat údaje na základě hodnot zobrazených v grafech nebo tabulkách, viz. obrázek 25.



Obrázek 26: *Komponenty a jejich dodavatelé – zobrazení dle času*

Obrázek 26 zobrazuje dashboard, který umožnil jednoduše vyhledávat v datech dle datumu a času. V pravém horním rohu je časová osa s frekvencí aktivit dle vybraného intervalu v levém horním rohu dashboardu. V dolním levém rohu je tabulka s komponenty a jejich dodavateli seřazená dle počtu součástek, které se v daný den nacházely ve skladu. V pravém dolním rohu je pak v podobě donut grafu vidět, jaké procento z těchto komponent již sklad opustilo a jaké procento bylo pouze uloženo do regálu a stále se ve skladu nachází.



Obrázek 27: *Dashboard dodavatelé dle chybovosti a času*

Další dashboard byl vytvořen za účelem sledování dodavatelů komponent, chybovosti a času, které KLT přepravka stráví ve skladu. Analýzu a vyhledávání v datech je možné provádět napříč všemi výše popsanými dashboardy. Taková základní analýza a vizualizace dat poskytuje zajímavý vhled do problematiky a může sloužit jako vstup pro další, složitější a konkrétnější metody analýzy dat, či podporu rozhodování.

5.3 Process Mining

Tato kapitola popisuje aktivity, které předcházely implementaci metod process miningu a prezentuje výstupy, kterých bylo dosaženo. Kapitola se skládá z dvou podkapitol, první popisuje proces přípravy dat a druhá samotnou implementaci a výstupy. Cílem této kapitoly je poukázat na zajímavý způsob, jakým je možné se na data dívat.

5.3.1 Příprava dat pro Process Mining

Pro účely process miningu musí data obsahovat název procesní instance a vykonané aktivity v chronologickém pořadí. Tedy je nutné mít dva základní údaje – aktivita a čas, ve kterém je daná aktivita vykonávána. Strojový log obsahuje informaci o času, ve kterém k aktivitě stroje došlo a za aktivitu je v našem případě považován typ strojového logu. Tabulka 3 znázorňuje formát souboru REFIDtel_export.csv, který byl využit k vytvoření procesního logu. Kvůli množství probíhajících procesů ve skladu byla vstupní data rozdělená na dva soubory. Jeden datový soubor popisoval pohyb KLT přepravky na nižší neboli detailnější úrovni, přesně tak jak jdou za sebou včetně všech pozic, kterými prochází a údaje o cílové destinaci. Druhý datový soubor popisoval pohyb KLT přepravky na vyšší úrovni a každý záznam v sobě obsahoval čas trvání jedné aktivity a pozice na které byla aktivita vykonávána.

Procesní log nižší úrovně

Pro vytvoření procesního logu nižší úrovně bylo opět využito programovacího jazyku Python a jeho Pandas knihovny. Na základě několika podmínek byly záznamy v datovém souboru upravené a označené. Jednalo se například o rozpoznání dvojznačnosti záznamů typu LAM, kdy při náležitosti určitého symbolu v sloupci s údajem o destinaci, bylo možné určit, zda tento záznam se zmíněným typem, znamená uložení KLT přepravky do regálu, nebo naopak, jeho vyjmutí z pozice v regálu, což by vedlo k jejímu dalšímu pohybu.

Záznamy od zakladače obsahující typ LAM byly označeny dle údajů o destinaci a zdroji symboly:

- IN – uložení KLT přepravky na konkrétní místo ve skladu
- OUT – vyjmutí KLT přepravky z konkrétního místa ve skladu
- AP – umístění KLT přepravky na dopravníkový pás

time	REFID	type	zdroj	destinace	cilovadest	chyba
2021-10-31 21:50:51	03121437985142	0017	0017	NaN		
2021-10-31 22:07:34	03121437985142	MPR	WE_PRN06	NaN		
2021-10-31 22:07:34	03121437985142	01	B2MP301	B2SC30224328	B2-PREZONE	
2021-10-31 22:07:45	03121437985142	01	B2SC302	B2SC09115608	B2-PREZONE	
2021-10-31 22:08:09	03121437985142	01	B2SC091	B2SC10106207	B2-PREZONE	
2021-10-31 22:08:09	03121437985142	0001	0001	NaN		
2021-10-31 22:08:09	03121437985142	0002	0002	NaN		
2021-10-31 22:08:20	03121437985142	01	B2SC101	B2SC10116215	B2-PREZONE	
2021-10-31 22:08:36	03121437985142	01	B3SC020	B3SC02020205	B3-AI02	
2021-10-31 22:09:13	03121437985142	01	B3SC020	B3EP02210217		
2021-10-31 22:10:29	03121437985142	LAM	B3EP02	LAM02.P3	B30402800811	
2021-10-31 22:10:51	03121437985142	LAM	LAM02.P3-IN	B30402800811	B30402800811	
2021-11-04 05:53:39	03121437985142	LAM	LAM01.P1-OUT	LAM01.P1	B2-R301	6401
2021-11-04 05:54:17	03121437985142	LAM	LAM01.P1-AP	B4AP02110244	B2-R301	6401
2021-11-04 05:55:01	03121437985142	01	B2SC011	B2SC02116252	B2-R301	6401
2021-11-04 05:55:03	03121437985142	MPR	WA_PRN01	NaN		
2021-11-04 05:55:03	03121437985142	01	B2SC021	B2SC02216253	B2-R301	6401
2021-11-04 05:55:11	03121437985142	01	B2SC052	B2VP25414640	B2VP25414640	
2021-11-04 05:55:19	03121437985142	0010	0010	NaN		
2021-11-04 05:59:02	03121437985142	01	B2SC254	R30200400300	R30200400300	
2021-11-04 06:05:32	03121437985142	0011	0011	NaN		

Tabulka 5: *Procesní log nižší úrovně*

Tabulka 5 obsahuje ukázku části procesního logu nižší úrovně. Většina atributů je stejná, jak již bylo popsáno v předchozích kapitolách. Data byla upravená tak, aby byl procesní log co nejjednodušší a nejjasnější. Z procesního logu je možné vypozorovat pohyb konkrétní KLT přepravky s referenčním id 03121437985142. Sloupec s názvem type poskytuje informaci o činnosti, kdy např. hodnota 01 označuje pohyb mezi jednotlivými body na dopravníku, LAM typ zprávy, kterou posílá zakladač, MPR typ zprávy produkuje tiskárna, typ 0010 potvrzuje konkrétní skladové místo a typ s číslem 0011 poskytuje informaci k odpisu přepravky na místo potřeby, tedy vyskladnění. Z ukázky procesního logu nižší úrovně je také možné získat informaci o chybě, kdy a v jaké fázi procesu k ní došlo. Chyba s číslem 6401 například informuje o nečitelnosti čárového kódu na přepravce.

Procesní log vyšší úrovně

Stejně jako procesní log nižší úrovně, byl vytvořen i procesní log vyšší úrovně pomocí programovacího jazyku Python a knihovny pro nakládání s daty Pandas. Účelem tohoto typu procesního logu bylo poskytnout obecnější pohled na procesy probíhající ve skladu. Vstupními daty pro vytvoření tohoto procesního logu byl soubor ve formátu zobrazeném v tabulce 3.

Jak již bylo zmíněno, k vytvoření procesního logu jsou zapotřebí čas a aktivita. Procesní log vyšší úrovně byl vytvořen na základě typu aktivity, času a pozice, na kterém byla daná aktivita provedena. Tento log byl vytvořen pomocí jednoduchého skriptu, který procházel již zpracovaná data a dle referenčního id, typu zprávy a zdroje vytvářel nové atributy start_time a end_time. Díky těmto dvěma údajům bylo pak možné na první pohled vidět, jak dlouho konkrétní aktivita na dané pozici trvala.

start_time	end_time	refid	type	zdroj
2021-10-31 21:50:51	2021-10-31 21:50:51	03121437985142	0017	0017
2021-10-31 22:07:34	2021-10-31 22:07:34	03121437985142	MPR	WE_PRN06
2021-10-31 22:07:34	2021-10-31 22:08:09	03121437985142	01	B2MP301
2021-10-31 22:08:09	2021-10-31 22:08:09	03121437985142	0001	0001
2021-10-31 22:08:09	2021-10-31 22:08:09	03121437985142	0002	0002
2021-10-31 22:08:20	2021-10-31 22:09:13	03121437985142	01	B2SC101
2021-10-31 22:10:29	2021-10-31 22:10:51	03121437985142	LAM	LAM02.P3-IN
2021-11-04 05:53:39	2021-11-04 05:54:17	03121437985142	LAM	LAM01.P1-OUT
2021-11-04 05:55:01	2021-11-04 05:55:01	03121437985142	01	B2SC011
2021-11-04 05:55:03	2021-11-04 05:55:03	03121437985142	MPR	WA_PRN01
2021-11-04 05:55:03	2021-11-04 05:55:11	03121437985142	01	B2SC021
2021-11-04 05:55:19	2021-11-04 05:55:19	03121437985142	0010	0010
2021-11-04 05:59:02	2021-11-04 05:59:02	03121437985142	01	B2SC254
2021-11-04 06:05:32	2021-11-04 06:05:32	03121437985142	0011	0011

Tabulka 6: *Procesní log vyšší úrovně*

Tabulka 6 popisuje pohyb KLT přepravky s referenčním číslem 03121437985142. Z ukázky procesního logu vyšší úrovně je možné sledovat proces od jeho začátku (zpráva typu 0017, která oznamuje vstup přepravky do skladu) až k odpisu přepravky na místo potřeby, tedy vyskladnění.

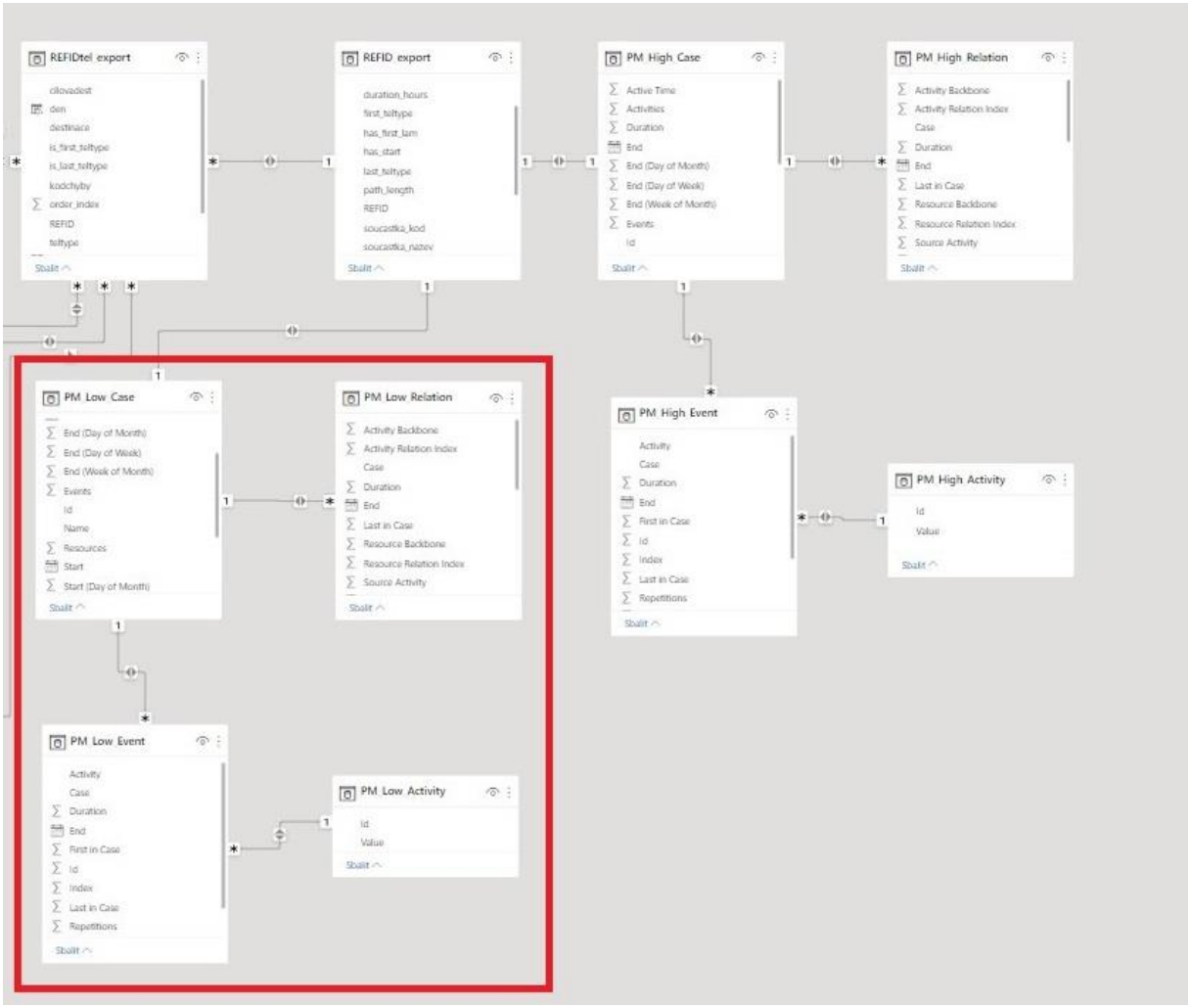
5.3.2 Zpracování procesních logů

Po přípravě dat a vytvoření dvou typů procesních logů následovalo jejich zpracování a vytvoření tzv. procesní mapy. Data byla zpracována algoritmy pro účely process miningu, jejichž výstupem byly čtyři soubory pro procesní log vyšší úrovně (PM_High_Case, PM_High_Event, PM_High_Relation a PM_High_Activity) a čtyři soubory pro procesní log nižší úrovně stejného formátu.

Obrázek 28: *Power BI process mining high case [vlastní zpracování]*

Na obrázku 28 výše jsou zvýrazněné relace mezi jednotlivými soubory vyšší úrovně, které byly vytvořeny algoritmy pro process mining. Veškeré relace jsou vytvořeny pomocí referenčního čísla, které je unikátní pro každou KLT přepravku. PM_High_Case představuje pohyb jedné KLT přepravky ve skladu a zapouzdřuje PM_High_Event, který reprezentuje jeden konkrétní pohyb přepravky. Jak je z modelu vidět, jedna KLT přepravka s unikátním referenčním

identifikačním číslem může obsahovat jeden a více pohybů (relace 1:* mezi PM_High_Case a PM_High_Event). Relace 1:* mezi PM_High_Activity a PM_High_Event říká, že jedna aktivita může být provedena vícekrát. PM_High_Relation pomáhá vzniku procesní mapy a umožňuje tak sledovat procesy mezi jednotlivými KLT přepravkami.

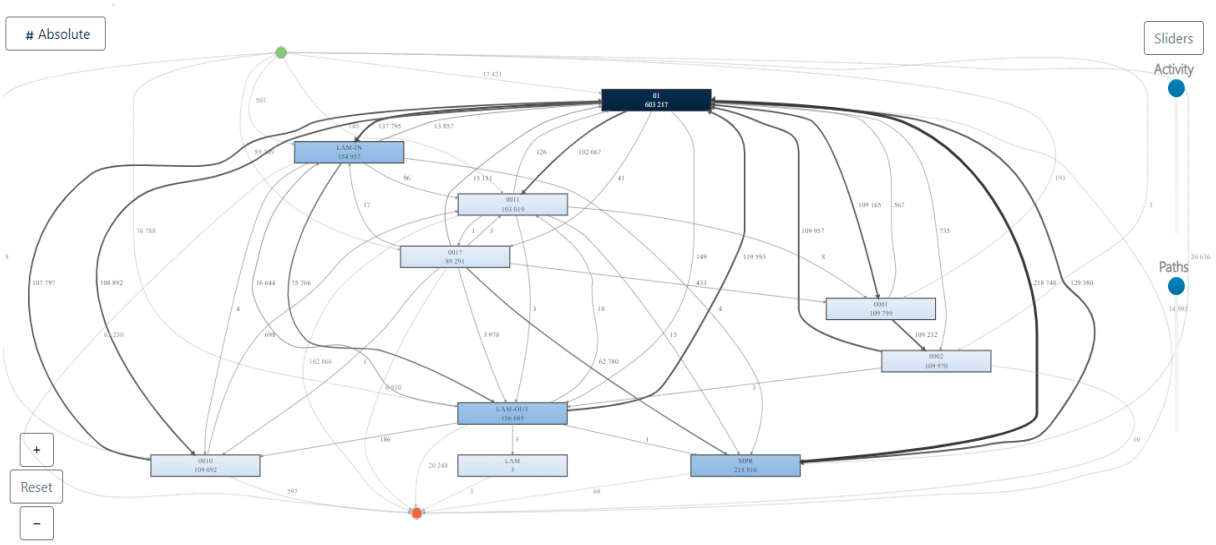


Obrázek 29: *Power BI process mining low case [vlastní zpracování]*

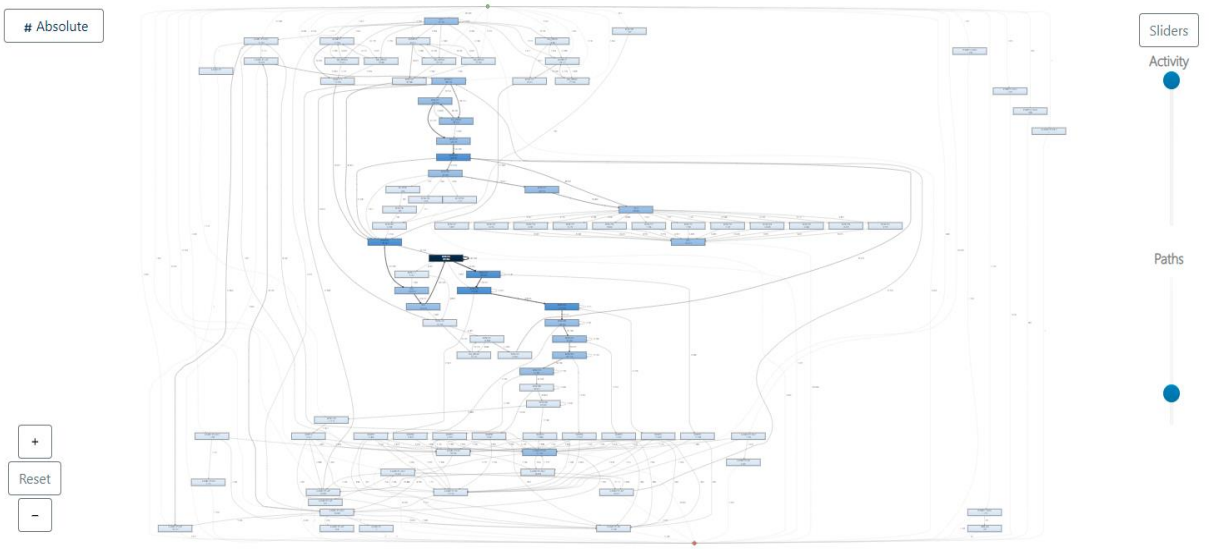
Na obrázku 29 jsou zvýrazněné relace mezi zpracovanými procesními logy nižší úrovně. Jedná se o stejný princip, jako je tomu u předchozího obrázku s rozdílem interpretace dat, kdy je model zaměřen na detailnější popis procesů, než je tomu u vyšší úrovně. Oba obrázky vyobrazující jednotlivé části datového modelu pochází z prostředí Power BI a vstupními daty jsou jak zpracovaná data, vytvořená v průběhu transformace logových zpráv, s cílem jejich analýzy, tak i zpracované procesní logy, které sloužily jako vstup pro vizualizaci procesů pomocí procesní mapy. Klíčovým atributem pro vznik relací mezi jednotlivými daty je referenční id (REFID), časový údaj a typ prováděného procesu (type).

5.3.3 Procesní mapa

K vizualizaci a analýze procesů v centrálním skladu bylo využito Power BI vizuálu pro procesní mapy. Vstupními daty pro procesní mapu byly zpracované procesní logy v podobě výše zmíněných souborů. Díky této vizualizaci bylo možné sledovat konkrétní procesy, jejich trvání a místa, ve kterých se odehrávaly



Obrázek 30: *Procesní mapa veškerých aktivit skladu (vyšší úroveň) [vlastní zpracování]*

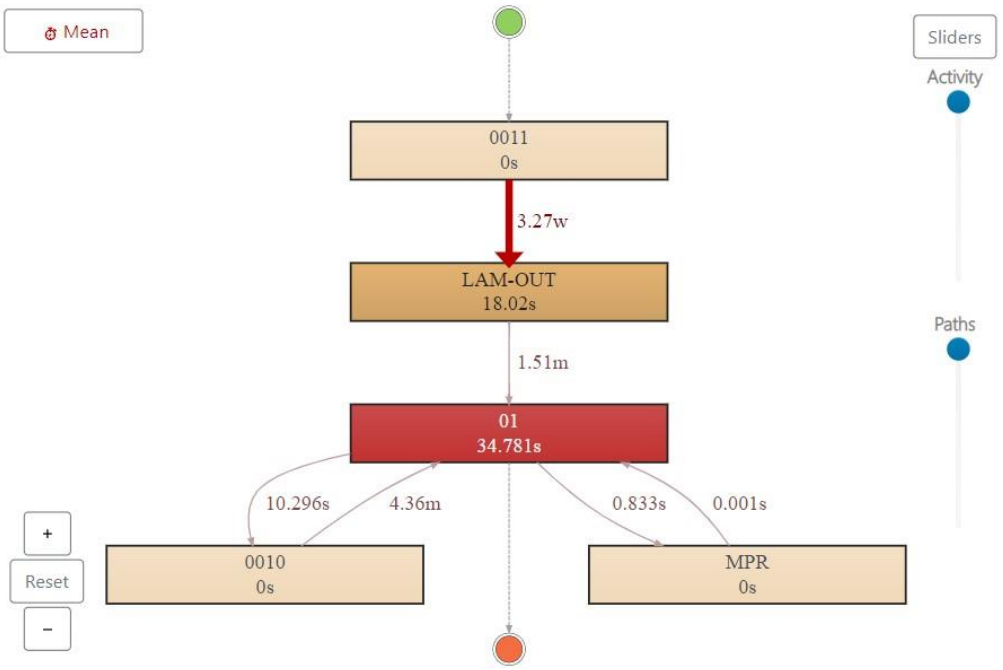


Obrázek 31: *Procesní mapa veškerých aktivit (nižší úroveň) [vlastní zpracování]*

Obrázky 30 a 31 zachycují veškeré probíhající aktivity ve skladu. U procesní mapy nižší úrovně není možné, kvůli velkému počtu relací mezi aktivitami, zobrazit jejich kompletní počet. Jelikož je obecný pohled složitý, je vhodné pro analýzu probíhajících procesů filtrovat aktivity dle možných atributů. Tyto dva obrázky výše jsou ilustrací složitosti procesu.

V Power BI po implementaci procesní mapy vzniklo několik dashboardů vycházejících ze scénářů, které bylo možné pozorovat:

- Case x Component a Case duration – Zobrazení jednotlivých součástek dle průměrného času mezi pozicemi. Bylo tak možné pozorovat nejdelší čas mezi jednotlivými pozicemi.

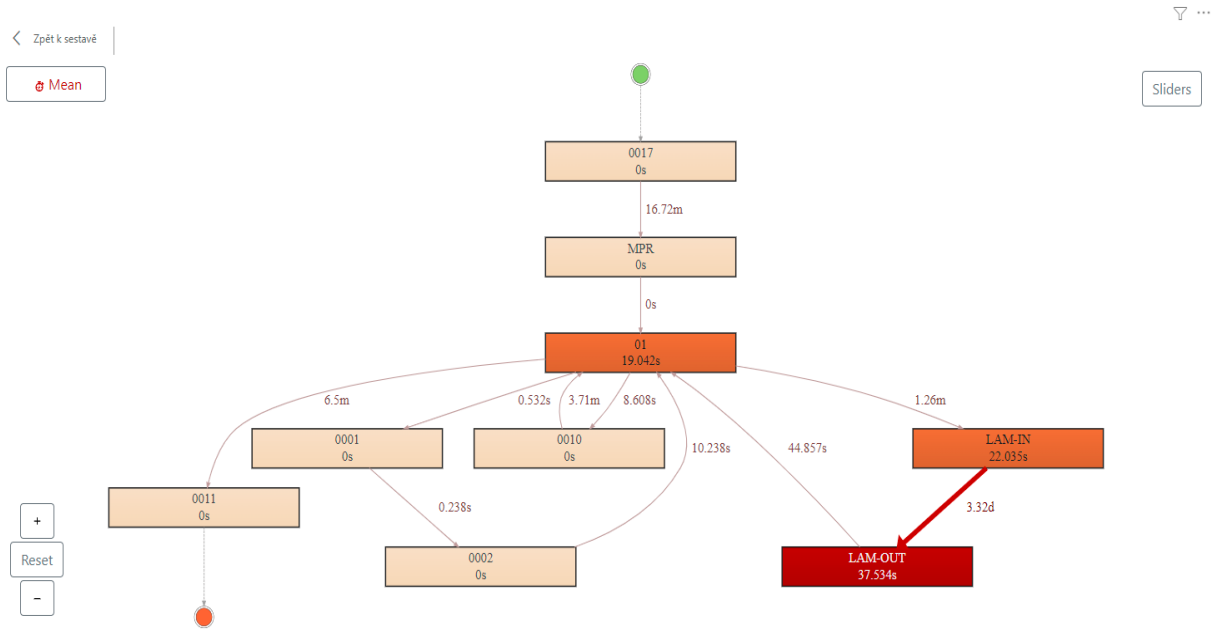


Obrázek 32: *Procesní mapa – Case x Component a Case duration (vyšší úroveň) [vlastní zpracování]*

Na obrázku 32 je možné vidět veškeré kroky, kterými prošla konkrétní KLT přepravka. Jelikož by byl model procesní mapy v případě zobrazení veškerých aktivit probíhajících ve skladu příliš složitý, je zapotřebí filtrovat a vybírat tak konkrétní KLT přepravky, komponenty, nebo chyby, které jsou předmětem zájmu. Díky provázanosti veškerých zpracovaných dat podle referenčního id (REFID), je možné filtrovat jednotlivé přepravky a komponenty a analyzovat tak jejich pohyb skladem. Tato procesní mapa umožňuje zobrazit údaje jako jsou například, průměrná, maximální či minimální doba trvání procesu. Hodnoty je možné vybrat v levém horním rohu. Další výhodou této procesní mapy je detail, ve kterém je možné se na probíhající procesy dívat. Ten lze nastavit v pravém horním rohu (Activity – pro zobrazení veškerých aktivit konkrétní přepravky, Paths – pro zobrazení veškerých přechodů mezi aktivitami). Na obrázku proces začíná v zeleném bodě a končí červeným bodem. Konkrétní proces z obrázku výše je možné považovat za anomální a je potřeba ho detailněji analyzovat a prověřit, protože aktivita číslo 0011 signalizuje vyskladnění KLT přepravky a neměl by po ní následovat

další pohyb. Zde je však možné vidět, že přepravka byla vrácena na tiskárnu a nemáme údaj o jejím dalším pohybu.

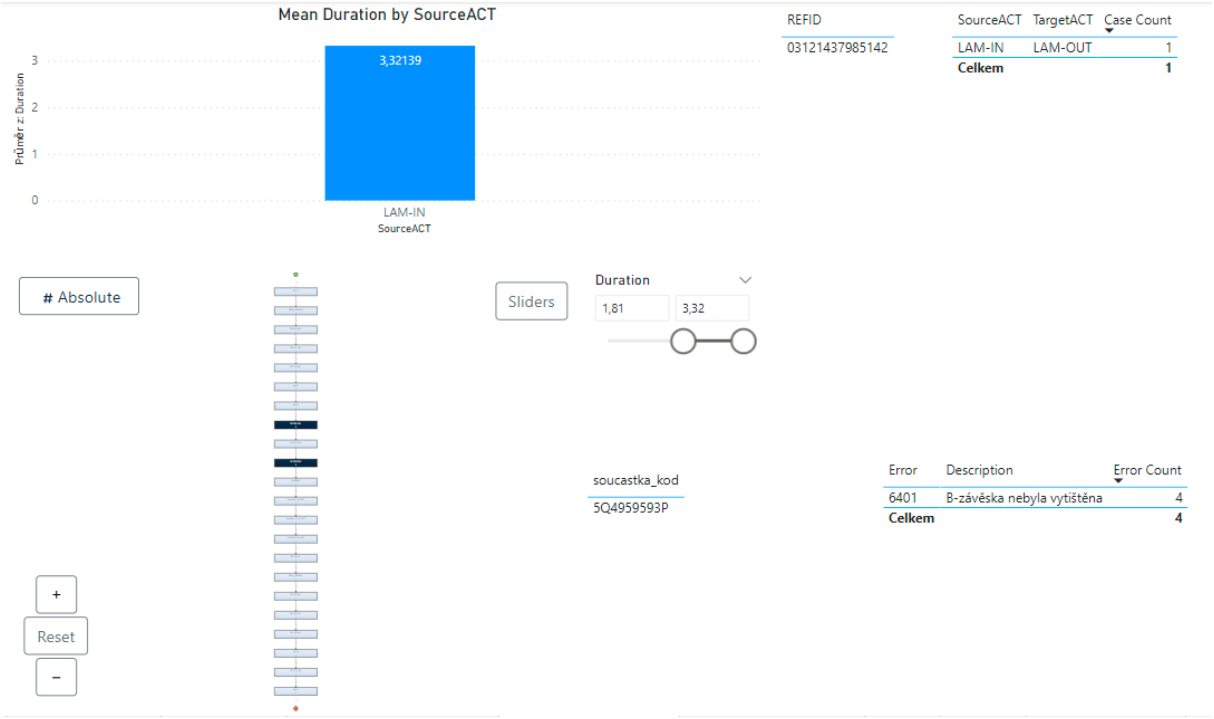
- Chyby komponent dle pozic a pohybu ve skladu – Umožňuje detailní pohled na součástky s největším počtem chyb a zobrazení průměrných a maximálních časů mezi pozicemi podle součástky, díky čemuž je možné jednoduše dohledat součástky s největším počtem chyb. Analýza takových procesů může odhalit nežádoucí chování robotů ve skladu.



Obrázek 33: *Procesní mapa – Kompletní pohyb KLT přepravky (vyšší úroveň) [vlastní zpracování]*

Obrázek 33 popisuje pohyb konkrétní KLT přepravky s referenčním id 03121437985142. Tentokrát se jedná o očekávané chování, kdy proces začíná aktivitou 0017, prochází veškerými potřebnými kroky, včetně uložení přepravky do regálu, její vyložení, označení informačním štítkem a vyskladnění. Relace mezi aktivitami jsou zobrazeny v průměrných časech se zvýrazněním nejdéle trvajících aktivit.

- Nejdelší relace – Možnost zobrazení nejdelší relace podle průměrné doby trvání a následné analýzy prostojů, či anomálií mezi jednotlivými procesy a odhalení potenciálních problémů.



Obrázek 34: *Power BI dashboard nejdelší relace (nižší úroveň) [vlastní zpracování]*

Na obrázku 34 je možné vidět dashboard, který obsahuje sloupcový graf s údaji o nejdelší průměrné délce relace konkrétní KLT přepravky (REFID 03121437985142). Nejdelší relace této přepravky je logicky mezi pozicemi LAM-IN (umístění přepravky do regálu) a LAM-OUT (následné vyjmutí přepravky). V levém dolním rohu je procesní mapa nižší úrovně, která popisuje cestu KLT přepravky skladem krok po kroku. Z důvodu podrobnosti procesní mapy nižší úrovně není prezentován samostatný obrázek, který by byl kvůli množství kroků jednotlivé KLT přepravky nečitelný. Pokud by nebyl tento dashboard omezen pouze na konkrétní přepravku, bylo by možné přímo ve vizuálu filtrovat například pouze doby strávené ve skladu nebo chyb, které obsahují.

Process mining přináší velmi zajímavý pohled na dostupná data. Díky představené implementaci je možné se dívat na jednotlivé aktivity z různých stran a analyzovat tak dění ve skladu. Mezi hlavní výhody tohoto řešení patří bezesporu možnost detailní analýzy a přepínání pohledů mezi vyšší a nižší úrovní zobrazení, jednoduchá grafická reprezentace a flexibilita práce s dostupnými daty. Zmíněné řešení disponuje větším počtem dashboardů, než jsou v této kapitole představeny a jejich popis by zabral značnou část diplomové práce.

5.4 Machine Learning

Tato kapitola popisuje data a jejich přípravu pro implementaci modelů strojového učení, implementaci strojového učení a následnou evaluaci modelů. Kapitola je rozdělena do dvou částí. V první části této kapitoly jsou popsána vstupní data a představeny cíle. Druhá část se zabývá nasazením modelů strojového učení na dvě odlišné datové sady, popis implementace a evaluace dosažených výs[]ledků.

5.4.1 Data

Z úpravy výše prezentovaných dat, popisujících aktivity v centrálním skladu ŠKODA AUTO, a. s. byla vyfiltrována data s informací o jednotlivých chybách, které se mohly v průběhu pohybu jednotlivé KLT přepravky skladem vyskytnout. Data tedy obsahovala informaci o typu chyby a čase, ve kterém k dané chybě došlo. Data byla získána za období 7 měsíců (od 1.11.2021 do 3.6.2022) a jejich podobu demonstruje tabulka níže. Vzhledem k citlivosti dat byly veškeré numerické údaje anonymizované. Tato úprava nijak nenarušila vypovídací hodnotu níže popsaných zjištění.

time	error code
01.11.2021 0:13	6129
01.11.2021 0:13	6129
01.11.2021 0:18	7702
01.11.2021 0:22	6129
01.11.2021 0:22	6129
01.11.2021 0:23	6002
01.11.2021 0:23	6002
01.11.2021 0:29	6129
01.11.2021 0:31	7702
01.11.2021 0:33	7702

Tabulka 7: *Časová řada (čas a typ chyby)*

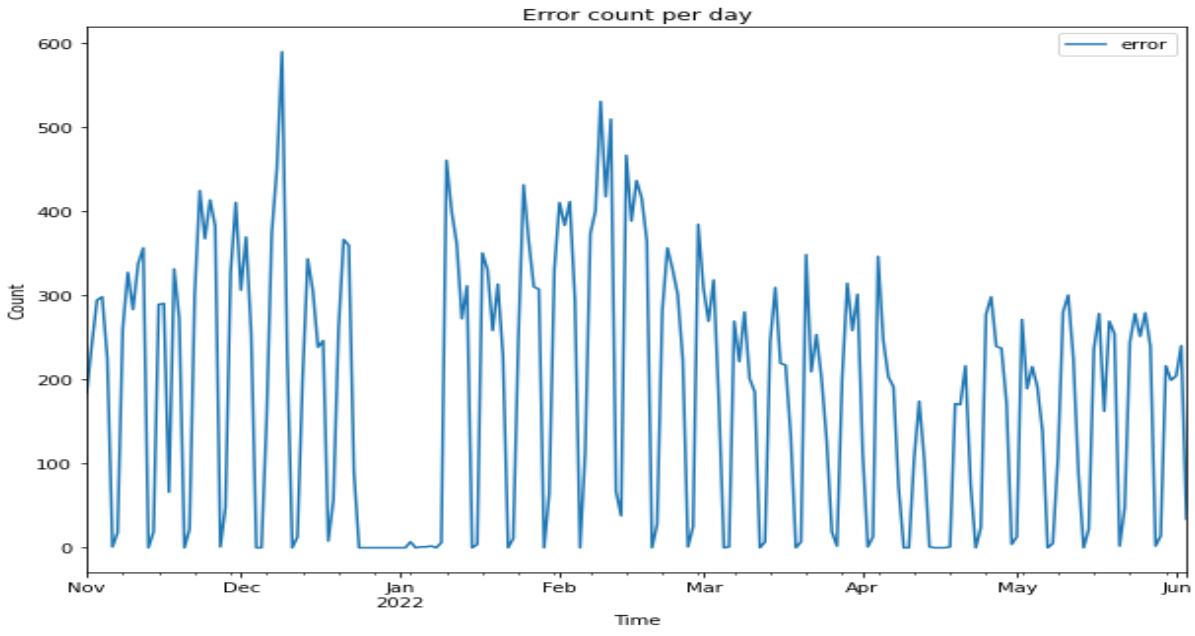
Z těchto dat lze jednoduše získat údaj o počtu chyb, které vznikly za jednotlivý den. Na základě tohoto údaje je možné predikovat očekávané množství chyb v následujících dnech. Cílem implementovaného statistického modelu tedy bylo získání odhadu očekávaného počtu chyb v následujících dnech a také předpovědět četnost typu chyby. Pro predikci takové časové řady bylo využito, již v teoretické části zmíněného, modelu ARIMA. S ohledem na sezónní charakter vstupních dat byl model ARIMA porovnán s tzv. modelem SARIMA, který umožňuje nastavení sezónní složky jako jednoho ze vstupních parametrů.

5.4.2 Predikce počtu chyb

Před implementací modelů bylo zapotřebí data upravit. Z dat byly odebrány nežádoucí hodnoty, jako například nulové hodnoty či kód chyby číslo 6401, která informuje o nečitelnosti kódu na přepravce. Tato chyba se vyskytuje poměrně často a není zapotřebí věnovat tomuto typu chyby zvýšené pozornosti. Po očistění dat o nežádoucí hodnoty následovalo jejich tzv. „převzorkování“, kdy došlo k součtu výskytu chyb v 24hodinových intervalech. Veškeré úpravy dat probíhaly za pomoci python knihovny Pandas. V tabulce níže je zobrazená podoba časové řady, která je uložená do tzv. dataframe – datové struktury python knihovny Pandas s názvem df. Obrázek 35 zobrazuje stejná data pomocí liniového grafu.

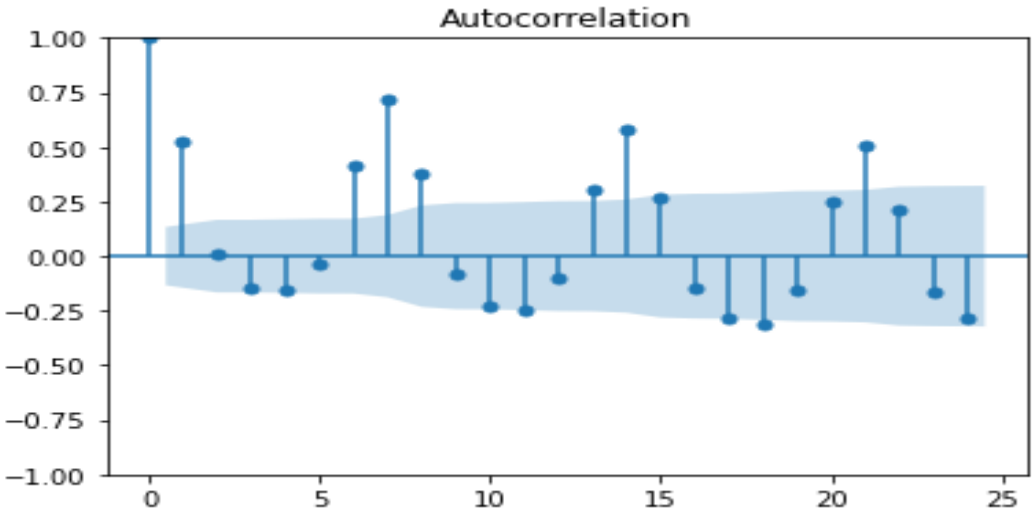
time	error count
01.11.2021	184
02.11.2021	242
03.11.2021	294
04.11.2021	298
05.11.2021	225
06.11.2021	1
07.11.2021	18
08.11.2021	260
09.11.2021	327
10.11.2021	283

Tabulka 8: *Upravená časová řada (počet chyb v 24hodinových intervalech)*

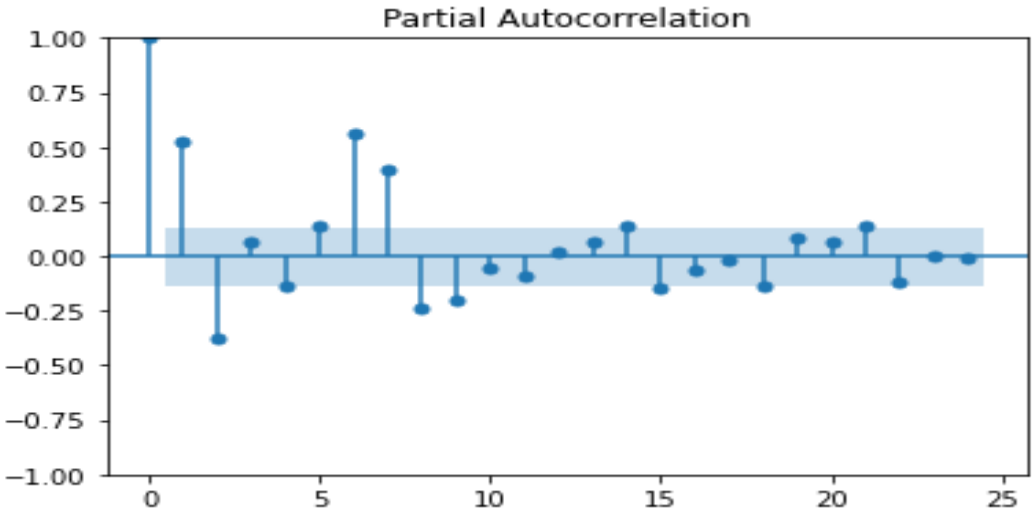


Obrázek 35: *Vizualizace časové řady [vlastní zpracování]*

ARIMA model použitý k analýze a predikci vývoje časových řad, popsaný v teoretické části práce, závisí na vstupních parametrech p, d, q. K odhadu parametru autoregresní části modelu p, je možné využít autokorelační (ACF) funkci. Pomocí parciálně-autokorelační (PACF) funkce lze odhadnout parametr q. Doporučuje se, aby se hodnota parametrů p a q rovnala hodnotě funkce v místě prvního průniku tzv. intervalu spolehlivosti. Třetí vstupní parametr d určuje míru diference při absenci stacionarity, viz. teoretická část (4.3.3) v případě vstupních dat se jedná o hodnotu 1.



Obrázek 36: Graf ACF pro vstupní data [vlastní zpracování]



Obrázek 37: Graf PACF pro vstupní data [vlastní zpracování]

Vzhledem k charakteru dat a nepravidelnosti výskytů chyb, s výjimkou nízké, někdy i nulové aktivity ve skladu během víkendů a svátků je vidět, že výše zobrazené grafy nejsou jednoznačné. Obě funkce zobrazují odlehlé od intervalu spolehlivosti hodnoty, se znaky sezónnosti v podobě 5denního pracovního týdnu. Z grafů je však možné odhadnout parametry modelu ARIMA, kdy parametry p a q by mohly mít vstupní hodnotu 2 nebo 3.

Rozdělení datového souboru

Při implementaci modelů, které využívají tzv. učení s učitelem, viz. kapitola 4.1.1, je vhodné rozdělit datový soubor na minimálně dvě množiny – trénovací a testovací. Použitý datový soubor se skládá z 215 záznamů, kdy každý záznam reprezentuje jeden den. Jelikož tato datová sada není velká, bylo zvoleno rozdělení 80-20, kdy trénovací množina tvoří 80 % záznamů z celkového datového souboru a testovací množina 20 %.

Implementace modelu ARIMA

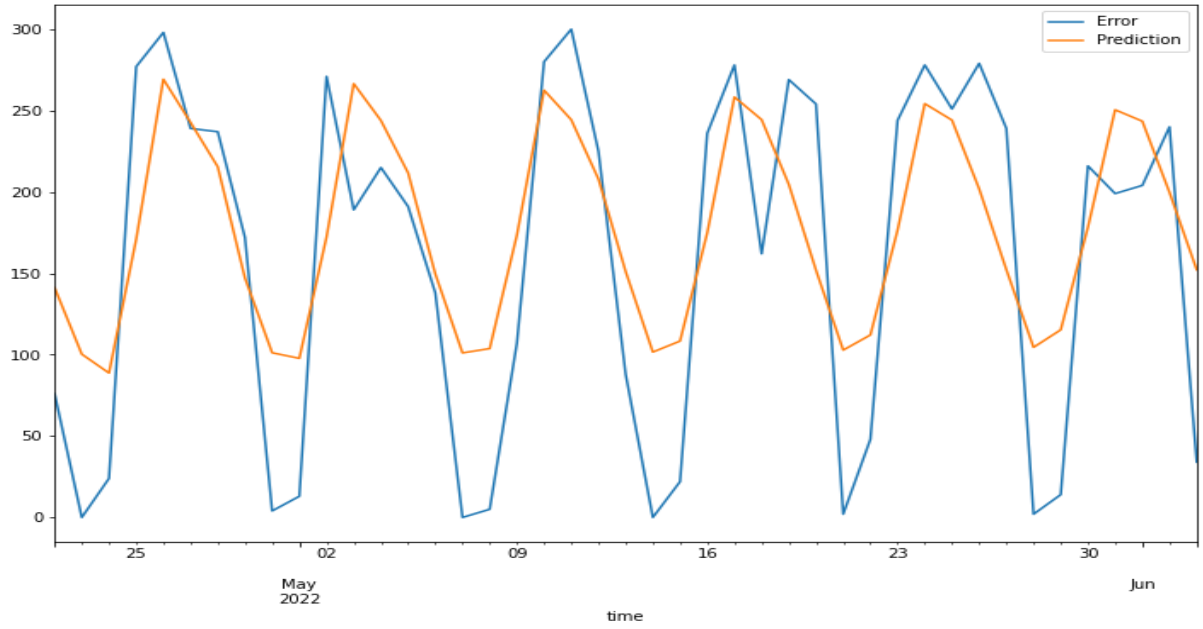
Model ARIMA je součástí python balíčku statsmodels a je ho tedy možné jednoduše importovat. Následuje natrénování modelu pomocí umístění trénovací části datového souboru do proměnné modelu a uvést výše odhadované parametry. Jelikož datová sada není tak rozsáhla a testovací množina tvoří velmi sezónní část, a trénovací množina obsahuje vánoční a svátky, výše odhadované parametry je třeba nahradit. Po testování různých vstupních parametrů, byly zvoleny hodnoty p, d, q (7, 1, 7).

```
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(train,order=(7,1,7))
model_results = model.fit()
prediction = model_results.predict(len(train), end=len(df)-1)
```

time	error count	predicted
22.04.2022	76	141.166532
23.04.2022	0	100.305932
24.04.2022	24	88.732596
25.04.2022	277	170.935769
26.04.2022	298	269.266886

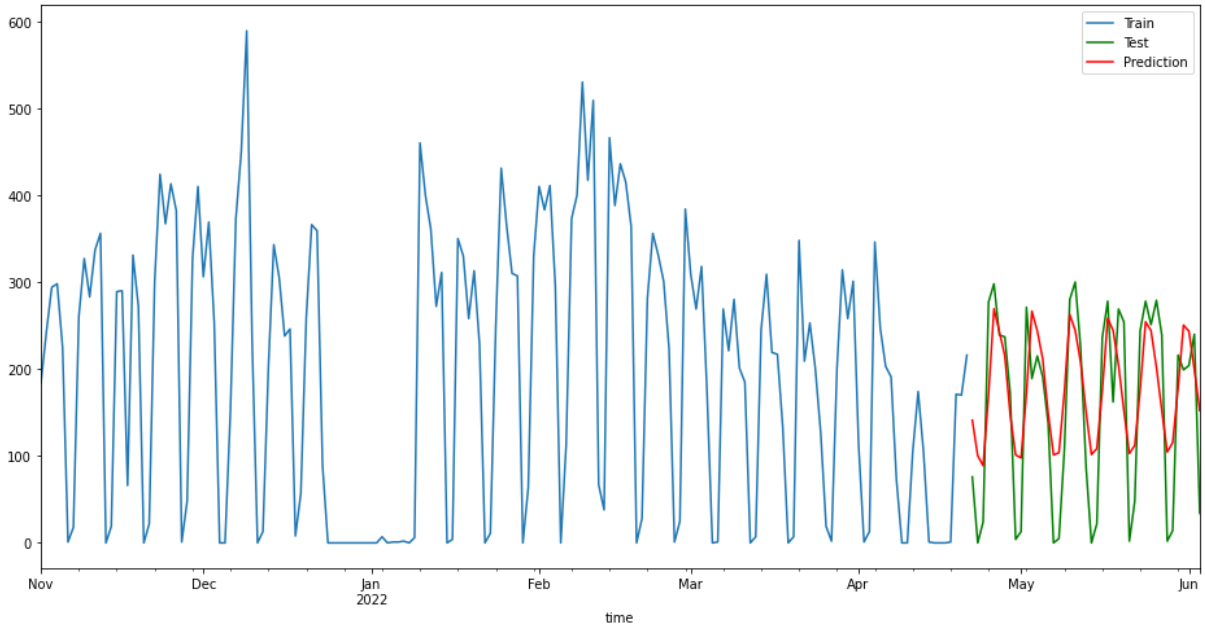
Tabulka 9: *Obsah dataframu prediction*

Tabulka 9 obsahuje časový údaj, počet chyb za jeden den a modelem předpokládanou hodnotu v daném čase. Výše zobrazený kód popisuje implementaci modelu na trénovací data se vstupními parametry a následnou predikci na základě trénovacího souboru.



Obrázek 38: *ARIMA model – testovací množina [vlastní zpracování]*

Na obrázku 38 je zobrazena shoda predikovaných hodnot (oranžová linie) s testovacími daty (modrá linie) po natrénování modelu ARIMA na trénovací množině.



Obrázek 39: *ARIMA model – trénovací a testovací množina [vlastní zpracování]*

Obrázek 39 shrnuje trénovací množinu dat (modrá linie), testovací (zelená) a predikci (červená) v jednom grafu. Na ose y se nachází počet chyb a ose x čas.

Jak je již z grafů výše vidět, predikované hodnoty počtu chyb za den se značně liší od skutečných hodnot v testovací množině datového souboru. Model však nikdy nemůže být naprosto shodný se skutečnými daty, jednalo by se tak o tzv. přeučení modelu.

K vyhodnocení přesnosti modelu bylo využito následujících metrik:

- Střední absolutní chyba – neboli MAE (Mean Absolut Error) je ukazatelem odchylky predikované hodnoty od hodnoty skutečné. Střední absolutní chyba je definována následujícím vztahem:

$$MAE = \frac{1}{N}\sum|(y - \hat{y})|$$

- Střední kvadratická chyba – neboli MSE (Mean Squared Error) je dalším jednoduchým a často používaným údajem při evaluaci regresních modelů. Hodnota MSE reprezentuje odchylku predikce od skutečné hodnoty v druhé mocnině a penalizuje tak vzdálené predikované hodnoty.

$$MSE = \frac{1}{N}\sum(y - \hat{y})^2$$

- RMSE chyba – jedná se odmocninu střední kvadratické chyby, díky které je možné pozorovat odchylku v měřítku hodnot původního datového souboru.

$$RMSE = \sqrt{\frac{1}{N}\sum(y - \hat{y})^2}$$

- Koeficient determinace – odpovídá na otázku, jaká část variability proměnných byla regresním modelem vysvětlena. Pro výpočet je třeba určit veličinu udávající celkovou variabilitu výsledkové proměnné:

$$S_T = \sum_{i=1}^{n}(Y_i - \bar{Y})^2$$

a veličinu udávající nevysvětlenou variabilitu výsledkové proměnné (reziduální součet čtverců). Hodnota koeficientu determinace R^2 se nejčastěji uvádí v procentech.

$$S_e = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$R^2 = 1 - \frac{S_e}{S_T}$$

Vlivem častých výkyvů v datech je přesnost implementovaného modelu ARIMA nízká. Maximální chyba na testovacích datech po natrénování modelu je 118.185. Jedná se o velkou odchylku odhadu od skutečných hodnot. Výsledné metriky nad testovací množinou dat jsou zobrazeny v tabulce níže.

Veličina	Hodnota
MAE	62.549
MSE	5015.608
RMSE	70.820
R^2	0.571

Tabulka 10: *Metriky modelu ARIMA na testovací množině*

Implementace modelu SARIMA

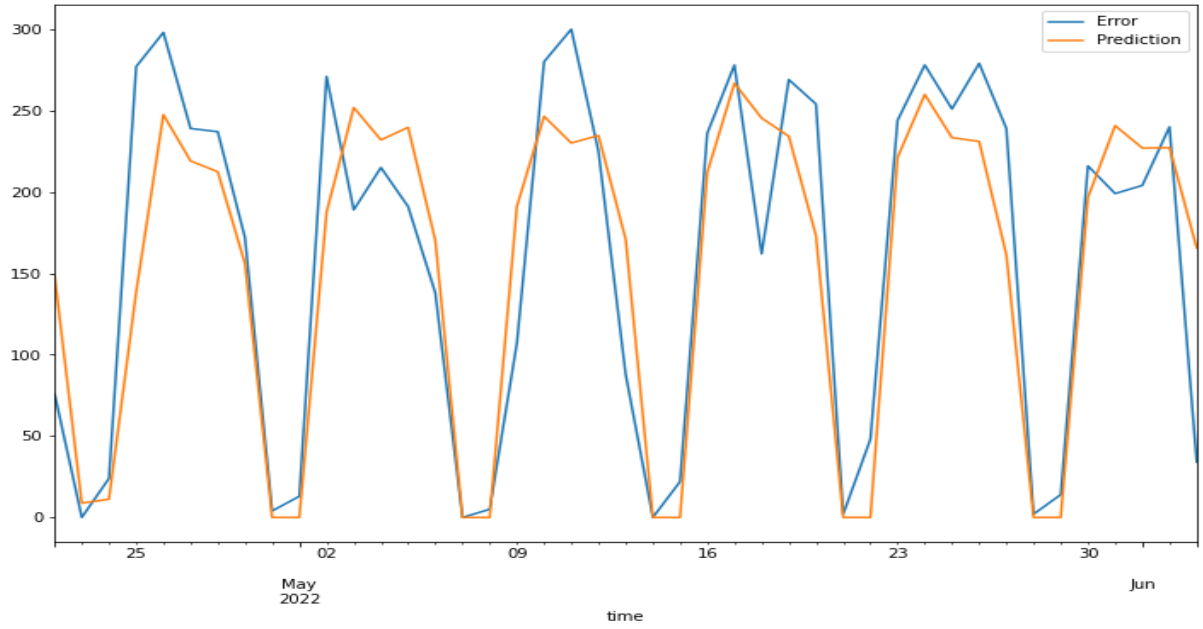
Model SARIMA je stejně jako ARIMA součástí python balíčku statsmodels, ve kterém je pojmenován jako SARIMAX, kde se X uvádí z důvodu podpory exogenních proměnných. Model SARIMAX má navíc tři vstupní parametry jejichž účelem je definovat sezónní složku. Jedná se o sezónní AR složku, sezónní integraci, sezónní MA složku a periodicitu. Po testování několika vstupních parametru dosáhla nejlepších výsledků kombinace (5,1,3,7). Periodicita je nastavená na délku jednoho týdne.

```
import statsmodels.api as sm
seasonal_model = sm.tsa.statespace.SARIMAX(train, order=(5,1,3), seasonal_order=(5,1,3,7))
seasonal_results = seasonal_model.fit()
seasonal_forecast = seasonal_results.predict(start=len(train), end=len(df)-1)
```

time	error count	predicted
22.04.2022	76	148.679938
23.04.2022	0	8.779168
24.04.2022	24	11.202606
25.04.2022	277	138.556813
26.04.2022	298	247.485695

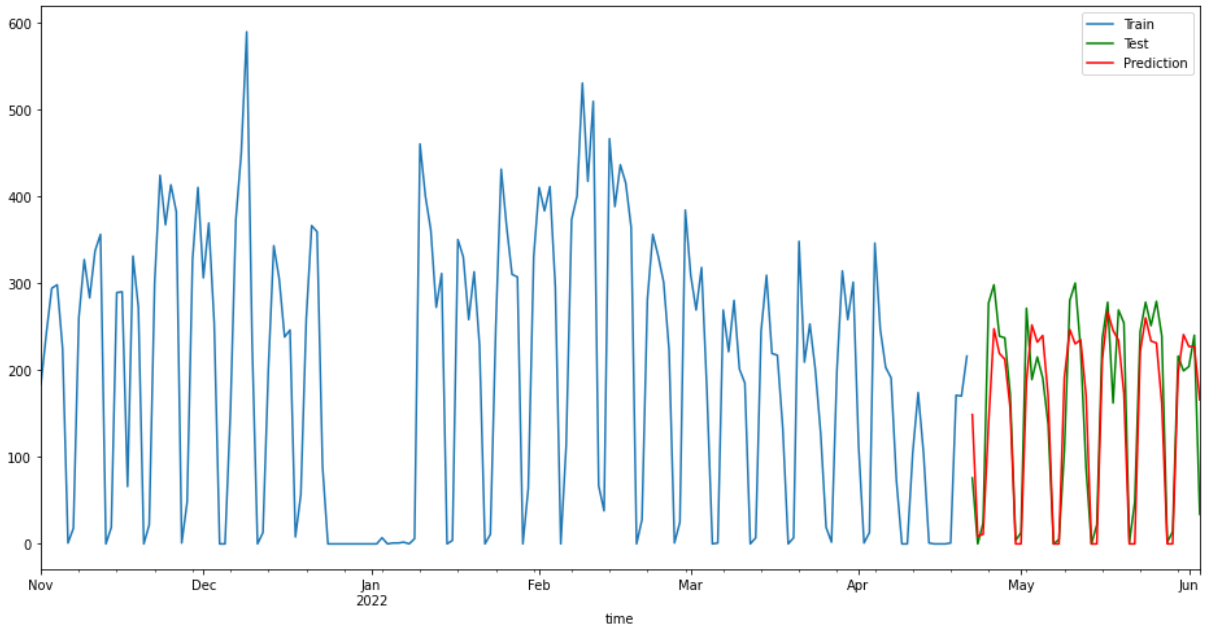
Tabulka 11: *Obsah dataframu seasonal_forecast*

Tabulka 11 obsahuje časový údaj, počet chyb za jeden den a modelem předpokládanou hodnotu v daném čase. Výše zobrazený kód popisuje implementaci modelu na trénovací data se vstupními parametry a následnou predikci na základě trénovacího souboru.



Obrázek 40: *SARIMA model – testovací množina [vlastní zpracování]*

Na obrázku 40 je graficky znázorněn odhad počtu chyb na testovací množině. Model SARIMA predikoval místy záporné hodnoty, a proto byl výsledek tohoto modelu upraven a záporné hodnoty byly nahrazeny nulou.



Obrázek 41: *SARIMA model – trénovací a testovací množina [vlastní zpracování]*

Z grafu na obrázku 41 je vidět, že model SARIMA si poradil s odhadem testovací množiny lépe než předchozí model ARIMA, který neumožňuje ve svých parametrech zohlednit sezónní složku.

Lepší výsledky modelu SARIMA na testovací množině jsou vidět i v metrických ukazatelích. Při implementaci SARIMA bylo dosaženo vyšší maximální odchylky predikovaných hodnot od skutečných hodnot v testovací množině, a to 132,443. Metriky pro porovnání predikovaných hodnot modelu SARIMA se skutečnými hodnotami z testovacího souboru dat jsou zobrazeny v tabulce 12.

Veličina	Hodnota
MAE	39.437
MSE	2655.280
RMSE	51.529
R^2	0.772

Tabulka 12: *Metriky modelu SARIMA na testovací množině*

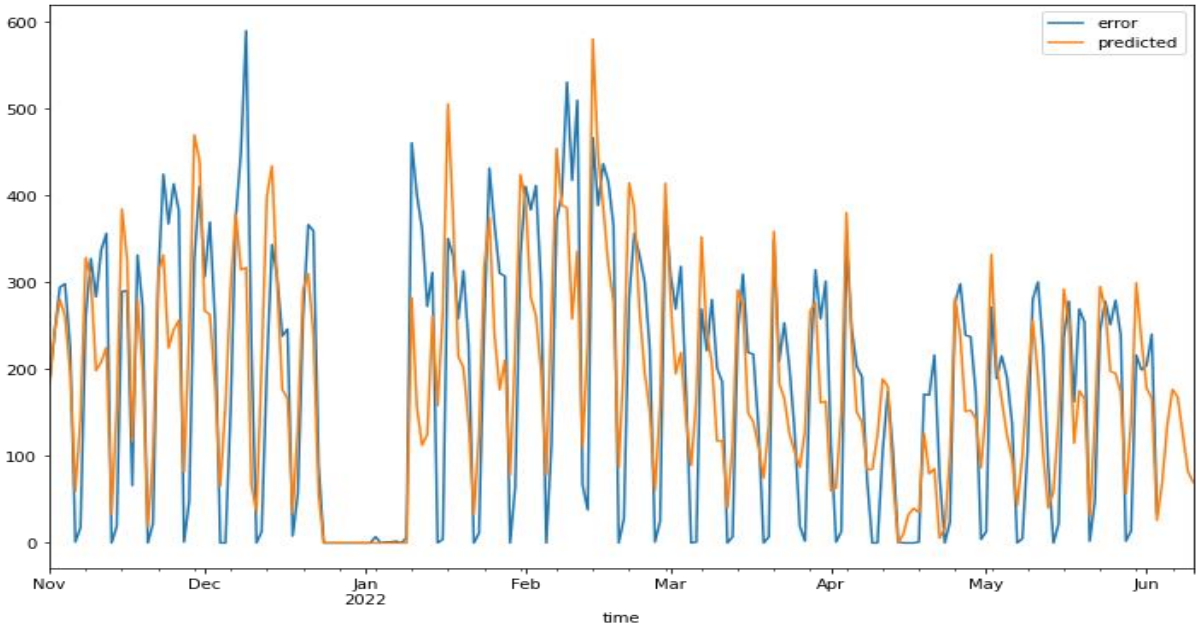
Veškeré hodnoty v tabulce 12 jsou nižší než u modelu ARIMA. Nejvýznamnější je zde hodnota koeficientu determinace, která říká, že tento model byl schopen vysvětlit 77 % variability hodnot testovací množiny. Z hlediska přesnosti modelu se však stále nejedná o velmi kvalitní model, ale vzhledem k charakteru datového souboru a většího množství odlehlých hodnot se dá vyšší míra chybovosti při vysvětlování časové řady předpokládat.

5.4.3 Evaluace modelů

Protože cílem implementovaného statistického modelu bylo získání odhadu očekávaného počtu chyb ve skladu v následujících dnech, byl celý datový soubor použit jako trénovací. Tato praktika umožňuje modelu pokračovat v predikci i mimo rozsah datového souboru. Poskytnutý odhad nelze ověřit na testovacích datech a nelze se tedy spolehnout na nic jiného než na přesnost modelu při vysvětlování celého datového souboru.

Učení modelu ARIMA bylo v jeho parametrech nastaveno na dostupná data ze skladu, tedy období 7 měsíců. Další změnou vůči předešlým modelům bylo rozšíření období pro predikci o 7 dnů více, než je velikost datové sady za účelem predikce hodnoty v budoucím týdnu. Pro těchto 7 dní bylo nezbytné vytvořit časový údaj, který odpovídá dnům, jež následují po posledním datumu dostupného datového souboru.

```
final_model = ARIMA(df,order=(3,1,2))
final_results = final_model.fit()
prediction = pd.DataFrame(final_model.predict(start=1,end=len(df)+7))
```



Obrázek 42: *ARIMA model – predikce výskytu chyb v následujícím týdnu [vlastní zpracování]*

Na obrázku 42 je zobrazen model s predikcí počtu chyb v následujícím týdnu a tabulka 13 obsahuje predikované hodnoty s doplněným datumem. Sloupec „error count“ obsahuje NaN hodnoty, protože pro budoucí týden údaje o počtech chyb nejsou známé. Je tudíž možné porovnat přesnost odhadu modelu až po uplynutí 7 dnů.

time	predicted	error count
01.06.2022	177.975414	204.0
02.06.2022	166.178749	240.0
03.06.2022	25.950396	34.0
04.06.2022	68.627972	NaN
05.06.2022	135.918505	NaN
06.06.2022	176.466364	NaN
07.06.2022	167.670435	NaN
08.06.2022	124.367496	NaN
09.06.2022	82.174632	NaN
10.06.2022	69.546720	NaN

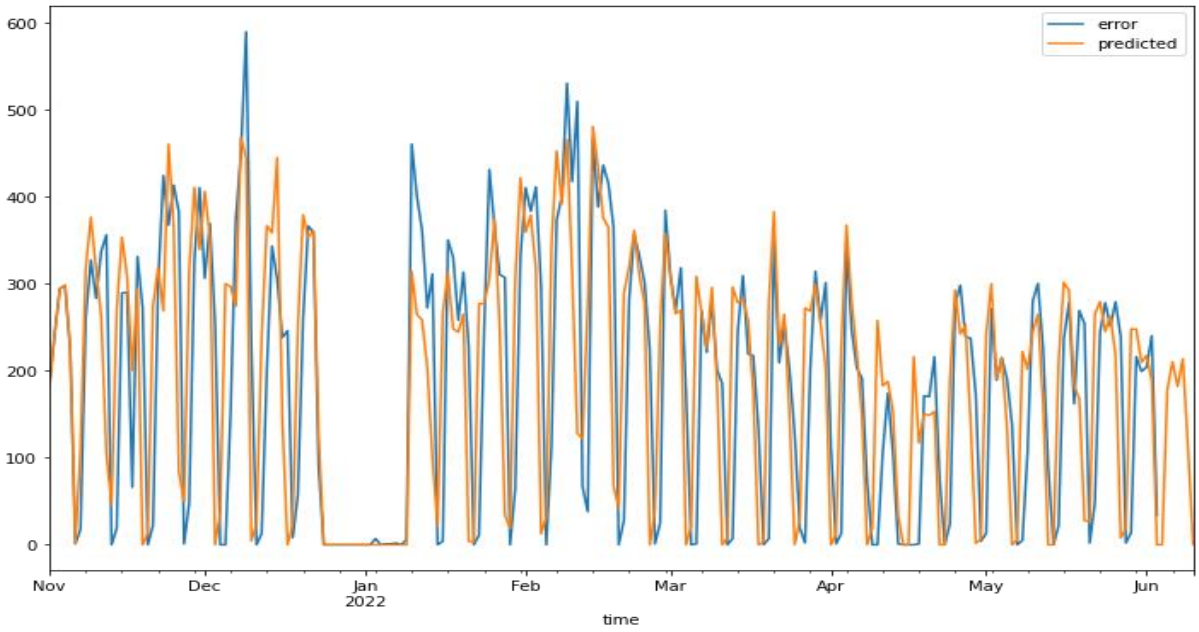
Tabulka 13: *Obsah prediction pandas dataframu*

Pro model SARIMA platí stejná pravidla, jako u předchozího modelu ARIMA, kde byla část predikce rozšířena o 7 dní.

```
final_seasonal_model = sm.tsa.statespace.SARIMAX(df, order=(3,1,2),
seasonal_order=(5,1,3,7))
final_seasonal_results = final_seasonal_model.fit()
```

```
final_seasonal_forecast = final_seasonal_results.predict(start=1, end=len(df)+7)
```

Na obrázku níže je zobrazen graf predikce počtu chyb v následujících 7 dnech vytvořený modelem SARIMA. V případě presence sezónního paramteru v predikované části je vidět vyšší aktivita v průběhu pracovního týdne a nulová aktivita během víkendů. Tabulka 14 obsahuje konkrétní čísla predikce v následujících dnech.



Obrázek 43: *SARIMA model – predikce výskytu chyb v následujícím týdnu [vlastní zpracování]*

time	predicted	error count
01.06.2022	217.247721	204.0
02.06.2022	187.842216	240.0
03.06.2022	0.000000	34.0
04.06.2022	0.000000	NaN
05.06.2022	177.633165	NaN
06.06.2022	210.159877	NaN
07.06.2022	181.868354	NaN
08.06.2022	213.543959	NaN
09.06.2022	111.012458	NaN
10.06.2022	0.000000	NaN

Tabulka 14: *Obsah final_seasonal_forecast pandas dataframu*

Protože při predikci budoucích dat není možné porovnat chybovost (počet predikovaných hodnot a skutečných hodnot se liší), a tudíž ani přesnost odhadu modelu vůči skutečným datům, byl model spuštěn nad dostupným datovým souborem. Takovým způsobem se z dostupných

dat stala testovací množina a bylo tak možné vypočítat metriky a zjistit odchylky odhadu modelu od datového souboru.

Tabulka 15 obsahuje porovnání modelů ARIMA a SARIMA v případě testování na celém datovém souboru, kdy byly porovnány predikované hodnoty se skutečnými hodnotami obou modelů.

	MAX ERROR	MAE	MSE	RMSE	R^2
ARIMA	272.293	73.567	8466.741	92.014	0.625
SARIMA	381.162	82.760	15179.798	123.206	0.329

Tabulka 15: *Porovnání modelů ARIMA a SARIMA*

Z porovnání chybovosti modelů vyplývá, že na dostupné datové množině je model ARIMA výrazně přesnější než SARIMA (ARIMA model disponující sezónním vstupním parametrem). Veškeré ukazatelé chybovosti jsou nižší na straně ARIMA modelu s výjimkou hodnoty R^2, která poskytuje údaj o schopnosti modelu vysvětlit část variability proměnných, a tudíž by měla být co možná nejvyšší. Pro predikci počtu chyb by tedy bylo vhodnější využít ARIMA modelu.

5.4.4 Predikce počtu chyb s ohledem na typ chyby

Pro predikci výskytu konkrétního typu chyby bylo nezbytné upravit vstupní data. Opět bylo využito knihovny Pandas k úpravě a transformaci dat a programovacího jazyku Python pro seřazení a roztřídění dat podle typu chyb a jejich četnosti v datovém souboru.

Nejdříve bylo zapotřebí seřadit záznamy o výskytu typu chyb dle času a následně skombinovat záznamy s identickými typy chyby pomocí groupby a spočítat četnost výskytů. Tabulka 16 popisuje podobu datového souboru po těchto úpravách kdy sloupec „counts“ obsahuje počet výskytu konkrétní chyby v jednom dni.

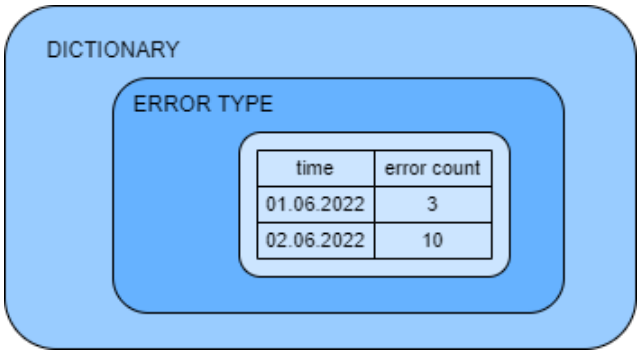
time	error	counts
01.11.2021	6002	45
01.11.2021	6010	3
01.11.2021	6129	30
01.11.2021	6134	2
01.11.2021	6573	3

Tabulka 16: *Datový soubor po úpravě pomocí Pandas*

Vstupními daty pro implementaci model ARIMA by měla být jednoduchá časová řada s informací o času a četnosti. Proto bylo využito seznamů unikátních klíčů a hodnot (dictionaries) a listů, do kterých byla vstupní data rozdělená a uložená pro následnou předpověď

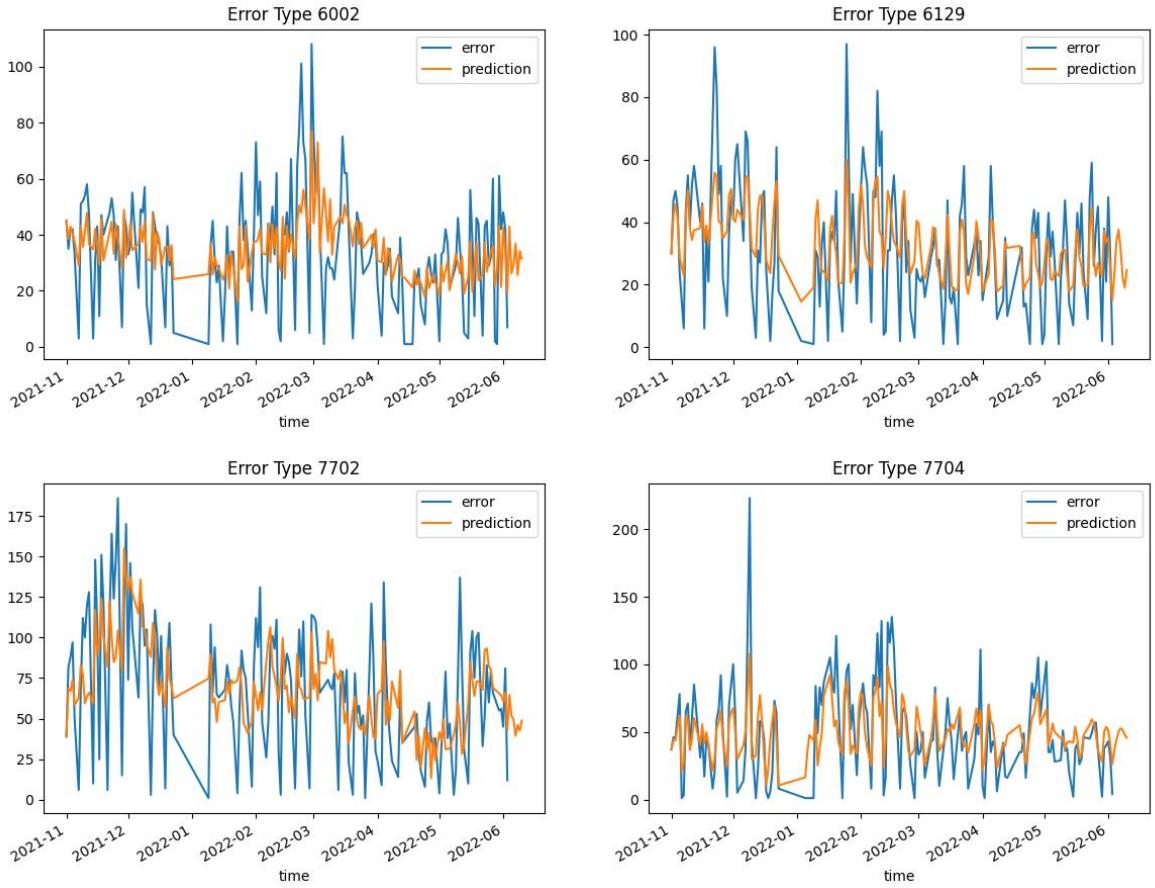
počtu výskytu konkrétního typu chyby. Obrázek 44 popisuje jednoduchou strukturu datového souboru, která byla vytvořená pomocí pythonu.

Nejdříve byla vytvořena dictionary s typem chyby v podobě klíče, a listu (na obrázku „ERROR TYPE“) jako hodnoty. Tento list byl následně naplněn dalšími dictionary, kdy každá dictionary představovala řádek v záznamu tabulky s časem a počtem výskytu konkrétní chyby. Dictionary v listu „ERROR TYPE“ měly za klíč sloupec time a hodnotu počet chyby (error count).



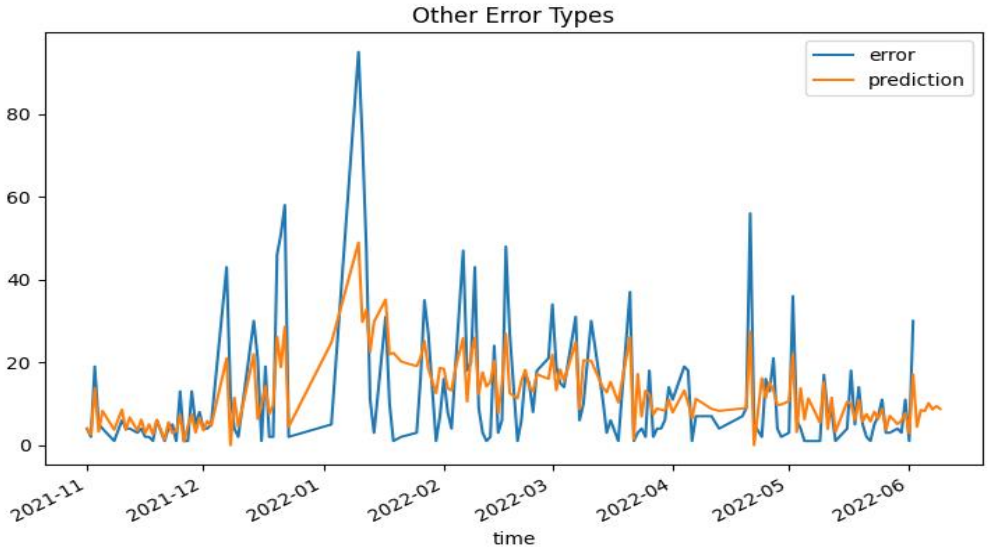
Obrázek 44: *Schéma struktury transformace dat – typ chyby [vlastní zpracování]*

Vytvořeny byly dvě dictionary výše popsané podoby. První obsahovala 7 typů chyb, které se vyskytovaly nejčastěji a v druhé dictionary byly uložené ostatní, méně početné, typy chyb.



Obrázek 45: Predikce počtu top 4 typů chyb [vlastní zpracování]

Obrázek 45 zobrazuje 4 grafy na kterých je možné vidět skutečný počet chyb značený modrou linií a modelem ARIMA odhadovaný počet chyb značený oranžovou linií. Jedná se o 4 nejčastější chyby – 6002, 6129, 7702, 7704. ARIMA odhaduje počet chyb na budoucích 7 dní.



Obrázek 46: Predikce počtu méně častějších typů chyb [vlastní zpracování]

Na obrázku 46 je vidět odhad modelu ARIMA pro typy chyb, jejichž výskyt není tak častý. Model usiluje o předpověď počtu výskytu chyb v následujícím týdnu. Při predikcích jednotlivých typů chyb zvlášť, vychází součet odhadu počtu chyb odlišně od predikcí nezávislých na jednotlivých typech chyb, jež byly představeny v podkapitole 5.4.2. Tento rozdíl pro ARIMA model tvoří 477 chyb, kdy celkový počet chyb predikovaný pro 7 následujících dnů bez ohledu na typ chyby byl 824 a při predikci každého typu chyby zvlášť byl celkový počet roven 1301 chybám. Predikce podle jednotlivých typů chyb zvlášť se zdají být přesnější, protože charakter vstupních dat není tak chaotický, což usnadňuje modelu vytvořit odhad na základě jasnějších vztahů mezi předcházejícími hodnotami.

6 Shrnutí výsledků

Základní analýza a vizualizace dat v první kapitole praktické části poskytuje zajímavý vhled do problematiky a slouží jako vstup pro další, složitější a konkrétnější metody analýzy dat a monitoringu. Představená implementace process miningu dovoluje analyzovat data z pohledu procesů, které se ve skladu odehrávají. Mezi hlavní výhody tohoto řešení patří bezesporu možnost detailní analýzy a přepínání pohledů mezi vyšší a nižší úrovní zobrazení, jednoduchá grafická reprezentace a flexibilita práce s dostupnými daty.

V předešlé kapitole byly představeny dva modely pro predikci časových řad, respektive jeden model, ARIMA a jeho nadstavba SARIMA. V průběhu řešení problému predikce počtu výskytu chyb ve skladu bylo porovnáním výše představených metrik zjištěno, že jednoduchý model ARIMA je na větším trénovacím souboru přesnější než jeho nadstavba s možností zohlednění sezónní složky.

Přestože výsledky představených modelů ARIMA a SARIMA nejsou ideální a odchylky predikovaných hodnot od hodnot skutečných jsou místy vysoké, poskytují poměrně jednoduchý způsob pro vytvoření předpovědi vývoje časové řady. Jak již bylo zmíněno v teoretické části, možností pro predikci časových řad je více, ale jsou složitější a jejich implementace je časově náročnější. Jelikož tato práce není zaměřená pouze na strojové učení a výkonnost jednotlivých statistických modelů, ale na analýzu a techniky využívané při práci s daty, jedná se o adekvátní řešení, které umožňuje jednoduše získat představu o dalším vývoji časové řady a využít tak získaných informací pro podporu při rozhodování.

7 Závěry a doporučení

V diplomové práci byly představeny metody a techniky, které se využívají při analýze velkého objemu dat. Od úprav a transformace dat pomocí nástrojů programovacího jazyka Python, přes vizualizaci a analýzu prostřednictvím Power BI za účelem získání informační hodnoty a využití přístupu process miningu, k pochopení logiky procesů, které se odehrávají ve skladu, až po implementaci strojového učení pro predikci výskytu počtu chyb. Všechny tyto aktivity vedou k lepšímu pochopení dat, které společnosti i jedinci mají k dispozici a jejichž množství neustále roste. Tato práce ukazuje možnost pohledu na data z různých úhlů v závislosti na problému, který je nutné řešit. Získané výstupy je možné využít jako podporu při rozhodování, odhalení anomálií ve skladu, sledování doby zaskladnění a vyskladnění jednotlivých součástek, předcházet možnému výskytu chyb změnou logistických procesů či navýšením personálu ve skladu apod.

Všechny metody použité v diplomové práci byly implementovány na historických datových souborech. Nabízí se tak možnost potenciálního rozšíření a úpravy řešení, které by umožňovalo analýzu a predikci nad daty v reálném čase. Jednalo by se o systém, který by pravidelně získával dostupné logové zprávy ze strojů, upravoval je dle potřeby další analýzy a za pochodu vytvářel dashboardy a prediktivní modely.

8 Seznam použité literatury

[1] A. M. TURING, ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM, 1936.

[2] IDC, „www.idc.com,“ wgfsdf, 24 Březen 2021. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS47560321. [Přístup získán asdead Listopad 2021].

[3] H. Skalská, Data mining a klasifikační modely, GAUDEAMUS, 2010.

[4] O. M. a. L. R. S. Edition, Data Mining and Knowledge Discovery Handbook, Springer, 2010.

[5] A. Abdel, Business Intelligence and Agile Methodologies for Knowledge-Based Organizations: Cross-Disciplinary Applications, Harshey: IGI Global, 2011.

[6] P. Pavel, Data Mining, Pardubice: Univerzita Pardubice, 2010.

[7] D. D. L. Olson a D. D. Delen, Advanced Data Mining, Berlin: Springer-Verlag Berlin Heidelberg, 2008.

[8] KDnuggets, „Data Science PM,“ 2022. [Online]. Available: https://www.datascience-pm.com/crisp-dm-2/#:~:text=Although%20designed%20to%20help%20guide,down%20from%2013%25%20in%202007..

[9] B. M. Ramageri, „DATA MINING TECHNIQUES AND APPLICATIONS,“ sv. 1, č. DATA MINING TECHNIQUES AND APPLICATIONS, 2010.

[10] R. Agrawal a R. Srikant, „Fast algorithms for mining association rules,“ sv. 1215, 1994.

[11] R. Keim, „How to Use a Simple Perceptron Neural Network Example to Classify Data,“ 17 November 2019. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/how-to-perform-classification-using-a-neural-network-a-simple-perceptron-example/.

[12] M. A. Nielsen, Neural Networks and Deep Learning, ACADEMICA, 2015.

[13] R. &. A. R. Adhikari, An Introductory Study on Time series Modeling and Forecasting, LAMBERT Academic Publishing, 2013.

[14] R. Adhikari a R. K. Agrawal, „An Introductory on Time Series Modeling and Forecasting," č. Time Series, 2009.

[15] IBM, „IBM," 22 02 2020. [Online]. Available: http://www.ibm.com.

[16] J. Kosek, Big Data a NoSQL databáze, Grada, 2015.

[17] S. Inc., „Splunk," Splunk, 2022. [Online]. Available: https://www.splunk.com/en_us/about-splunk.html.

[18] G. Qi, W.-T. Tsaiab, W. Li, Z. Zhu a Y. Luod, „A cloud-based triage log analysis and recovery framework," sv. 77, č. A cloud-based triage log analysis and recovery framework, 2017.

[19] Edureka, „Splunk Architecture," 29 July 2021. [Online]. Available: https://www.edureka.co/blog/splunk-architecture/.

[20] S. Documentation, „Splunk Documentation," 07 June 2021. [Online]. Available: https://docs.splunk.com/Documentation/Splunk/8.2.5/Capacity/Forwarder-to-indexerratios.

[21] Splunk, „Splunk Documentation," 17 July 2018. [Online]. Available: https://docs.splunk.com/Documentation/Splunk/8.2.5/Capacity/ComponentsofaSplunkEnterprisedeployment.

[22] S. Documentation, „Splunk Documentation Forwarders," 13 December 2019. [Online]. Available: https://docs.splunk.com/Documentation/Splunk/8.2.5/Capacity/Forwarder-to-indexerratios.

[23] D. Flair, „Power BI Architecture," [Online]. Available: https://data-flair.training/blogs/power-bi-architecture/.

[24] U. S. Mendi, „Power BI Architecture,“ 18 February 2022. [Online]. Available: https://mindmajix.com/power-bi-architecture.

[25] P. Tikait, „Everything You Need to Know about Microsoft Business Intelligence Tools,“ [Online]. Available: https://www.selecthub.com/business-intelligence/microsoft-business-intelligence-tools/.

[26] M. Russo a A. Ferrari, The Definitive Guide to DAX: Business intelligence with Microsoft Excel, SQL Server Analysis Services, and Power BI, Redmond, Washington 98052-6399: Microsoft Press, 2015.

[27] „DAX overview,“ 17 December 2021. [Online]. Available: https://docs.microsoft.com/en-us/dax/dax-overview.

[28] N. M. C. F.-L. Jorge Munoz-Gama, „Process mining for healthcare: Characteristics and challenges,“ sv. 127, 2022.

[29] W. v. d. Aalst, Process Mining in Action, Heidelberg New York Dordrecht London: Springer, 2016.

[30] E. Meijering, „A bird’s-eye view of deep learning in bioimage analysis,“ sv. 18, 2020.

[31] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, Gravenstein Highway North, Sebastopol, CA 95472: O’Reilly Media, Inc., 2019.

[32] A. S. Gupta, „Springboard,“ SpringBoard, 6 October 2021. [Online]. Available: https://www.springboard.com/blog/data-science/best-language-for-machine-learning/.

[33] „What is Python? Executive Summary,“ [Online]. Available: https://www.python.org/doc/essays/blurb/.

[34] Scikit-Learn, „Getting Started,“ [Online]. Available: https://scikit-learn.org/stable/getting_started.html.

[35] Keras, „Keras,“ [Online]. Available: https://keras.io/.

[36] TensorFlow, „TensorFlow,“ [Online]. Available: https://www.tensorflow.org/.

[37] R. Adhikari a R. K. Agrawal, An Introductory Study on Time Series Modeling and Forecasting, LAP Lambert Academic Publishing, 2013.

[38] P. Ing. Roman DANEL, „VSB,“ Listopad 2004. [Online]. Available: http://homel.vsb.cz/~dan11/publikace/Danel_Autoregresni_model_predikce_casovych_rad.pdf.

[39] R. Lewis a G. C. Reinsel, „Prediction of Multivariate Time Series by Autoregressive Model Fitting,“ sv. 16, 1985.

[40] Z. Ivanovski, A. Milenkovski a Z. Narasanov, „TIME SERIES FORECASTING USING A MOVING AVERAGE MODEL FOR EXTRAPOLATION OF NUMBER OF TOURIST,“ sv. 9, 2018.

[41] P. CORTEZ, M. ROCHA a J. NEVES, „Evolving Time Series Forecasting ARMA Models,“ sv. 10, 2014.

[42] I. C. Education, „Recurrent Neural Networks,“ IBM, 14 September 2020. [Online]. Available: https://www.ibm.com/cloud/learn/recurrent-neural-networks.

9 Seznam obrázků

Obrázek 1: Hierarchická struktura metody CRISP-DM, převzato z [4] 6
Obrázek 2: Životní cyklus data miningu, převzato z [6] 7
Obrázek 3: Fáze metodologie SEMMA, převzato z [6] 8
Obrázek 4: Popularita jednotlivých metodologií data miningu, převzato z [7] 9
Obrázek 5: Perceptron, převzato z [10] 12
Obrázek 6: Splunk, fáze zpracování dat, převzato z [17] 19
Obrázek 7: komponenty, převzato z [19] 22
Obrázek 8: Splunk, vlastnosti, převzato z [20] 22
Obrázek 9: Architektura Power BI, převzato z [22] 25
Obrázek 10: Aktivity process miningu, převzato z [27] 28
Obrázek 11: Procesní log, převzato z [27] 29
Obrázek 12: Posloupnost aktivit procesu, převzato z [27] 30
Obrázek 13: Model process miningu, převzato z [27] 30
Obrázek 14: Vědecké publikace související s Machine Learning, převzato z [28] 31
Obrázek 15: Trénování klasifikačního modelu, převzato z [29] 33
Obrázek 16: Regresní mode, převzato z [29] 34
Obrázek 17: Zpětnovazební učení, převzato z [29] 35
Obrázek 18: Model lineární regrese, převzato z [29] 39
Obrázek 19: Funkce logistické regrese, převzato z [29] 39
Obrázek 20: Rozhodovací strom, převzato z [2] 40
Obrázek 21: Rekurentní neuronové sítě vs umělé neuronové sítě, převzato z [38] 44
Obrázek 22: Schéma struktury transformace dat 46
Obrázek 23: Relace mezi třídami REFID_cls a TELEGRAM 47
Obrázek 24: Popisné statistiky Power BI 50
Obrázek 25: Dashboard Power BI dle přepravovaných komponent 51
Obrázek 26: Komponenty a jejich dodavatelé – zobrazení dle času 52
Obrázek 27: Dashboard dodavatelé dle chybovosti a času 52
Obrázek 28: Power BI process mining high case 56
Obrázek 29: Power BI process mining low case 57
Obrázek 30: Procesní mapa veškerých aktivit skladu (vyšší úroveň) 58
Obrázek 31: Procesní mapa veškerých aktivit (nižší úroveň) 58

Obrázek 32: Procesní mapa – Case x Component a Case duration (vyšší úroveň) 59
Obrázek 33: Procesní mapa – Kompletní pohyb KLT přepravky (vyšší úroveň) 60
Obrázek 34: Power BI dashboard nejdelší relace (nižší úroveň) 61
Obrázek 35: Vizualizace časové řady 63
Obrázek 36: Graf ACF pro vstupní data 64
Obrázek 37: Graf PACF pro vstupní data 64
Obrázek 38: ARIMA model – testovací množina 66
Obrázek 39: ARIMA model – trénovací a testovací množina 66
Obrázek 40: SARIMA model – testovací množina 69
Obrázek 41: SARIMA model – trénovací a testovací množina 69
Obrázek 42: ARIMA model – predikce výskytu chyb v následujícím týdnu 71
Obrázek 43: SARIMA model – predikce výskytu chyb v následujícím týdnu 72
Obrázek 44: Schéma struktury transformace dat – typ chyby 74
Obrázek 45: Predikce počtu top 4 typů chyb 75
Obrázek 46: Predikce počtu méně častějších typů chyb 75

10 Seznam tabulek

Tabulka 1: Process Mining Use Cases [vlastní zpracování] 27
Tabulka 2: Získané hodnoty z nestrukturované logové zprávy 46
Tabulka 3: Soubor REFIDtel_export.csv 48
Tabulka 4: Soubor REFID_export.csv 48
Tabulka 5: Procesní log nižší úrovně 54
Tabulka 6: Procesní log vyšší úrovně 55
Tabulka 7: Časová řada (čas a typ chyby) 62
Tabulka 8: Upravená časová řada (počet chyb v 24hodinových intervalech) 63
Tabulka 9: Obsah dataframu prediction 65
Tabulka 10: Metriky modelu ARIMA na testovací množině 68
Tabulka 11: Obsah dataframu seasonal_forecast 68
Tabulka 12: Metriky modelu SARIMA na testovací množině 70
Tabulka 13: Obsah prediction pandas dataframu 71
Tabulka 14: Obsah final_seasonal_forecast pandas dataframu 72
Tabulka 15: Porovnání modelů ARIMA a SARIMA 73
Tabulka 16: Datový soubor po úpravě pomocí Pandas 73

UNIVERZITA HRADEC KRÁLOVÉ | Studijní program: Informační management

Fakulta informatiky a managementu | Forma studia: Prezenční

Akademický rok: 2021/2022 | Specializace/kombinace: Informační management (im2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení: **Bc. Vladislav Poverin**

Osobní číslo: **I2000098**

Adresa: Bavlnářská 554, Semily, 51301 Semily, Česká republika

Téma práce: Analýza aktivit skladu pomocí Process Miningu a využití Machine Learning technologií jejich podporu

Téma práce anglicky: Analysis of warehouse activity using Process Mining followed by Machine Learning optimise the efficiency of storage facility

Vedoucí práce:

Ing. Barbora Tesařová, Ph.D.
Katedra informatiky a kvantitativních metod

Zásady pro vypracování:

Cíl:

Cílem této diplomové práce je představení metod a postupů využívaných k transformaci, vizualizaci a analýze strojových dat, jejich následná aplikace a zapojení machine learningových algoritmů na podporu vybraných procesů.

Osnova:

- Teoretická část

- Představení metod a technik dolování znalostí z dat
- Popis nástrojů pro analýzu a vizualizaci dat
- Uplatnění process mining tástrojů pro analýzu procesů
- Machine learning a využití predikčních modelů

- Praktická část

- Extrakce a transformace dat
- Analýza a vizualizace dat v Power BI
- Implementace procesní mapy a analýza procesů
- Výběr a implementace machine learning modelu pro predikci

Seznam doporučené literatury:

SKALSKÁ, Hana. Data mining a klasifikační modely. Hradec Králové: Gaudeamus, 2010. Recenzované monografie. ISBN 9788074350887.

Petr, Pavel. Data Mining.. Pardubice: Univerzita Pardubice, 2006. ISBN 80-7194-886-1.

GÉRON, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems. Second edition.

Beijing: O'Reilly, 2019. ISBN 978-1-492-03264-9.

OLSON, Dr. David L. a Dr. Dursun DELEN. Advanced Data Mining Techniques. 1. Verlag Berlin Heidelberg: Springer, 2008. ISBN 978-3-540-76916-3.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum: