Využití Python knihoven pro analýzu časových řad

Šimon Slovák

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Šimon Slovák**
Osobní číslo:	**A20018**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Využití Python knihoven pro analýzu časových řad**
Téma práce anglicky:	**The Usability of Python Libraries for Time Series Analysis**

Zásady pro vypracování

1. Zpracujte literární rešerši na téma využití metod dataminingu pro analýzu časových řad.
2. Vytvořte přehled typových úloh.
3. Představte možnosti Python knihoven zaměřených na datovou analýzu.
4. Vytvořte praktické ukázky pro typové úlohy v prostředí Jupyter notebook.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. PALMA, Wilfredo. Time Series Analyses. Hoboken, New Jersey: Wiley, [2016]. ISBN 978-1118634322.
2. MILLS, Terence C. Applied time series analysis: a practical guide to modeling and forecasting. London: Academic Press, [2019]. ISBN 978-0128131176.
3. MONTGOMERY, Douglas C., Cheryl L. JENNINGS a Murat KULAHCI. Introduction to time series analysis and forecasting. Second edition. Hoboken, New Jersey: Wiley, [2015]. ISBN 978-1118745113.
4. AGGARWAL, Charu C. Data Mining: The Textbook. 2015th Edition. Imprint: Springer, 2015. ISBN 978-3319141411.
5. ZELLE, John M. Python programming: an introduction to computer science. Third edition. Wilsonville: Franklin, Beedle & Associates, [2017], xv, 536 s. ISBN 978-1-59028-275-5.
6. WITTEN, I. H. a I. H. WITTEN. Data mining: practical machine learning tools and techniques. 4th Edition. Amsterdam: Elsevier, [2017]. ISBN 978-0128042915.

Vedoucí bakalářské práce: **Ing. Adam Viktorin, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 25. 5. 2023 …….

podpis studenta

ABSTRAKT

Abstrakt česky

Bakalářská práce cílí na využití Python knihoven pro analýzu časových řad. Součástí práce je literární rešerše, která se zabývá metodami dataminingu pro analýzu časových řad. Práce se dále zabývá vytvořením přehledu typových úloh, které lze řešit skrze analýzu časových řad. V práci jsou také představeny běžné i specifické Python knihovny, které lze využít pro datovou analýzu. V rámci této práce jsou vytvořeny i ukázky typových úloh, které jsou ve formě několika Jupyter notebooků.

Klíčová slova: časové řady, datamining, datová analýza, Python, Jupyter notebook

ABSTRACT

The bachelor's thesis aims to utilize Python libraries for time series analysis. The thesis includes a literature review that focuses on data mining methods for time series analysis. Furthermore, it provides an overview of typical tasks that can be solved through time series analysis. The thesis also introduces common and specific Python libraries that can be utilized for data analysis. As part of this work, practical examples of typical tasks are created and presented in the form of Jupyter notebooks.

Keywords: time series, datamining, data analysis, Python, Jupyter notebook

Chtěl bych poděkovat zejména svému vedoucímu bakalářské práce Ing. Adamu Viktorinovi, Ph.D., že měl se mnou svatou trpělivost a byl mi k dispozici i v posledních chvílích před odevzdáním bakalářské práce.

Dále bych chtěl poděkovat své rodině za to, že mi byla psychickou podporou v průběhu celé doby studia.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Tématem bakalářské práce je vytvoření přehledu o aktuálním využití dataminingových metod v analýze časových řad.

Zejména v dnešní době se setkáváme s množstvím různých zařízení, které v reálném čase produkují obrovské množství dat. Tyto data je třeba dále zpracovávat a analyzovat, aby se docílilo získání cenných znalostí, na kterých lze postavit budoucí rozhodnutí.

V teoretické části jsou nastíněny typy sektorů, ve kterých se můžeme potkat s praktickým využitím časových řad. Dále jsou zde popsány rozdělení časových řad, jejich vlastnosti a jednotlivé komponenty. Následně je práce v rámci další kapitoly zaměřená na vysvětlení pojmu data mining. Součástí této kapitoly je i popis různorodých zdrojů dat, metodologií a také samotných fází dolování dat pro konkrétní metodologii. Hlavní teoretickou část tvoří komplexní přehled typových úloh. Typové úlohy jsou zde členěny do podkapitol, které popisují jejich druhy a důležité součásti. Pro typové úlohy jsou zde povrchově představeny i příklady algoritmů, které lze pro tyto typové úlohy využít. Teoretická část je zakončena přehledem základních ale i specifických Python knihoven, které lze využít pro datovou analýzu v časových řadách. Tyto knihovny také zahrnují již implementované základní řešení, které jsou v této práci využity.

V rámci praktické části je zde vytvořeno několik praktických ukázek typových úloh. Na samotném začátku jsou uvedeny datové sady z různých sektorů, aby bylo možné vidět reálné využití analýzy časových řad. Typové úlohy jsou realizovány skrze Jupyter notebooky, ve kterých je demonstrováno využití různých přístupů pro řešení dílčích typových úloh. Praktické ukázky poslouží jako podpůrné studijní materiály při výuce předmětu Datová analýza a inteligentní systémy.

I. TEORETICKÁ ČÁST

1 ČASOVÉ ŘADY

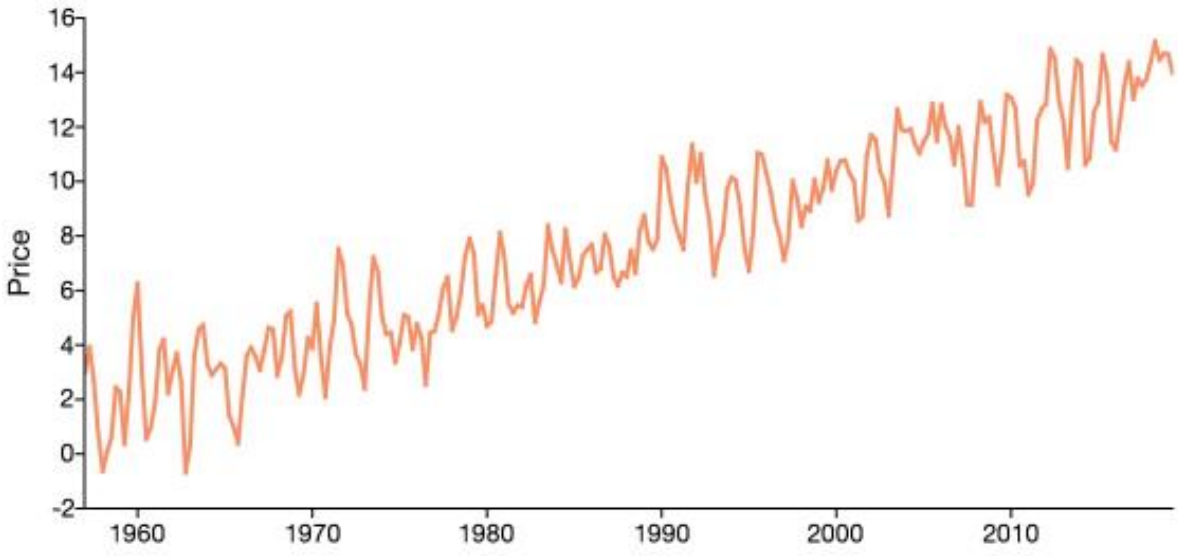
Pojem časová řada lze chápat jako kolekci dat určité náhodné veličiny *y*, která byly nasbírána za určitý časový interval *t* a jsou chronologicky uspořádány. Časový interval může zahrnovat malé jednotky jako milisekundy, sekundy, ale i ty, které jsou v řádech týdnů, měsíců nebo i roků. Tyto hodnoty mohou být jak kladné, tak i záporné, tudíž spadají pod celá čísla, které můžeme matematicky zapsat jako:

$$t \in \mathbb{Z}, \text{ kde } \mathbb{Z} = \{\ldots -3, -2, -1, 0, 1, 2, 3, \ldots\} \quad (1)$$

V závěru tedy můžeme časovou řadu jednoduše matematicky zapsat jako:

$$y(t) = \{y(t_1), y(t_2), y(t_3), \ldots, y(t_n)\} \quad (2)$$

nicméně v praxi vždy využíváme pouze časové řady s konečným počtem prvků. Příklad časové řady je znázorněn na obrázku č. 1. [1]



Obrázek č.1 – Příklad časové řady [2]

1.1 Oblasti výskytu

Se zaznamenáváním a zpracováváním dat se často můžeme setkat v mnoha oblastech, které jsou do značné míry ovlivněny právě zpracováním těchto dat, ze kterých lze získat potřebné znalosti. Příklady různých oblastí využívající data jsou uvedeny v následujících podkapitolách. [1]

1.1.1 Finančnictví

Finanční sféra je typickým příkladem oblasti, kde se časové řady využívají ve velké míře. Téměř každá bankovní aplikace v dnešní době sbírá informace o klientech, které jsou dále zpracovávány. Bez další analýzy dat by dnes jen stěží mohly fungovat finanční instituce nebo samotné banky. Znalosti získané analýzou dat z této sféry často slouží k specifickým účelům jako jsou zlepšení služeb pro klienty, maximalizace zisku, předpovídání chování trhu nebo měnového rizika a spoustu dalších využití. [3] [4]

1.1.2 Průmysl

V průmyslových podnicích dochází v posledních letech k silné digitalizaci a tím i nárustu obrovského množství dat. S termínem digitalizace do značné míry souvisí i pojem internet věcí (angl. Internet of Things, dále jen IoT). V rámci IoT se v různých oblastech začaly objevovat množství senzorů, které v průběhu času generují enormní množství dat. Typicky by takové senzory mohly měřit třeba teplotu, tlak, vlhkost, pH, rychlost nebo jiné veličiny spojené s průmyslovou sférou. Ještě do nedávna bylo možné skrze data časových řad pouze monitorovat chování různých zařízení. Díky datové analýze můžeme z těchto dat získat cenné informace, jako jsou vzory a vztahy mezi daty, které mohou mít zásadní dopad na zefektivnění průmyslových procesů jako je např. optimalizace výroby a následné snížení nákladů. [5]

1.1.3 Zdravotnictví

Místa jako jsou nemocnice nebo kliniky jsou zdrojem četných informací, které by mohly do budoucna transformovat a posunout oblast zdravotnictví. Data, které jsou zaznamenávána o pacientech, mohou zahrnovat měření tepové frekvence, záznamy elektrické aktivity mozku nebo monitorování tlaku pacienta. Správnou analýzou dat časových řad lze v reálném čase zjistit určité vzory, které mohou označovat např. infarkt myokardu, a tím uvědomit včas doktora. Dalším využitím je pak možnost z těchto dat podle určitých kritérií klasifikovat pacienta jako zdravého nebo nemocného. [3] [6]

1.1.4 Energetika

Vyvážená výroba a spotřeba jakékoliv energie má v dnešní době zásadní vliv na životní prostředí a cenu nákladů spojených s udržováním budov. Mezi ty největší výzvy už delší dobu patří snížení emisí a cen energií. K vyřešení těchto problémů by mohla přispět hlubší analýza

energetických dat a získání znalostí, které můžou potenciálně pomoci se správnou optimalizací produkce energie nebo umožnit předpovědět spotřebu pro určité období. [7] [8]

1.1.5 Meteorologie

Meteorologové pro svou práci často potřebují velké množství dat. Tyto data lze získat skrze speciální přístroje, jakými jsou teploměry na měření venkovní teploty, barometry pro zjištění atmosférického tlaku, vlhkoměry na zaznamenávání vlhkosti, srážkoměry pro určení hladiny napadených srážek, anemometry pro měření rychlosti větru nebo dokonce vesmírné satelity. Z takových dat se poté mohou provádět analýzy a následné předpovědi, jestli bude v následujících dnech pěkné nebo špatné počasí. Vliv předpovědí se však vztahuje i na ochranu našich produktů a majetku. V mnoha případech pak můžou tyto predikce pomoci varovat řidiče kvůli namrzlým cestám nebo chránit obyvatelstvo proti živelným pohromám, které zahrnují rozsáhlé lesní požáry, povodně, tsunami, tornáda nebo hurikány. [9] [10]

1.2 Rozdělení

Datová sada často obsahuje jednu nebo více vlastností pro jeden nebo více objektů v jakémkoliv časovém bodě. V závislosti na počtu vlastností a vztahy mezi nimi je možné rozdělit časové řady na:

- **Jednorozměrné** – jedná se čistě o sledování jedné proměnné nebo vlastnosti daného objektu v průběhu času. Příkladem by mohlo být sledování údajů o počasí. V rámci takového příkladu lze pozorovat třeba teplotu, tlak nebo vlhkost. Jednorozměrná časová řada vznikne naměřením čistě jedné z těchto veličin. Na základě této zaznamenané veličiny, jakou může být právě teplota, je možné očekávat, kdy může v průběhu dne nastat oteplení nebo ochlazení. [11]
- **Vícerozměrné** – tento typ dat se týká sledování více než jedné proměnné nebo vlastnosti pro jeden konkrétní objekt v čase. Pro porovnání předchozího typu poslouží stejný příklad na sledování počasí. Kupříkladu dvourozměrnou časovou řadou by mohlo být sledování právě dvou veličin, jakými jsou teplota a tlak, které jsou spolu často spojovány a vzájemně se ovlivňují. [11]
- **Vícenásobné** – jako poslední jsou zde typy dat, které lze chápat jako měření jedné až několika proměnných nebo vlastností pro několik vzájemně nezávislých objektů za dané časové období. Za příklad vícenásobné, ale také i vícerozměrné časové řady,

lze považovat tu, při které dochází k pozorování údajů o počasí v několika vzdálených oblastech, které se liší svou lokalitou. [11]

V závislosti na čase, ve kterém byly pozorovány, se časové řady mohou dále dělit na:

- **Diskrétní** – jako diskrétní lze označit časovou řadu, jejíž jednotlivá data obsahují konkrétní počet hodnot v rámci určitého časového intervalu. Jinak řečeno jsou hodnoty těchto dat svým počtem v rámci intervalu konečné, jednoduše počitatelné a dále nedělitelné. Jako příklad lze uvést třeba počty prodaných kusů výrobků, počet dopravních nehod nebo věk osob. [12]
- **Spojité** – tento typ je charakteristický tím, že data nemají konečný počet hodnot v rámci časového intervalu. V porovnání s diskrétními časovými řadami tak vždy obsahují daleko více hodnot, které jsou dále dělitelné. Tyto hodnoty nutně nemusí být vždy celočíselné. Často se mohou vyskytovat ve formě desetinných čísel nebo procent. Z přechozího vyplývá, že je tento typ díky absenci konečnosti počtu hodnot velmi přesný. Vhodnou ukázkou jsou např. měření teploty nebo tlaku. [12]

1.3 Stacionarita

Během práce s daty časových řad je často možné se potkat se situací, kdy časová řada vykazuje specifický jev, který se nazývá stacionarita. Podle tohoto jevu dělíme časové řady na stacionární a nestacionární. Stacionarita je stav, který sděluje, že se statistické vlastnosti časové řady nemění v závislosti na čase, tj. jsou konstantní v jakémkoliv intervalu časové řady. Mezi tyto vlastnosti patří průměr, rozptyl nebo kovariance. Jakákoliv časová řada, která nesplňuje tyto podmínky, se označuje jako nestacionární. Mezi časová řada, které jsou nestacionární, můžeme zařadit speciální druh časové řady zvaný bílý šum nebo časovou řadu s cyklickým chováním, které je podrobněji vysvětleno v další podkapitole o komponentách časových řad. V případě, že časová řada obsahuje sezonnost nebo trend, není možné tuto časovou řadu označit jako stacionární. [13] [14]

1.3.1 Druhy stacionarit

U časových se lze setkat hned s několika druhy stacionarit, které je možné rozdělit na:

- **Striktní stacionarita** – často označována také jako silná, je druh stacionarity, kdy časová řada vykazuje všechny nebo většinu statistických vlastností konstantní nezávisle při jakémkoliv posunu v čase.

- **Stacionarita prvního řádu** – tato varianta zahrnuje konstantní pouze průměr, ale ostatní vlastnosti se mohou v čase měnit.
- **Stacionarita druhého řádu** – častěji označována jako slabá stacionarita, je typ stacionarity, který je velmi podobný stacionaritě prvního řádu, ale zahrnuje kromě konstantního průměru i konstantní rozptyl a kovarianci. Ostatní vlastnosti se poté mění v čase podobně, jako u předchozího typu stacionarity.
- **Trendová stacionarita** – odstranění trendové složky u nestacionární řady, zanechá časovou řadu stacionární. Taková stacionarita se poté označuje jako trendová.
- **Sezonní stacionarita** – podobně jako u předchozí varianty tato stacionarita vzniká v případě, když dojde k odstranění sezonní složky.
- **Diferenční stacionarita** – jedná se o typ stacionarity, která vyžaduje často i několikanásobnou aplikaci procesu, který se nazývá diferencování.

Techniky pro stacionarizaci jsou představeny v dalších podkapitolách. [13] [14] [15]

1.3.2 Způsoby detekce stacionarity

Existuje hned několik způsobů nebo metod, jak by bylo možné zjistit stacionaritu. Mezi některé ze způsobů, jak stacionaritu detekovat, je možné zařadit:

- **Vizualizace** – jedním z nejjednodušších způsobů, jak zjistit přítomnost stacionarity, je grafické vykreslení dat nebo funkcí s jejichž pomocí je možné lépe rozeznat prvky stacionarity pro jednotlivé dílčí celky. Detekovat stacionaritu umožňují i autokorelační funkce a jejich vykreslení. Jako stacionární můžeme označit tu časovou řadu, u které mají hodnoty tendenci rychle se blížit nule. V opačném případě se jedná o nestacionární časovou řadu.
- **Statistické srovnání –** další z možností je rozdělit si časovou řadu na několik úseků, kdy se u každého úseku porovnávají statistické vlastnosti. Pokud jsou tyto vlastnosti jednotlivých úseků blízko sebe, tak je možné označit časovou řadu jako stacionární.
- **Statistické testy –** zde se jedná např. o testy jako je Dickey-Fullerův test, který detekuje přítomnost stacionarity pomocí p-hodnoty, která k označení časové řady jako stacionární musí být menší nebo rovna 0,05. Jako nestacionární je poté možné označit ty časové řady, které tuto podmínku nesplňují.

[14] [16]

1.3.3 Stacionarizace

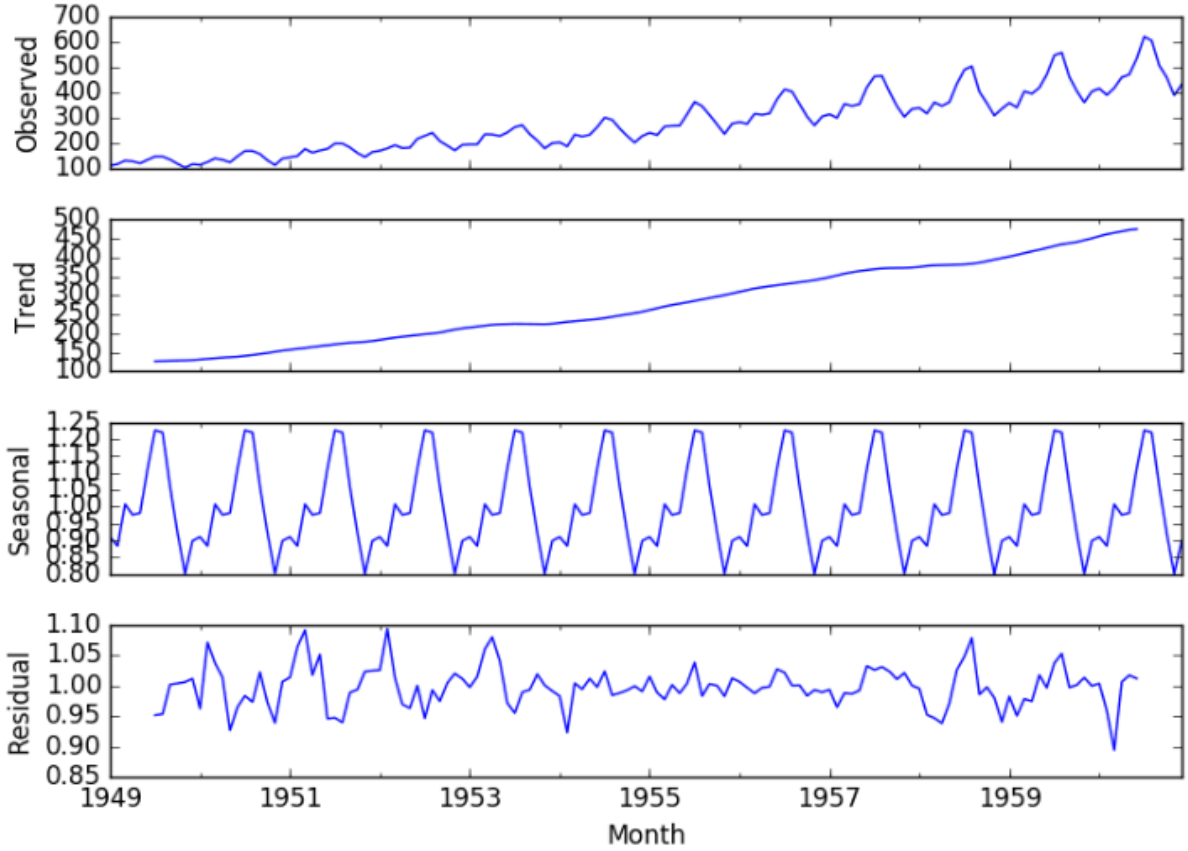
Tímto pojmem se rozumí převod nestacionární časové řady na stacionární. Mezi nejznámější způsoby, jak tento proces uskutečnit, patří:

- **Transformace** – tento způsob slouží k ustálení nekonstantního rozptylu v čase. Existuje hned několik typů, mezi které lze zařadit např. logaritmickou nebo kvadratickou transformaci. Účinnost jednotlivých transformací je třeba vyzkoušet, protože neexistuje transformace, která by byla univerzální.
- **Diferenciování** – jedná se o další způsob převodu, kterým je možné ustálit průměr v čase. Docílit toho lze tím, že se od každé hodnoty odečte hodnota, která je opožděna o jednu periodu. Tento princip lze aplikovat např. na dny v týdnu, kdy je možné toto diferenciování označit jako sezonní.

[17] [18]

1.4 Komponenty

U jednodušších časových řad je možné poměrně snadno vidět na první pohled různé vzory nebo alespoň částečně pochopit jejich chování. V oblasti zpracování velkých dat je ale možné narazit i na složitější časové řady, které je třeba rozložit na dílčí celky, aby zde byla možnost hlubší analýzy a následného pochopení jejich mechanismu. Samotnou časovou řadu lze rozložit do čtyř základních částí. Jedná se o trendovou složku, cyklickou složku, sezonní složku a o zbývající složku náhodnou, často označovanou také jako reziduální nebo stochastickou. Počet složek se nicméně může lišit v závislosti na literatuře nebo kontextu. U rozkladu časové řady je mnohdy uváděno, že trendová složka zároveň obsahuje i složku cyklickou a spolu tak představují cyklický trend. Tento případ je znázorněný na obrázku č. 2. [19] [20]



Obrázek č. 2 – Jednotlivé složky časové řady [21]

Proces rozkladu časové řady se nazývá **dekompozice**, kterou lze dále rozdělit na:

- **Aditivní** – u aditivní dekompozice se předpokládá, že funkce časové řady je součtem svých složek. Z logiky sčítání vychází, že sezonní složka S_t, trendová složka T_t a náhodná složka R_t musí mít tedy stejné jednotky, aby byla tato operace proveditelná. Funkci y_t pro dekompozici můžeme tedy vyjádřit jako:

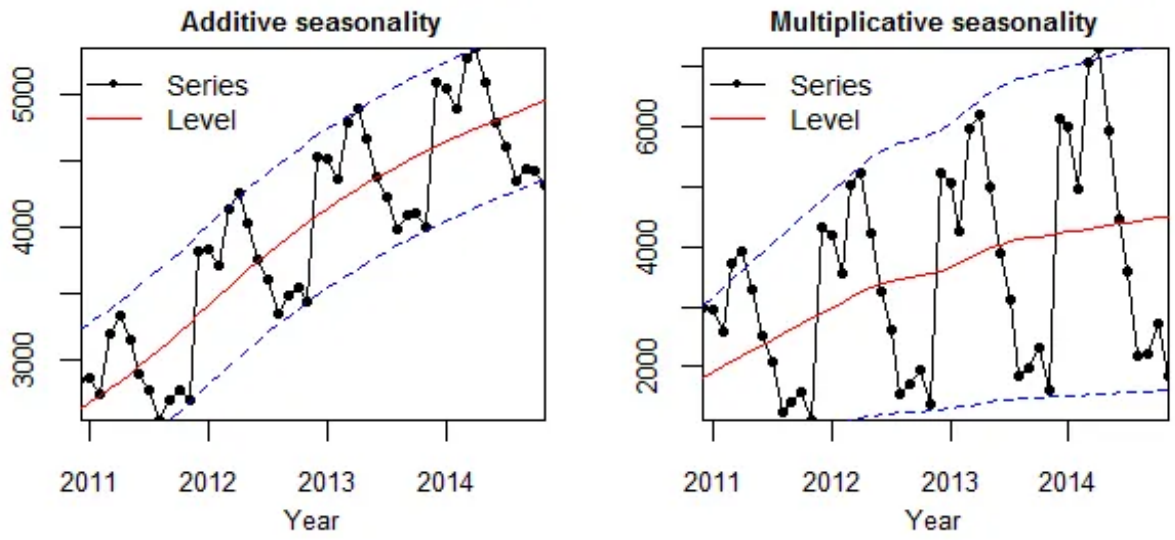
$$y_t = S_t + T_t + R_t \tag{3}$$

  Aditivní dekompozice je vhodná zejména tehdy, kdy se časová řada v rámci svého intervalu nerozšiřuje. V opačném případě je lepší využít multiplikativní dekompozice, která je pro tuto situaci vhodnější.

- **Multiplikativní** – tento typ dekompozice je znázorněn funkcí časové řady, jejíž celkový výsledek je roven násobku jednotlivých složek. Výsledná jednotka násobení je určena trendovou složkou. Zbylé složky slouží pouze k násobení té trendové a určují tak relativní změnu v součinu. Zapisuje se jako:

$$y_t = S_t \times T_t \times R_t \tag{4}$$

Jak bylo zmíněno výše, multiplikativní dekompozice je vhodná spíše v případě, kdy se časová řada v průběhu časového intervalu rozšiřuje, tj. se zvětšuje její rozptyl. V praxi je však možné se potkat i s hybridní variantou, která může obsahovat složky aditivní v kombinaci s těmi multiplikativními. Obrázek č. 3 znázorňuje rozšiřování časové řady v čase pro oba typy dekompozice.



Obrázek č. 3 – Porovnání aditivní a multiplikativní dekompozice [20]

V následujících podkapitolách jsou detailněji rozebrány jednotlivé složky časové řady. [20]

1.4.1 Trendová složka

Trendová složka udává obecný směr časové řady, který je pozorovatelný z dlouhodobějšího hlediska. Trend sám o sobě může být buď lineární nebo nelineární. Dále lze rozdělit podle směru stoupání na vzestupný nebo sestupný. Je třeba zmínit, že jsou i případy, kdy časová řada nevykazuje ani jeden z předchozích trendů, ale

udržuje trend konstantní neboli horizontální. Příkladem jasně vzestupného trendu může být vzrůst ceny Bitcoinu v předcházejících letech. [22] [23]

1.4.2 Sezonní složka

Sezonní složka označuje pravidelné krátkodobé období vzrůstu a poklesu kolem trendu s fixní frekvencí, kdy je známo, že je tento průběh běžný. Takový časový interval může být v jednotkách dnů, týdnů, případně i měsíců. Běžným příkladem časové řady vykazující takové sezonní chování může být zvýšení teploty v letních měsících a následné snížení v době měsíců zimních, zvýšený prodej alkoholu před svátky nebo zvýšená návštěvnost exotických destinací. [22] [23]

1.4.3 Cyklická složka

Stejně jako sezonní složka vykazuje i složka cyklická pravidelné, ale často i nepravidelné vzrůsty a poklesy kolem trendu. Cyklická složka bývá zasazena do dlouhodobějšího období s variabilní frekvencí, které bývá většinou těžko předvídatelné z důvodu jeho délky a kombinace různých událostí, které se v průběhu vyskytnou. Příčiny těchto událostí bývají často ekonomického charakteru. V samotné ekonomice bývá cyklus také velmi často spojován s termínem hospodářského cyklu. Cyklus je možné rozdělit do čtyř etap, kdy se střídá období prosperity, recese, úpadku a obnovy. Časový interval cyklu se může pohybovat i v řádech několika let. Příkladem časové řady vykazující cyklické chování může být třeba sledování prodeje rodinných domů. [22] [23]

1.4.4 Náhodná složka

Náhodná složka zůstává jako poslední část po očištění časové řady od složky trendové, sezonní a cyklické. Jedná se většinou o tzv. bílý šum, který má téměř nulovou střední hodnotu, konstantní rozptyl a vzájemně nekorelované po sobě jdoucí prvky do takové míry, že jsou následné posloupnosti téměř nepredikovatelné. Analýza této čistě náhodné složky však může mít zásadní vliv na další operace s časovými řadami. Typicky se může jednat o výpadky senzorů ve výrobě, přírodní katastrofy typu erupce sopky nebo jinou neméně významnou událost, která by do budoucna mohla ovlivnit třeba výsledek predikce. [22] [23] [24]

2 DOLOVÁNÍ DAT

Dolování dat neboli data mining (dále jen DM) je analytická metoda pro analyzování velkého množství surových dat. Hlavním účelem DM je především identifikování různých druhů vzorů nebo získání užitečných informací, které můžou mít zásadní vliv na řešení spousty různorodých problémů. Dále je díky DM možné snižovat rizika, optimalizovat procesy, zjistit příležitosti na trhu a tím i ovlivnit konkurenceschopnost firmy. Samotný název DM vznikl na základě podobnosti s termínem procesu těžení nerostných surovin. Samotná hornina zde reprezentuje surová data, která musí projít procesem zpracování, aby se docílilo získání pomyslných, na první pohled neviditelných, cenných kovů, které jsou v tomto případě znázorněny jako skryté znalosti. [25] [26]

2.1 Zdroje a struktura dat

Data často pochází z různých systémů a díky tomu se tyto data odlišují svou strukturou, ve které se k nám dostávají. Podle stupně, jak moc jsou data strukturálně organizovaná, je možné rozdělit data na strukturovaná, semi-strukturovaná a nestrukturovaná. Více informací o těchto strukturách dat i s jejich specifickými příklady je rozebráno v následujících podkapitolách. Ukázka všech typů tří typů dat je znázorněna na obrázku č. 4. [27]

Obrázek č. 4 – Znázornění struktury dat [28]

2.1.1 Strukturovaná data

Tato varianta dat je charakteristická svou vysokou mírou organizace. Data se nejčastěji objevují v tabulkové formě, která se skládá z řádků obsahujících záznamy a sloupců označujících vlastnosti daných záznamů. I přes fakt, že tyto data zastupují jen přibližně 20 % všech dostupných dat, tak jsou často preferována, protože jsou díky své struktuře snadno dostupná. Typickým příkladem jsou data zaznamená v SQL databázi,

Excelových souborech nebo velmi často i souborech obsahující hodnoty oddělené čárkami ve formátu CSV. Ačkoliv se formát CSV často jeví jako nepřehledný, je možné ho převést přímo do tabulkové podoby, která je pro další zpracování čitelnější. [27] [29]

2.1.1 Semi-strukturovaná data

Jako semi-strukturovaná data lze označit ty, které vykazují alespoň částečnou míru organizovanosti. Mnohdy se jedná o textové soubory, které obsahují např. tagy, díky kterým je struktura těchto dat méně chaotická pro další zpracování. S dodatečnými úpravami lze tyto data předělat i do tabulkové formy. Tento proces však bývá poměrně pracný. Jedná se zejména o soubory typu HTML, XML nebo JSON. [27] [29]

2.1.1 Nestrukturovaná data

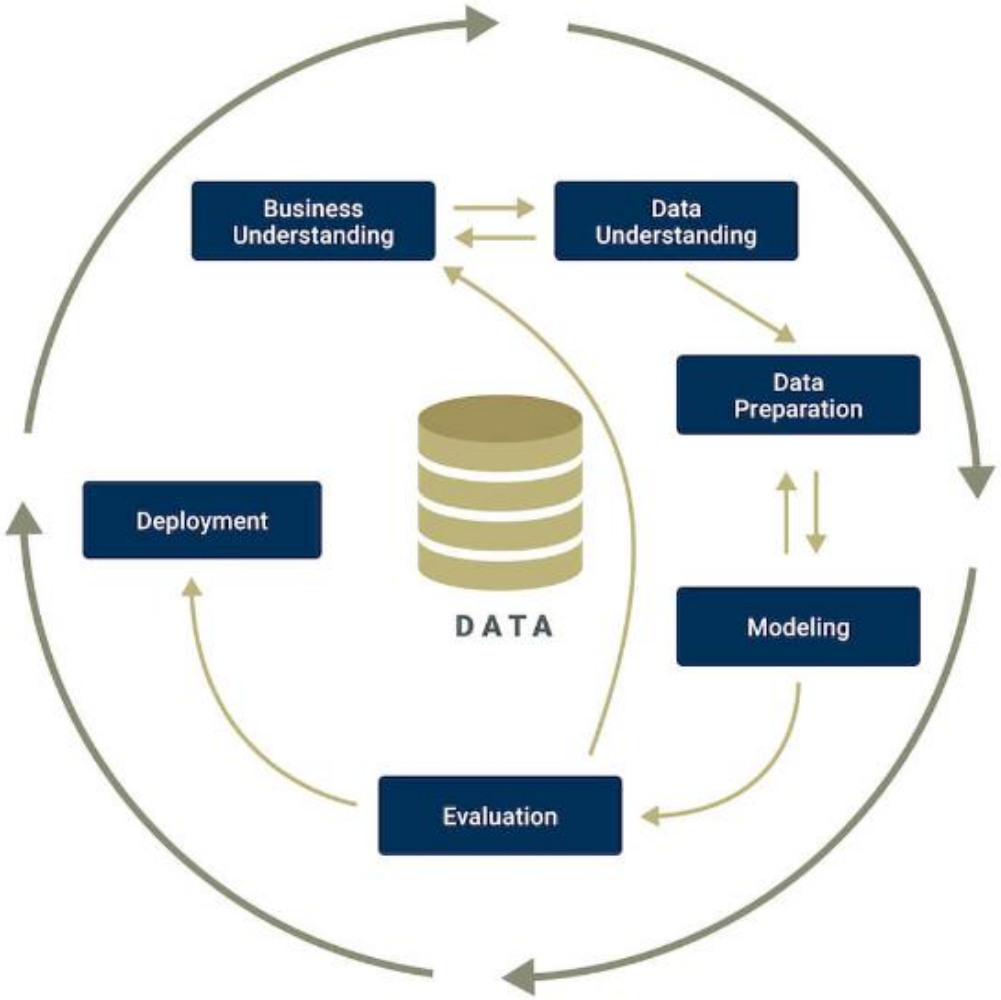
Nestrukturované data se vyskytují v předem nedefinovaných formách a jejich zastoupení v rámci všech dat zabírá největší podíl. Z lidského pohledu jsou tyto data poměrně snadno analyzovatelné, nicméně v rámci počítačového zpracování se tato situace značně komplikuje. Jedná se zejména o textové, zvukové a obrazové soubory, které je třeba dále zpracovat vhodnými metodami strojového učení, aby zde byla možnost dalšího počítačového zpracování a získání potřebných znalostí. [27] [29]

2.2 Metodologie

Na začátku každého datového projektu může být často velmi obtížné odhadnout jakým způsobem se data budou zpracovávat a jaké bude pořadí jednotlivých kroků zpracování dat. Podobně jako u vývoje softwaru je tedy i při práci s daty důležité dodržet určitý postup, který často vede ke zvýšení šancí na správné a efektivní zpracování dat. Tyto metodologie do datových projektů přináší jasnější představu o průběhu zpracování, vhodnou strategii se stanovenými cíli a celkově i lepší týmovou spolupráci, protože každý člen týmu je v průběhu více zasvěcen do jednotlivých kroků zpracování dat. V samotném vývoji softwaru se často objevuje metodologie Scrum. V rámci datových projektů se však tato metodologie kombinuje s různými DM metodologiemi, mezi které lze zařadit KDD (Knowledge Discovery in Databases), SEMMA (Sample, Explore, Modify, Model, Assess) a snad nejvíce používanou CRISP-DM (CRoss-Industry Standard Process for Data Mining), která je podrobněji vysvětlena v podkapitole o fázích DM. [30]

2.3 Fáze

Metodologie typu KDD a SEMMA jsou čistě iterativní a zaměřují se spíše na samotné procesy DM. V porovnání s CRISP-DM obsahují tyto metodologie shodný počet kroků s podobným cílem, zatímco CRISP-DM, která je také iterativní, obsahuje ještě další kroky navíc. Jedná se o fáze porozumění oblasti problému a samotné zprovoznění. Tyto dodatečné kroky pozitivně ovlivňují průběh dalšího zpracování a tím i celkovou rychlost implementace DM v reálných úlohách. Metodologie typu CRISP-DM je zobrazena na obrázku č. 5. [30]



Obrázek č. 5 – Jednotlivé fáze CRISP-DM metodologie [31]

2.3.1 Porozumění problémové doméně

Jedná se o první fází, kdy je třeba zodpovědět si několik otázek, které budou mít za následek stanovení cílů a určení základní strategie pro jejich dosažení. Nezodpovězení základních otázek cílů projektu a neúplné pochopení oblasti problému může poté vést ke špatnému postupu samotného zpracování a následně i celkově špatnému výstupu. Hlavními body tedy jsou:

- Porozumění klientovi z jeho úhlu pohledu, tj. který problém potřebuje vyřešit.
- Posouzení projektu, které zahrnuje požadavky, očekávání a případné omezení.
- Stanovení cílů z technického hlediska v rámci provedení a realizace.
- Propracovaný plán s postupem, který zahrnuje, jak daný problém řešit, fáze projektu a jejich odhadovaný čas nebo vhodné technologie pro zpracování.

[32] [33]

2.3.2 Porozumění datům

Na samém počátku je sběr dat, který jako takový předchází samotnému porozumění datům. Další operací je načtení dat a jejich integrace. Následné porozumění datům může probíhat formou vizualizačních technik nebo dotazováním se na konkrétní vzorky dat. Mezi typické příklady těchto procesů patří vykreslení různých distribucí dat a zjišťování jejich statistických vlastností nebo např. rozklad časové řady na jednotlivé složky, jak je vysvětleno v předchozích kapitolách o časových řadách. [32] [34]

2.3.3 Předzpracování dat

Samotná surová data často přicházejí zanesená šumem, nekompletní nebo nekonzistentní. V takovou chvíli přichází na scénu fáze zvaná předzpracování, která má za cíl vzít tyto hrubá data a přetransformovat je do vhodné podoby k dalšímu zpracování. Absence tohoto procesu by mohla mít za následek degradaci kvality výstupů z DM algoritmů. Jedná se o nejdůležitější ale zároveň i mnohdy nejpracnější část celého DM procesu, která může zabírat až 90 % celkového času. Samotné předzpracování se dělí do 4 kategorií, které jsou znázorněné na obrázku č. 6. [34] [35] [36]

Obrázek č. 6 – Jednotlivé úlohy předzpracování [35]

2.3.3.1 Integrace dat

Pro některé úlohy je vzhledem k nedostatku dat z hlavního zdroje potřeba čerpat data z více zdrojů najednou. Tyto zdroje je poté třeba vhodně shromáždit, spojit a vytvořit tak celek. V průběhu tohoto procesu se obvykle dostavují problémy, mezi které lze zařadit:

Integrita schémat – jedná se o nekompatibilitu dat, které sice obsahují nebo reprezentují informace stejného druhu, ale jsou např. jinak pojmenované. Příkladem je sloučení dat ze dvou zdrojů, kdy sloupec z prvního zdroje obsahuje označení ve zkratce a ten druhý ho má jako celý název.

Kolize hodnot dat – ke kolizi může dojít v situaci, kdy data obsahují stejné hodnoty, ale zaznamenané třeba v jiném formátu nebo měřítku. K tomu může dojít např. při agregování časových údajů, které mohou být v jednom zdroji řazeny vzestupně od dnu po roky a v tom druhém zase sestupně.

Duplicity – dva a více zdrojů mohou obsahovat více totožných záznamů, což má za následek nekonzistentnost finálního zdroje dat. [36]

2.3.3.2 Redukce dat

Není až tak neobvyklé, že jsou datové sady obrovské velikosti a obsahují velké množství vlastností. Někdy stačí vybrat pouze nejvhodnější vzorek dat nebo jen určité vlastnosti, které mohou podstatně urychlit výpočetní čas pro další procesy DM. Mezi nejčastější druhy redukce dat patří:

Redukce dimenzí – v rámci datové sady se vhodnou metodou vyberou jen ty vlastnosti, které jsou důležité pro další zpracování.

Redukce četnosti dat – zde se jedná o výběr menšího vzorku dat, který zastupuje celou datovou sadu, která by byla příliš náročná na zpracování vzhledem k výpočetnímu času.

Komprese dat – snížení velikosti dat lze docílit pomocí dvou základních typů komprese, které zahrnují variantu bezeztrátovou a ztrátovou. [35]

2.3.3.3 Očištění dat

Datová sada může mít často různé vady, které zahrnují zašumění dat, neúplné nebo zcela chybějící záznamy. Tyto vady často degradují kvalitu samotných dat a můžou mít za následek špatný výsledek analýzy. Proto je třeba učinit opatření vedoucí ke zkvalitnění dat, které zahrnují:

Vyřešení problémů chybějících dat – volba řešení většinou silně závisí na oblasti problému, ke kterému se tyto data vztahují. Typickými akcemi jsou kompletní vynechání záznamu, nahrazení skrze předcházející nebo následující hodnotu, využití průměru či mediánu. Sofistikovanějšími technikami jsou potom predikce co možná nejvhodnější hodnoty.

Vyřešení problémů s šumem a odlehlými hodnotami – nežádoucí šum je možné vyhladit pomocí procesu, který se nazývá binning. Vyhlazovat můžeme např. skrze průměr nebo medián. Odlehlé hodnoty je poté možné dále vyhlazovat nebo je zcela odstranit a dále k nim přistupovat podobně jako u předchozího problému s chybějícími daty.

Vyřešení problémů s nežádoucími data – může se jednat o nepodstatná data nebo duplicity, které se řeší prostým odstraněním celého záznamu. [36] [37]

2.3.3.4 Transformace dat

Jedná se o proces úpravy dat do vhodnější podoby, která závisí na potřebách daného projektu. Mezi nejpoužívanější operace patří:

Normalizace – označuje způsob, kterým se data napasují do vhodného měřítka. Toto měřítko bývá v intervalu od -1 do 1 nebo od 0 do 1.

Standardizace – jedná se o proces, který částečně slouží k převedení dat do vhodného formátu.

Agregace – tento proces slouží k zjednodušení dat, kdy je možné vhodně posčítat např. počet provedených transakcí zaznamenaných za dny k vytvoření týdenního či měsíčního přehledu.

Vyhlazování – samotné vyhlazování již bylo představeno v podkapitole o čištění dat. I tak je ale třeba vyzdvihnout některé další přínosy, mezi které patří lepší pochopení dat nebo zlepšení výsledků predikce. [35] [36]

2.3.4 Sestavení modelu

V tu chvíli, kdy jsou všechny předchozí kroky zpracování hotové a datová sada je kompletně připravena, přichází na scénu vytvoření vhodného modelu. Pro zhotovení samotného modelu se často využívají různorodé techniky, mezi které lze zařadit matematické a statistické výpočty, algoritmy strojového učení nebo algoritmy umělé inteligence. Takový model je třeba vybrat s přihlédnutím k samotnému cíli projektu a zejména ke specifické oblasti daného problému, která byla již zmíněna v předchozí podkapitole o porozumění problémové domény. Cíle modelu zahrnují nalezení opakujících se vzorů dat a jejich vzájemných vztahů, nebo předpovídání budoucích hodnot. Zjednodušeně řečeno se teda jedná právě o ten krok, který ze surových dat získá potřebné znalosti. [26] [38]

2.3.5 Vyhodnocení

Zde se jedná o fázi vyhodnocení úspěšnosti sestaveného modelu, pomocí kterých se poté rozhoduje o dalším postupu. Dle výsledku hodnocení je pak nutné jednoznačně rozhodnout o tom, kdy je přípustné pustit celkové řešení do další fáze. V případě dobrých výsledků, které jsou ve shodě např. s obchodními plány, se pak celý projekt může posunout do finální fáze. V opačném případě je třeba vrátit se na samý začátek a opětovně projít

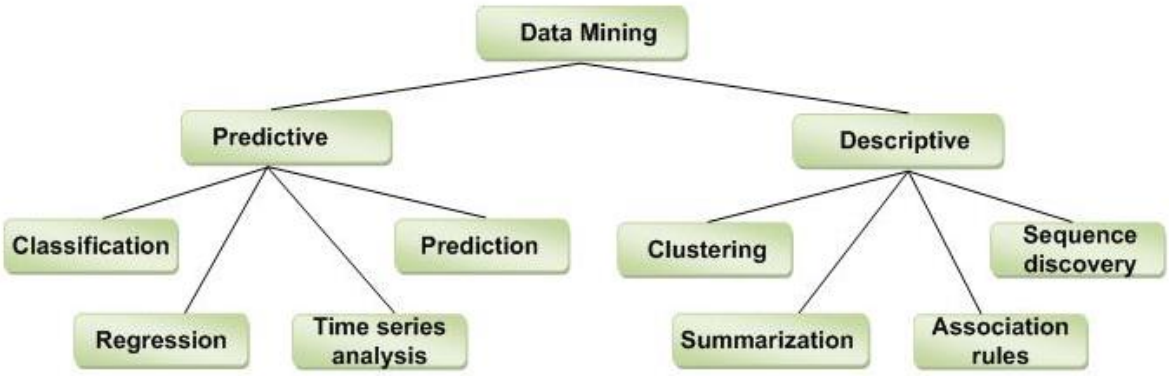
veškeré kroky, které by mohly mít za následek zlepšení úspěšnosti daného modelu a jeho následné zprovoznění v praxi. [34]

2.3.6 Zprovoznění

Poslední fáze DM zahrnuje způsob realizace celkového řešení tak, aby byly výsledky prezentovány s ohledem na uživatelskou přívětivost. Kromě uživatelské přívětivosti je třeba navrhnout i podrobný postup, jak se bude výsledné řešení udržovat nebo sledovat. Je teda potřebné vytvořit uživatelsky přívětivý způsob, který bude mít za následek to, že uživatel bude schopný využít jednodušší ale i složitější funkce. Mezi jednodušší funkce lze zahrnout třeba generování reportu. Ty složitější se vztahují např. na opakovatelnost celého procesu se všemi již zmiňovanými kroky v rámci celé společnosti, kde je potřeba využit toto řešení. [34]

3 PŘEHLED TYPOVÝ ÚLOH

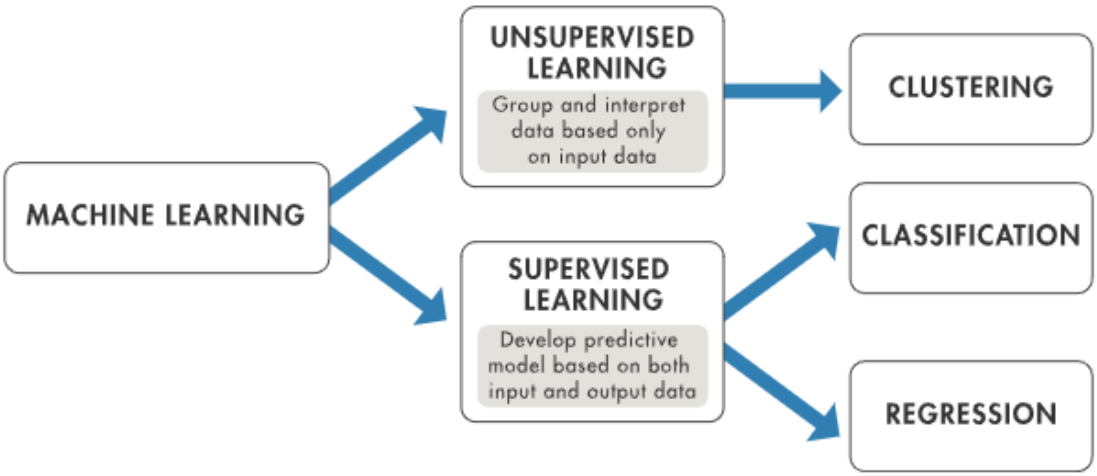
Typové úlohy zahrnují zjišťování důležitých vzorů, vzájemných vztahů mezi daty, vizualizace a shrnutí, rozdělení dat do předefinovaných skupin, řadit data dle podobnosti a vytvářet tak skupiny nové nebo předpovídat následující hodnoty v blízké budoucnosti. Vzhledem k účelu, kterému má typová úloha sloužit, je možné rozdělit úlohy na kategorie, které jsou uvedené na následujícím obrázku č. 7 i s uvedenými příklady. [39]



Obrázek č. 7 - Rozdělení základních typových úloh podle účelu [38]

- **Deskriptivní** – tyto typové úlohy se zabývají pouze popisem toho, co se dělo v minulosti a také tím co se děje zrovna teď. K tomu dochází formulováním otázek, jejichž zodpovězení přinese přesné informace o tom, kde a kdy probíhaly určité typy důležitých událostí, jak často se tyto události objevovaly, co jim předcházelo a tím také jaká byla vlastně samotná příčina těchto událostí. Výsledkem tohoto procesu je poté reakce na tyto události a provedení určitých opatření. Mezi typové úlohy, které jsou popisného charakteru, je možné zařadit např. sumarizaci, klastrování, hledání asociací nebo objevování sekvencí. [39] [40]
- **Prediktivní** – pomocí prediktivních úloh je naopak snaha se na základě historických dat dobrat informací, co by se v budoucnosti mohlo stát. To zahrnuje předpovědi chování určitého objektu, předvídání událostí v blízké budoucnosti nebo zařazení blíže neurčitého objektu do předem definovaných skupin. S tím souvisí i využití prediktivních úloh i jako prevence proti různým druhům událostí, na které by bylo dobré se předem připravit. Zde je třeba počítat s tím, že samotné výsledky nikdy dokonale nereflektují samotnou realitu a jsou teda přesné jen do určité míry. Řadí se zde např. predikce, klasifikace nebo regrese. [39] [40]

V rámci využití strojového učení lze pak tyto úlohy řadit ještě do základních kategorií, které jsou také znázorněny na následujícím obrázku č. 8 i s uvedenými příklady.



Obrázek č. 8 - Rozdělení základních typových úloh podle typu učení [41]

- **Úlohy využívající učení s učitelem** – jedná se o metody využívající vstupní data, které mají i předem daný výstup, k získání souvislostí, díky kterým pak zvládnou předpovídat budoucí vývoj nebo správně zařadit cílová data do předem definovaných skupin. Podle této informace se pak rozlišují dva druhy typových úloh. Jsou jimi regrese zaměřující se na předpověď budoucího chování a klasifikace, které souvisí se zařazením dat do správné skupiny. Příkladem využití regrese je předpovídání cen na burze. U klasifikace by se na druhou stranu jednalo třeba o zařazení senzoru do funkční nebo nefunkční skupiny senzorů. V rámci trénování algoritmů je však tento způsob v mnohých případech dosti náročný na čas. [42] [43]

- **Úlohy využívající učení bez učitele** – jedná se o metody využívající vstupní data bez předem známého výstupu k jejich pochopení a získání logických souvislostí, pomocí kterých je poté možné tyto data popsat. Často tak dochází k detekování vzorů, nalezení předem neznámé struktury nebo vytvoření skupin dat na základě podobností. Nejčastěji se zde řadí typové úlohy, jako je klastrování nebo hledání asociací. Příkladem klastrování by mohlo být vytvoření skupin zákazníků, kteří jsou si podobní svým chováním. Příkladem asociací by poté bylo nalezení vztahu mezi více kupovanými předměty daného zákazníka. V porovnání s typem učení s učitelem je tento způsob o něco méně přesný a často i výpočetně složitější. [43] [44]

3.1 Reprezentace, indexování a hledání podobností

Časové řady jsou často vícerozměrné a obsahují nespočet záznamů, se kterými roste jejich komplexnost. V rámci snížení výpočetního času se proto využívají techniky k zjednodušení těchto složitých časových řad. Toho je možné docílit redukcí dimenzí, která má za následek kratší výpočetní čas pro další úlohy s ohledem na zachování co nejvíce informací z původní časové řady. Dalšími rysy této reprezentace jsou také schopnost zachovat lokální nebo globální tvary jednotlivých křivek a jejich odolnost vůči případnému šumu. [45]

3.1.1 Rozdělení technik pro reprezentaci

Techniky lze rozdělit podle vhodnosti pro data a cílů další analýzy na druhy, které jsou uvedeny v následujících kapitolách.

3.1.1.1 Neadaptivní techniky

Jedná se o techniky, které jsou pevně dané s fixním množstvím parametrů. Mezi takové techniky lze zařadit např. Discrete Fourier Transformation (DFT), Discrete Wawelet Transformation (DWT), Principal Component Analysis (PCA) nebo Piecewise Aggregate Approximation (PAA). [45]

3.1.1.2 Adaptivní techniky

Parametry těchto technik se vhodně volí podle typu dat. Tímto způsobem jsou poté flexibilnější a efektivnější než neadaptivní techniky. Lze mezi ně zařadit např. techniku Singular Values Decomposition (SVD), Symbolic Aggregate Approximation (SAX), Minimum Description Length (MDL) nebo Derivative Segment Approximation (DSA). [45]

3.1.1.3 Založené na modelu

Jde o techniku, která se využívá za předpokladu, že data byly vyprodukovány nějakým modelem. Jinými slovy lze menší dílčí celky reprezentovat jako body proloženou křivku nebo přímku. Mezi tyto modely patří AutoRegressive Moving Average (ARMA) nebo Hidden Markov Models (HMM). [45]

3.1.2 Indexovací struktury

Reprezentace časové řady je dále možné obohatit o indexovou strukturu, která slouží k ještě efektivnějšímu vyhledávání. Tyto vyhledávací procedury je možné rozdělit na tři typy a to:

S rozmezím – dotázáním se z časové řady získá úsek, který je v určitém intervalu.

S přesnou shodou – nalezne úsek časové řady, který je přesně definovaný.

S přibližnou shodou – vyhledá úsek časové řady, který je určen proložením bodů.

Příkladem indexovacích struktur by mohly být TS-tree, R-tree nebo R*-tree. Samotné indexovací struktury jsou děleny ještě do dvou typů, které jsou podrobně vysvětleny v následujících podkapitolách. [45]

3.1.2.1 Struktury založené na vektorech

Po redukování dimenzí vícerozměrné časové řady vznikají komprimované vektorové úseky, které jsou dále rozděleny do jednotlivých shluků. Uspořádaní samotných shluků může být hierarchické nebo nehierarchické. [45]

3.1.2.2 Struktury založené na metrikách

Oproti předchozímu typu nevyužívá zkomprimovaných vlastností, ale pracuje na relativním vzdálenostním přístupu. [45]

3.2 Sumarizace a vizualizace

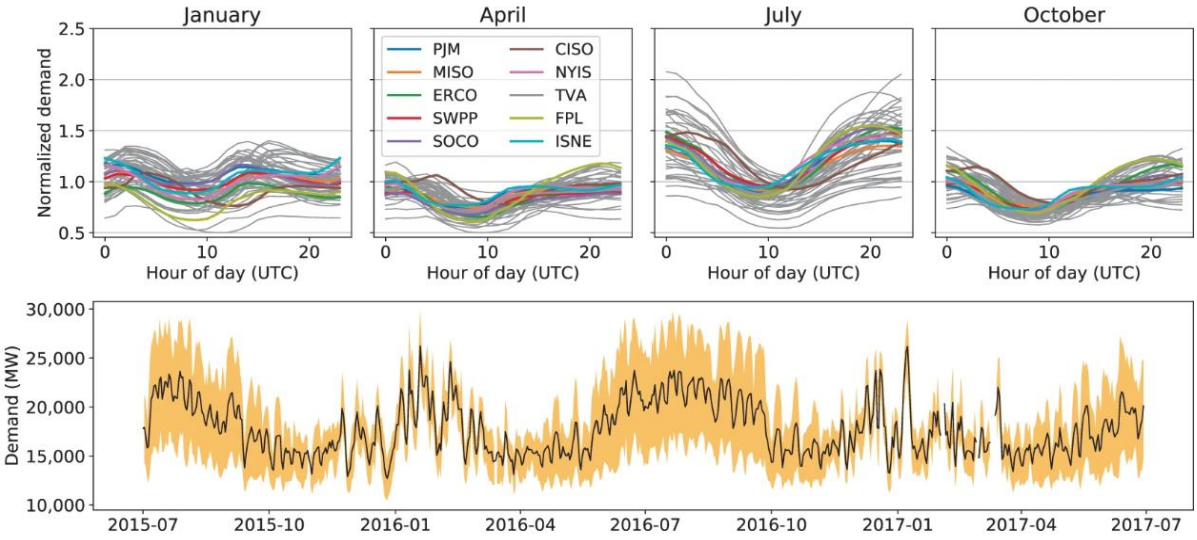
Samotné datové sady časových řad jsou často velmi komplexní. V jednom případě se jedná třeba o vykreslení běžného spojnicového grafu pro velké množství dat jednorozměrné časové řady, který by byl při takovém množství dat velmi nepřehledný. V jiném případě zde může zase nastat problém, že je časová řada vícerozměrná a uvědomit si v takové případě souvislosti mezi daty z jednotlivých spojnicových grafů je často nereálné. Proto existují techniky, které jsou schopné tyto problémy vyřešit a tím přinést cenné informace. Tyto techniky jsou dále vysvětleny v následujících podkapitolách. [45]

3.2.1 Vyhledávání podle časového intervalu

Tato technika využívá výběru vzorku dat skrze určení časového intervalu. Tento interval neboli dílčí část časové řady se poté může využít pro vykreslení grafu, který zobrazí více podrobností a informací právě o tomto konkrétním úseku časové řady. [45]

3.2.2 Vizualizace založená na kalendáři a shlucích

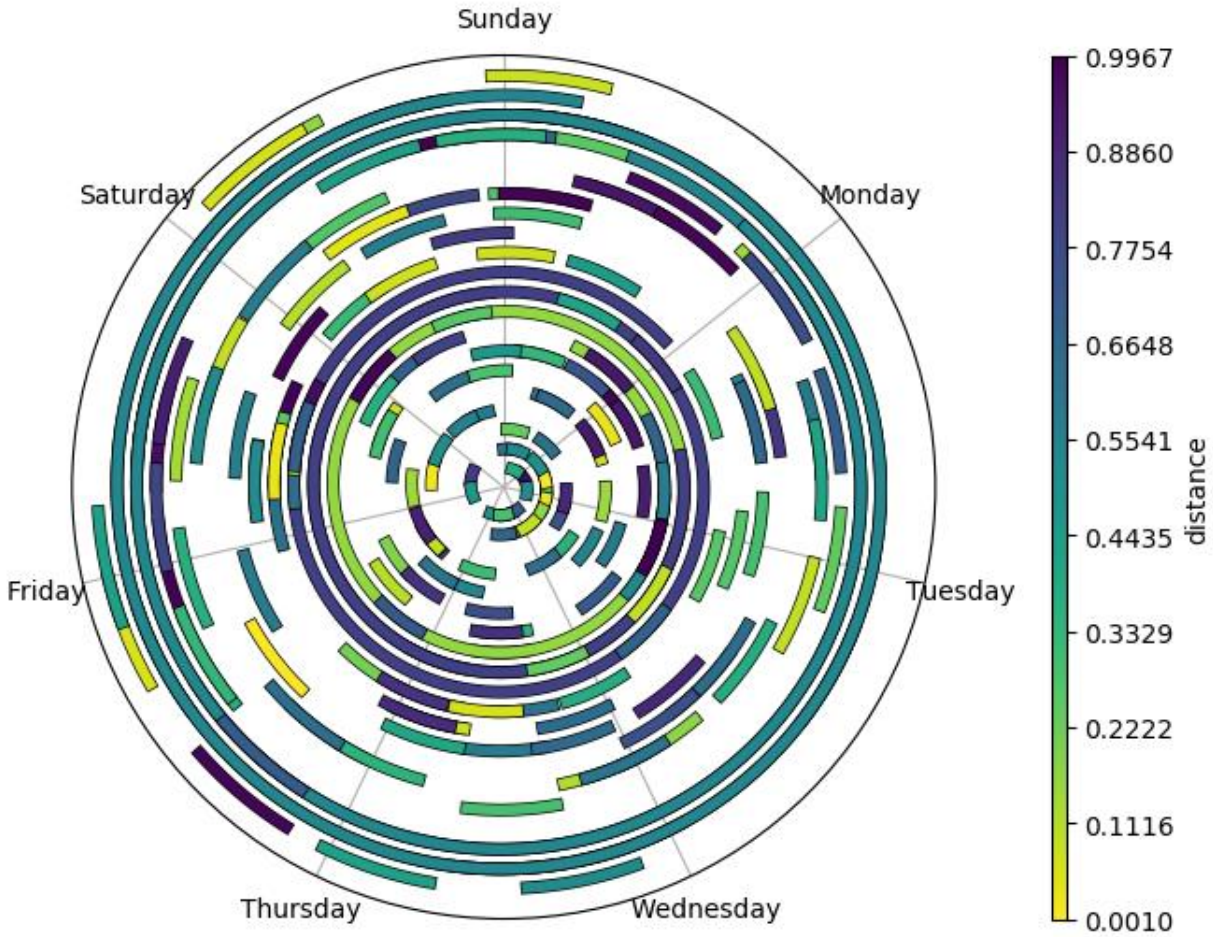
Tímto způsobem je možné data různě seskupovat podle hodin, dnů, týdnů, měsíců a roků. Tyto data se seskupí či agregují a podle potřeby se dále vytvoří průměr, medián nebo suma, které jsou potom reprezentovány v jednotlivých grafech jako souhrn pro určité časové období. Jako příklad poslouží teploty zaznamenané po hodinách, které je poté možné převést na průměrnou teplotu za den. Jiným příkladem by pak bylo např. shlukování dnů s podobným průběhem teplot. Obrázek č. 9 zobrazuje vizualizaci ve formě souhrnů hodnot za období pro měsíce leden, duben, července a srpen. [45]



Obrázek č. 9 – Příklad souhrnu dat za jednotlivé měsíce [46]

3.2.3 Vizualizace založené na spirále

Tato vizualizace je způsob, jakým lze vykreslit data časovou řadu do kruhové podoby. Z hlediska využití se tak jedná zejména o vykreslení periodických časových řad. Tímto způsobem je poté dobře rozeznatelné, jak se časová řada průběžně chová ve stejném periodickém úseku. Příklad tohoto typu vizualizace je zobrazený v obrázku č. 10. [45]



Obrázek č. 10 – Příklad zobrazení dat ve spirále [47]

3.2.4 Vizualizace založené na stromové struktuře

Jedná se o způsob vizualizace, jakým lze docílit odhalení předem neznámých opakujících se vzorů a odlehlých hodnot. Časová řada se rozloží na jednotlivé symbolické reprezentace. Dále se pak vykreslí do stromové struktury, která obsahuje větve s barevným značení a jiné vizuální prvky pro určení frekvence a vlastností jednotlivých vzorů. Je nutno zmínit, že časovou řadu je potřeba předem diskretizovat. [45]

3.3 Segmentace a hledání bodů změny

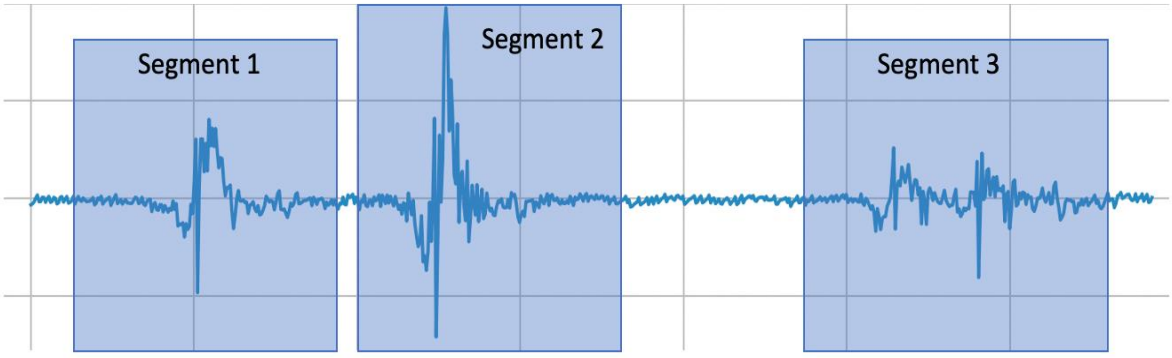
Segmentace je ve své podstatě velmi podobná typová úloha jako reprezentace časových řad. Obě typové úlohy redukují dimenze časových řad a tím vytvářejí segmenty, které zachovají rysy a základní informace o původní časové řadě. Rozdíl je zejména v tom, že reprezentace řeší redukci dimenzí časové řady implicitně. V rámci čisté segmentace existují algoritmy, které různými přístupy dělí časovou řadu na jednotlivé segmenty. Mezi tyto algoritmy je možné zařadit:

Sliding windows – segment se neustále zvětšuje, dokud nepřekoná určitou hranici maximální chybovosti pro tento konkrétní úsek.

Top-down – časová řada se dělí dále na menší úseky, dokud není porušena určitá podmínka.

Bottom-up – funguje na opačném principu jako Top-down, takže z menších celků vytváří postupně větší segmenty.

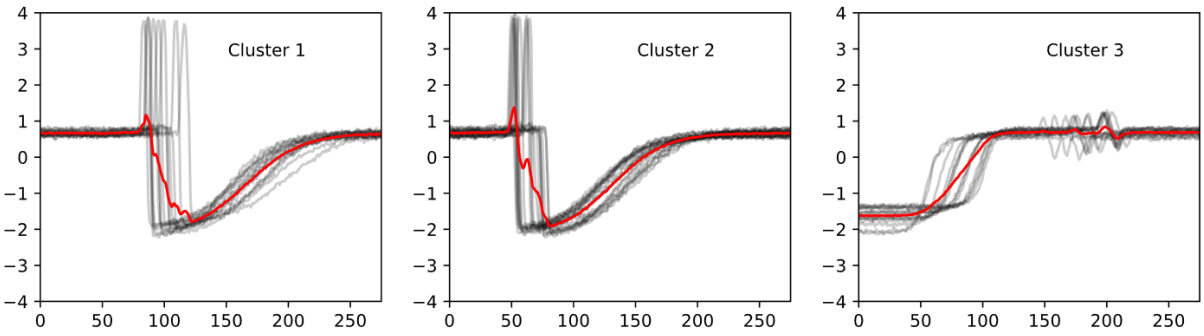
Ukázka samotné segmentace je k vidění na obrázku č. 11. [49]



Obrázek č. 11 – Příklad segmentování [50]

3.4 Shluková analýza

Klastrování neboli shluková analýza je technika strojového učení, která využívá učení bez učitele k získání jistých vzorů dat nebo k seskupení předem neurčených dat do shluků, ve kterých jsou si tyto data vzájemně podobné. Zobrazení shluků přímo pro časové řady je k vidění na následujícím obrázku č. 12. [51]



Obrázek č. 12 – Příklad zobrazení jednotlivých shluků dat [52]

3.4.1 Rozdělení

Algoritmy a typy shlukové analýzy lze rozdělit na další podskupiny, které jsou k vidění níže v dalších podkapitolách. [53]

3.4.1.1 Hierarchické

V rámci hierarchického klastrování dochází k vytváření jednotlivých shluků v určitém pořadí. Hierarchické je odvezeno od faktu, že jsou jednotlivé shluky řazeny v určité struktuře, která je velmi podobná struktuře stromové. Tato struktura je dána tím, že se vzájemně podobné shluky seskupují. Díky této struktuře jsou rozložení shluků vcelku přehledné, ale dochází tak i ke sčítání chybovosti, která má za následek, že je tato technika často nespolehlivá. Tento typ shlukové analýzy se v rámci dalších podskupin dělí ještě na dělící a aglomerativní. [54]

- **Aglomerativní hierarchické shlukování** – bere v potaz každý jednotlivý bod jako samostatný shluk, který je následně spojován do větších dílčích celků. Tento typ by bylo možné nazvat i jako vzestupný. [53]
- **Dělící hierarchické shlukování** – druhou variantou je případ, kdy jsou všechny body obsaženy v jednom shluku, který se pak dále dělí na menší. Funguje tedy na opačném přístupu než aglomerativní, a proto je možné ho nazývat sestupným. [53]

3.4.1.2 Nehierarchické

Oproti hierarchickému způsobu se zde nevyužívá specifických struktur jako jsou ty stromové, ale využívá se dělení klastrů na menší dílčí celky, které mezi sebou nemají hierarchický vztah. V tomto případě jsou sice shluky méně přehledné, ale na druhou stranu je celý proces stabilnější, s menší chybovostí, a i výsledky jsou tím pádem věruhodnější. [54]

3.4.1.3 Shlukování založené na modelu

Jedná se o typ shlukové analýzy, který je založen na statistickém přístupu. U více rozměrných dat se ke každému vlastnosti přistupuje jako k dílčímu celku, který je reprezentován určitou distribucí. Tento přístup se snaží těžit ze situace, kdy je známo, že data jsou tvořena právě kombinací různých distribucí dat. [55]

3.4.1.4 Shlukování založené na hustotě

Funguje na principu shlukování dat do skupin, které se vyznačují vysokou hustotou těchto dat. Oblasti, které jsou mimo sousedící izolované shluky s vysokou hustotou, jsou poté označovány jako šum. [53]

3.4.1.5 Shlukování založené na hybridním přístupu

Zde se využívá principu kombinování jednotlivých algoritmů pro shlukovou analýzu. Tím lze dosáhnout optimálních výsledků, které těží z výhod použitých typu algoritmů. Tento přístup se používá zejména tehdy, kdy je datová sada velmi komplexní. [53]

3.4.2 Algoritmy

V následujících podkapitolách jsou uvedeny některé příklady algoritmů, které spadají do výše zmíněných skupin a podskupin.

3.4.2.1 K-Means

Je to snad jeden z nejvíce jednoduchých algoritmů typu učení bez učitele, které se používají ke shlukové analýze. Využívá se zde volby čísla K, které slouží k definování počtu centroidů, které si lze představit jako průměry datových bodů tvořící středy shluků. Na základě nejbližších centroidů se poté jednotlivé datové body přiřadí k určitému shluku. [56]

3.4.2.2 K-Medoids

Tento algoritmus lze označit jako modifikovanou verzi K-Means, která odstraňuje jeho nedostatky. Mezi tyto nedostatky patří to, že K-Means je náchylný na odlehlé hodnoty, které zvyšují jeho průměr. K-Medoids tento problém řeší tak, že místo průměru využívá bod, kterému se říká medoid. Tento medoid využívá nejmenšího možného součtu vzdáleností od datových bodů. [57]

3.4.2.3 Fuzzy C-Means

Nejedná se o nic jiného než další modifikovanou variantu K-Means, která v rámci jednotlivých datových bodů připouští i pravděpodobnost, se kterou datové body náleží do určitého shluku. Tento algoritmus je sice pomalejší než jeho základní varianta K-Means, ale na druhou stranu produkuje lepší výsledky i pro promíchaná data. [58]

3.4.2.4 DBSCAN

DBSCAN (angl. Density-Based Spatial Clustering of Applications with Noise) je velmi známý algoritmus založený na hustotě, který seskupuje jednotlivé datové body podle vzdálenosti a samotného minimálního počtu daných bodů, které závisí na počtu dimenzí datové sady. Samotné minimum bodů jsou 3 body. Zde je ale nutné si uvědomit, že s rostoucí velikostí datové sady musí být větší i minimální počet stanovených bodů. Dále je potřeba se zvolit ještě parametr *eps*, který při příliš nízké hodnotě sice seskupí data do shluků, ale spousta datových bodů je poté označena jako odlehlá hodnota. V rámci příliš vysokého čísla se pak často stává, že se jednotlivé body seskupí pouze do jedno shluku. Obecně se ale vždy volí spíše nižší hodnota. Kromě hledání vzorů a asociací se často využívá i na detekování odlehlých hodnot, které jsou charakteristické pro oblast shluků s nižší hodnotou hustoty. [59]

3.4.2.5 GMM

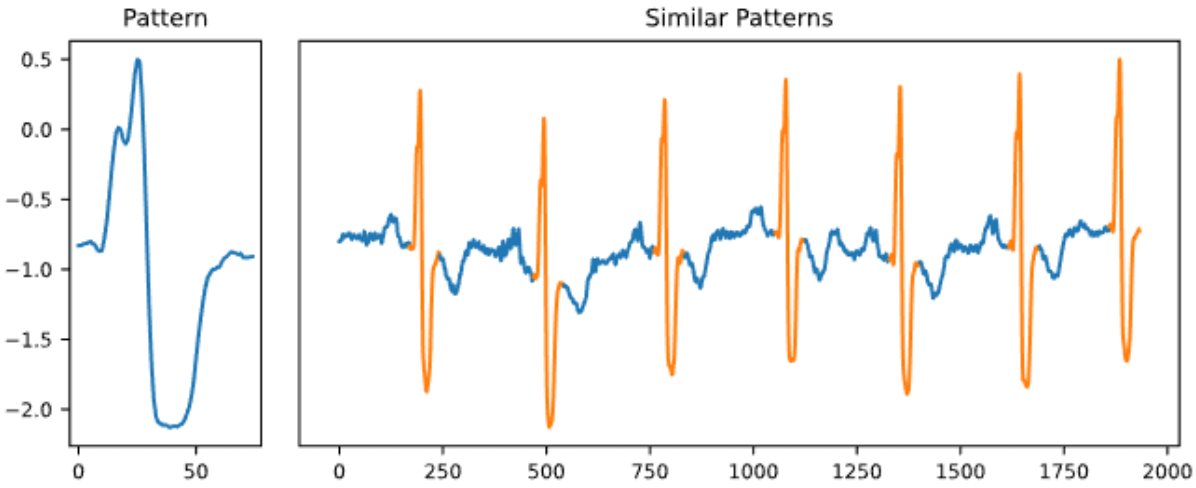
Model Gaussovské směsi (angl. Gaussian Mixture Model) při shlukování nachází využití zejména tehdy, kdy data nejsou čistě normální rozdělení, tj. mají několik vrcholů. Další případ pro jeho využití je situace, kdy shluky nejsou přímo kulovitého tvaru. Z toho vyplývá, že je tento způsob velmi vhodný pro komplikované datové sady. Často se využívají u časových řad pro zjištění trendu nebo rozpoznávání vzorů, které zahrnují i sezonní a cyklické chování. Další výhodou je i detekce odlehlých hodnot, které se vyznačují nízkou hustotou pravděpodobnosti. [60]

3.4.2.6 SOM

Samoorganizující se mapy (angl. Self Organizing Maps) jsou druh neuronových sítí využívajících učení bez učitele. Tyto neuronové sítě se však od ostatních sítí poměrně dost liší. Využívají redukce dimenzí vícerozměrných vstupních dat k vytvoření nízkorozměrné mapy, která nabývá většinou dvou rozměrů. Oproti běžným neuronovým sítím, které využívají zjištění chyby a následného aktualizování jednotlivých vah, se zde používá principu učení s odměňováním. To funguje tak, že se prvně nastaví vektory vah. Poté se náhodně vybere vzorový vektor, který se následně porovnává, aby se našli váhy jemu nejvíce podobné. Váhy, které jsou vybrány, jsou poté ohodnoceny jako více pravděpodobné. To samé se provede i u sousedních vah těchto vektorů, které jsou poté také více pravděpodobné v porovnání vůči vzorovému vektoru. Tímto způsobem se pak mapa mění do různých podob, které často nabývají čtvercových, obdélníkových nebo hexagonálních tvarů. [61]

3.5 Hledání motivů

Tato typová úloha má za cíl hledání úseků, které lze považovat jako odlehlé hodnoty, překvapivé nebo frekventovaně opakující se vzory. K hledání motivů se často může přistupovat jako ke shlukové analýze pro seskupování podobných dat. Při hledání určitého motivu však tato typová úloha často vyžaduje i parametr, který slouží pro definování délky tohoto motivu. Ukázka opakujícího se vzoru i s testovanou časovou řadou je znázorněna na následujícím obrázku č. 13. [45]



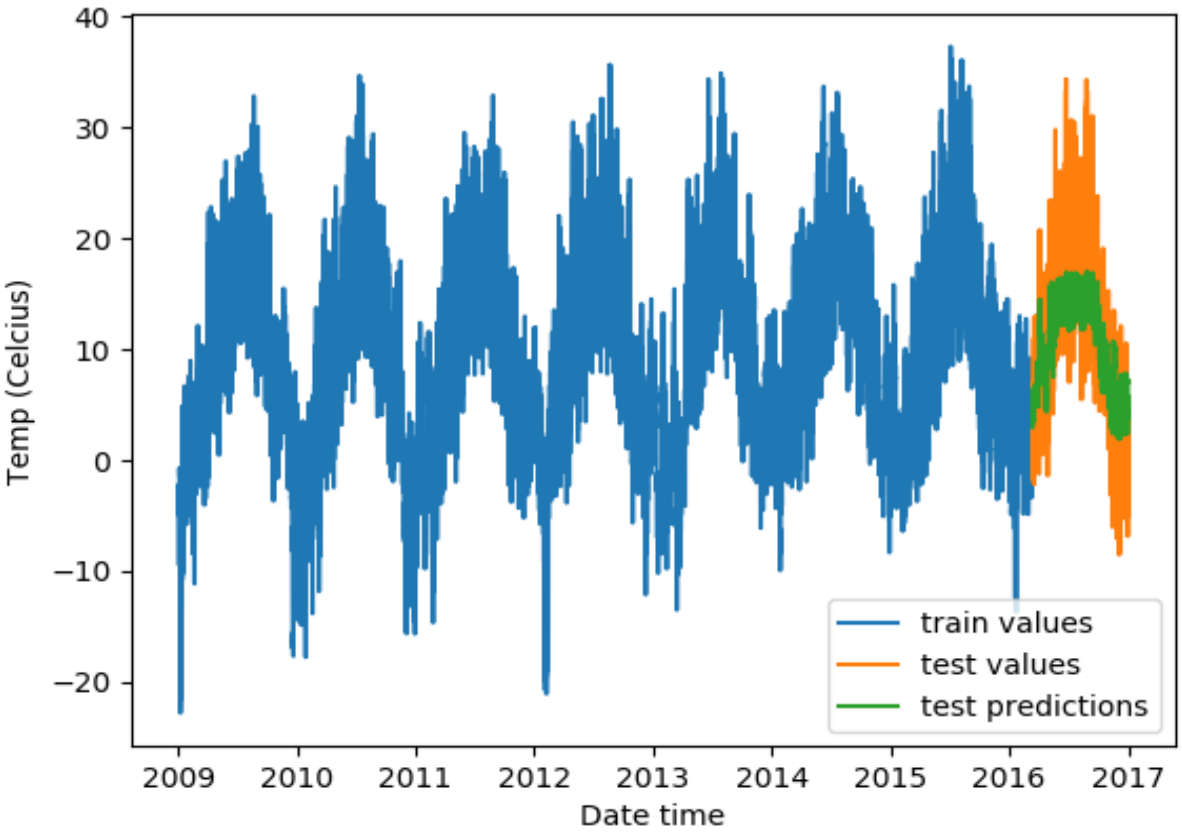
Obrázek č. 13 – Příklad hledání vzorů [62]

3.6 Hledání asociačních pravidel

Jedná se o vyhledávání blízkých vztahů mezi více vlastnostmi, které jsou přítomné v opakujících se vzorech. Aby však bylo možné vzájemné vztahy nějak nalézt, je třeba nejprve vyhledat často opakující se vzory v časových řadách, které jsou jako samostatná typová úloha popsány v předchozí kapitole. [45]

3.7 Predikce – regrese

Tato typová úloha je zodpovědná za modelování vztahů mezi historickými daty časové řady, které lze poté použít pro predikce neboli předpovědi budoucích hodnot. Znázornění predikce i s jednotlivými částmi pro trénování a testování je na obrázku č. 14. [63]



Obrázek č. 14 – Příklad regrese (predikce) [64]

3.7.1 Algoritmy a modely

Algoritmy a modely, které jsou specifické pro regrese jsou uvedeny v dalších podkapitolách.

3.7.1.1 ARMA

Jedná se o model složený ze dvou částí pro předpovídání následujících hodnot. První částí je AR, která označuje autoregresy (angl. AutoRegression). Tento proces je možné chápat jako předpovídání následující hodnoty na základě n-počtu hodnot předchozích. Tento počet předcházejících hodnot je znázorněn parametrem *p*. Další částí je MA což je klouzavý průměr (angl. Moving Average), který využívá průměru časové řady a n-počtu předchozích chyb. Počet předchozích chyb je označený parametrem *q*. [65]

3.7.1.2 ARIMA

Samotná ARMA označuje autoregresní model s klouzavým průměrem. Model ARIMA (angl. AutoRegressive Integrated Moving Average) vznikne obohacením tohoto modelu o složku I, která označuje počet diferencí, aby se časová řada mohla stát stacionární. Z toho vyplývá, že u časové řady, která diferenciování nepotřebuje, je nejvhodnější právě ARMA. Parametr složky I se nazývá *d* a označuje tedy počet diferencí. [65]

3.7.1.3 SARIMA

Pod zkratkou SARIMA (angl. Seasonal AutoRegressive Moving Average) se skrývá sezonní autoregresní model s klouzavým průměrem. Jedná se o modifikovanou variantu modelu ARIMA. Jediný rozdíl je v tom, že tento typ modelu má vylepšené diferenciování, které mu umožňuje použít nastavení i podle frekvence sezony. [66]

3.7.1.4 SARIMAX

Model SARIMAX (angl. Seasonal AutoRegressive Moving Average with eXogenous factors) není nic jiného než model SARIMA, který je modifikovaný tak, aby byl schopný zahrnout i externí vlivy. [66]

3.7.1.5 VAR

V případě, že je třeba modelovat vícerozměrné časové řady, tak se využívá vektorového modelu VAR (angl. Vector AutoRegression), popř. jeho modifikací, které zahrnují i obyčejnou variantu s klouzavým průměrem VMA (angl. Vector Moving Average), kombinaci obou složek v rámci modelu VARMA, možnost diferenciování u modelu VARIMA nebo zahrnutí externích vlivů u modelu VARIMAX. Samotné značení vyplývá z předchozí varianty pro jednorozměrné časové řady. [67]

3.7.1.6 GARCH

Některé časové řady často vykazují měnící se rozptyl v čase. Toto chování časové řady je tedy charakteristické zvýšeným kolísáním hodnot. V takovém případě většinou není úplně vhodné používat modely jako je ARIMA. Model ARCH (angl. Autoregressive Conditional Heteroskedasticity), který by bylo možné česky označit jako model s podmíněnou heteroskedasticitou, využívá explicitního modelování rozptylu v průběhu času. V rámci modelu GARCH (angl. Generalized Autoregressive Conditional Heteroskedasticity) se zde jedná o zobecněné rozšíření modelu ARCH, které pouze doplňuje o klouzavý průměr. [68]

3.7.1.7 Gradient Boosting

Jedná se o velmi efektivní algoritmus strojové učení, který se využívá nejen pro regrese, ale také i pro klasifikace. Samotný princip fungovaní je založen na posilování skrze kombinování několika modelů, kdy přidáním nového modelu lze docílit opravení těch více chybových. [69]

3.7.1.8 Random Forest

Algoritmus náhodný les (angl. Random Forest) typu učení s učitelem, který hodnotí hodnoty jednotlivých vlastností, které pak využívá pro zjištění nejmenší možné chybovosti. První najde nejdůležitější vlastnost, ze které pak udělá kořenový prvek. Tento prvek pak rozdělí datovou sadu na více částí. Podobným způsobem pak proces probíhá i u dalších vlastností časové řady, dokud nenarazí na dno stromu nebo nejsou splněná určitá kritéria. [70]

3.7.1.9 LSTM

LSTM (angl. Long Short Term Memory) speciální typ neuronové sítě pro jednorozměrné i vícerozměrné časové řady je rozšířením rekurentní neuronové sítě, která využívá algoritmu zpětného šíření chyby. Místo neuronů má tato síť několik paměťových bloků, které jsou pospojovány napříč všemi vrstvami. Tímto způsobem si síť v těchto blocích zaznamenává aktuální stavy a tím si dokáže pamatovat obrovské sekvence dat. Jedná se tak o efektivní způsob, jak si udržet informace o gradientu. [71]

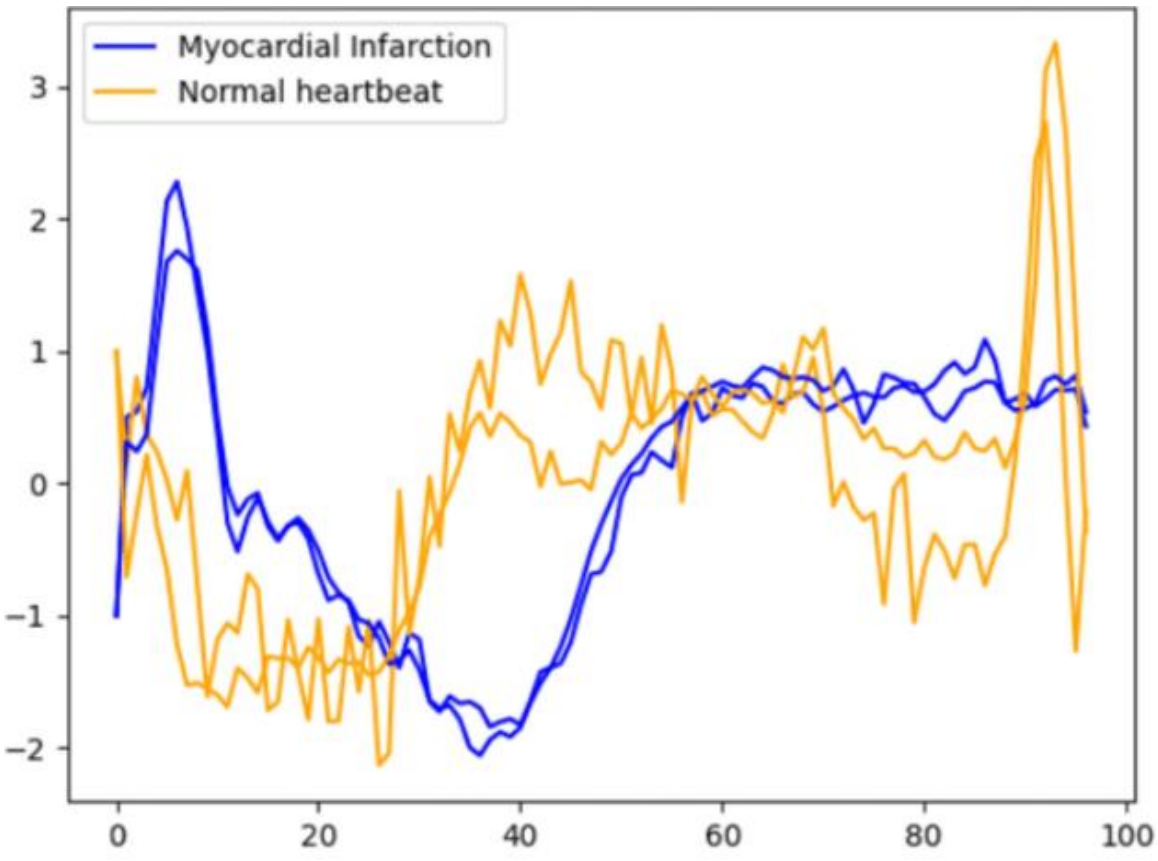
3.7.1.10 TBATS

V případě časové řady, která obsahuje mnoho složitých a četných sezonních vzorů, se zde naskytuje možnost použití modelu TBATS (angl. Trigonometric Seasonal, Box-Cox Transformation, ARMA residuals, Trend and Seasonality) nebo jeho zredukovanou variantu

BATS (angl. Box-Cox Transformation, ARMA residuals, Trend and Seasonality). Tento model hledá co možná nejvíce vhodný způsob, jak vytvořit finální model na základě několika dalších technik a modelů, mezi které se řadí Box-Coxova transformace, případné aplikování modelu ARIMA či ARMA, uvážení sezony a trendu i s jeho tlumením nebo využití trigonometrické sezonnosti. V rámci nalezení vhodné kombinace je však tento typ algoritmu nevhodný pro rozsáhlé časové řady vzhledem k délce výpočetního času. Také je třeba zmínit, že oproti modelům jako je SARIMAX zde není možnost, jak zahrnout externí vlivy. [72]

3.8 Predikce – klasifikace

Jedná se o typovou úlohu učení se s učitelem, kdy pomocí více časových řad, které jsou přiřazeny k určité skupině, jsme schopni zařadit neznámou časovou řadu do jedné z těchto skupin. Samotné data je nejprve nutné rozdělit na dvě hlavní složky, a to na trénovací a testovací. Algoritmus se nejdříve naučí na trénovacích datech a následně se zkouší úspěšnost na těch testovacích. Často je zde přítomná i validační část, která slouží k určení optimálního algoritmu s nejvyšší přesností. Příklad klasifikace je znázorněn na obrázku č. 15. [73]



Obrázek č. 15 – Příklad klasifikace (predikce) [74]

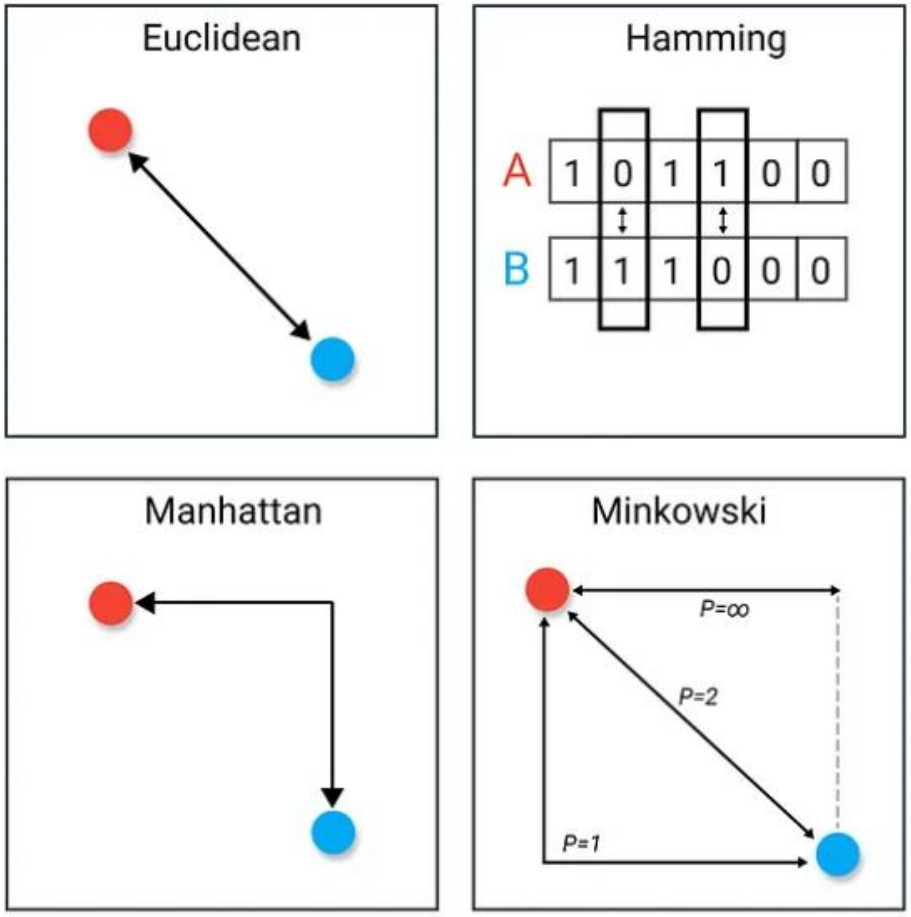
Samotná úspěšnost klasifikování se poté musí ještě odzkoušet např. pomocí matice zmatku, která obsahuje shrnutí správně a špatně klasifikovaných prvků. [75]

3.8.1 Rozdělení

Na klasifikaci se lze dívat z různých úhlů pohledu. Tyto jednotlivé přístupy ke klasifikací jsou dále vysvětleny v nadcházejících podkapitolách. [76]

3.8.1.1 Přístupy založené na vzdálenostech

Pracují na principu relativního porovnání dvou a více časových řad. Podle vzdálenosti jednotlivých bodů se pak u časové řady určuje míra podobnosti. Příklady takových vzdáleností, často označovaných i jako metrik, jsou potom Euklidovská, Manhattanská, Hammingova nebo Minkowského, které jsou znázorněny na obrázku č. 16. [76]



Obrázek č. 16 – Příklad druhů metrik [77]

3.8.1.2 Přístupy založené na intervalech

Využívají rozdělení časové řady na úseky, které se následně zpracují jednotlivými algoritmy. Výsledné klasifikování pak probíhá podle toho, která třída se vyskytovala nejčastěji. [76]

3.8.1.3 Přístupy založené na slovníku

Tyto přístupy jsou založeny na vhodném pojmenování křivek a následném zaznamenávání kolikrát se křivky v časových řadách celkově objeví. [76]

3.8.1.4 Přístupy založené na tvaru křivky

Svým způsobem se jedná o přístup velmi podobný jako je ten založený na vzdálenosti. Liší se pouze tím, že vůči třídě porovnává všechny možné úseky časových řad. [76]

3.8.1.5 Přístupy založené na kombinování

Skrze kombinaci několika klasifikátorů, kdy každý z nich zodpovídá za jednu třídu. U tohoto přístupu je důležité, aby chybovost jednotlivých klasifikátorů nebyla korelována. [76]

3.8.2 Algoritmy

Klasifikaci časových řad je možné zrealizovat pomocí několik algoritmů. Tyto algoritmy jsou detailněji popsány v následujících podkapitolách.

3.8.2.1 K-NN

K-NN (angl. K-Nearest Neighbours) je jeden z nejjednodušších a nejpopulárnější algoritmů pro klasifikaci založeného na vzdálenostním přístupu. Principiálně funguje tak, že si algoritmus dopočítá Euklidovskou vzdálenost a najde K počet nebližších prvků, jejichž počet je třeba si nejprve zvolit. Následně neurčitý prvek přiřadí do třídy, která je zastoupena více prvky dané třídy. [78]

3.8.2.2 SVM

Algoritmus SVM (angl. Support Vector Machine) lze označit jako jeden z nejvíce populárních algoritmů, jehož princip je založen na nalezení extrémních bodů, které musí vytvořit co největší prostor, skrze který pak mohou být jednotlivé třídy odděleny od sebe. Díky maximalizaci tohoto prostoru je pak u předem neznámého prvku možné celkem přesně zjistit, do které třídy vlastně spadá. [79]

3.8.2.3 Logistic Regression

Logistická regrese je klasifikační technika pro vyhledávání vztahů mezi dvěma a více nezávislými proměnnými. V rámci klasifikace slouží tahle technika nejčastěji k oddělení dat křivkou do dvou tříd. Pro zjištění výkonu se pak používají metriky nebo speciální funkce pro odhadnutí výkonu. Jedná o efektivní techniku k trénování, která se dobře aplikuje na lineárně oddělitelné datové sady. V rámci účinnosti je však často překonávána výkonnějšími algoritmy, které ji mnohdy předčí nejen výkonem ale i svou kompaktností. [80]

3.8.2.4 Deep Learning

Modely hlubokého učení jsou typy komplexních neuronových sítí, které jsou založené na učení se velkého množství parametrů. První vrstvy této sítě vždy slouží k rozeznání vzoru, zatím co ty poslední určují, jaké bude rozdělení do jednotlivých tříd. Ještě do nedávna tyto typy nebyly zrovna nejvhodnější pro časové řady, nicméně aktuálně už vykazují celkem dobrý poměr přesnost ku výpočetnímu času, který by se mohl poměřovat i s populárnějšími modely. [81]

3.8.2.5 DTW

Algoritmus DTW (angl. Dynamic Time Warping) odstraňuje problém Euklidovské vzdálenosti, která je příliš náchylná na posunutí v čase. Vyhledává vždy nejkratší spojení dvou bodů a tím minimalizuje zkreslení v čase nebo časový posun částí časové řady. [82]

3.8.2.6 Random Forest

Algoritmus Random Forest již byl popsán v předchozí kapitole o regresích. Nicméně je třeba zmínit, že tento algoritmu nachází dobré využití i při klasifikačních úlohách. [83]

3.8.2.7 Decision Trees

Rozhodovací stromy (angl. Decision Trees) je dalším algoritmem typu učení s učitelem, který se hojně vyskytuje u klasifikačních úloh. Funguje na bázi rozdělování dat do menších podskupin. Tuto operaci provádí tak dlouho, dokud není splněno určité kritérium, které může zahrnovat např. maximální hloubku zanoření. [84]

3.8.2.8 Gaussian Naive Bayes

GNB (angl. Gaussian Naive Bayes) je další algoritmus vhodný pro klasifikaci, který má položené základy na Bayesově větě. Jedná se o rozšířenou variantu původního algoritmu Naive Bayes. Tento algoritmus je jednoduchý a vhodný na rychlé klasifikace. [85]

https://www.upgrad.com/blog/gaussian-naive-bayes/

3.9 Detekce odlehlých hodnot

Odlehlá hodnota nebo anomálie je typ záznamu, který se projevuje výrazně jinak než většina ostatních záznamů. Těmto záznamům by se vždy měla věnovat pozornost, protože mohou zásadně ovlivnit výsledky dalších operací. Podle jejich významu je můžeme dělit na:

Nechtěné události – jsou to záznamy, které mohou vzniknout např. poškozením senzoru. Tyto data je dále potřeba odstranit.

Přínosné události – zde se jedná o určitý záznam, který může při vhodné analýze přinést cenné informace.

Dále je možné tyto záznamy rozdělit podle rozsahu porovnávání s ostatními záznamy na:

Lokální – takhle jsou identifikovány ty záznamy, které se vyznačují jiným chování oproti záznamům v blízkém okolí.

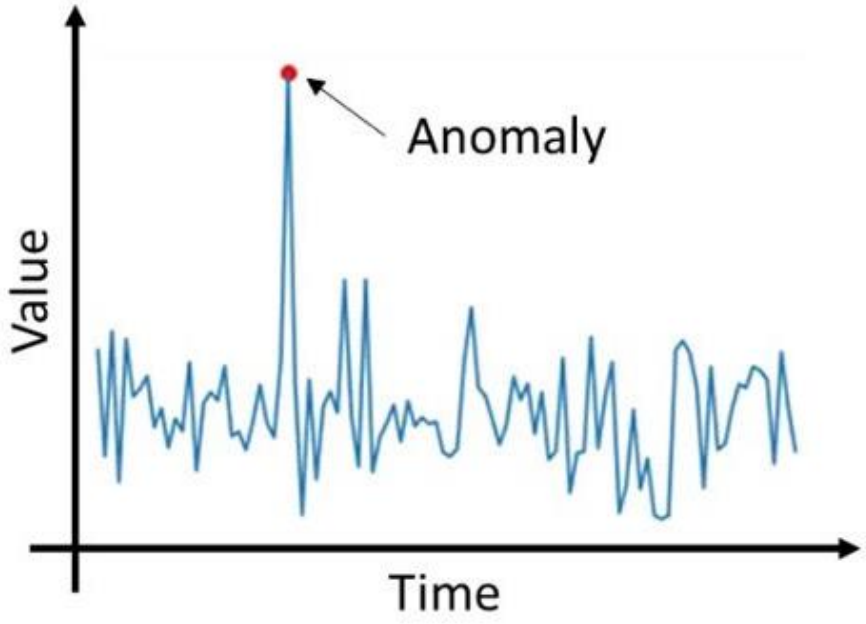
Globální – tyto záznamy jsou označovány jako globální, protože se jasně odlišují vůči chování v rámci celé časové řady.

Podle samotného rozsahu těchto anomálií je nakonec dělíme ještě na:

Bodové – jedná se pouze o jeden nečekaný záznam.

Sekvenční – představují sled několika po sobě jdoucích záznamů, které společně tvoří sekvenci vykazující atypické chování vůči časové řadě.

Ukázka detekce odlehlých hodnot je znázorněna na následujícím obrázku č. 17. [86]



Obrázek č. 17 – Příklad detekce odlehlé hodnoty [87]

3.9.1 Algoritmy a techniky k detekování

V rámci detekce anomálií existuje celá řada algoritmů a technik, které lze svou povahou zařadit mezi statistické, využívající predikcí, neuronových sítí a mnoho dalších. Příklady některých technik a algoritmů jsou uvedeny v podkapitolách níže. [86]

3.9.1.1 Z-Score

Využívá se zejména tehdy, kdy jsou data v normální distribuci. Pracuje na principu, který využívá počtu směrodatných odchylek k detekování odlehlých hodnot. Například pomocí Z-Score s hodnotou 3 je možné detekovat odlehlé hodnoty vzdálené více než 3 směrodatné odchylky od průměru. [88]

3.9.1.2 IQR

IQR (angl. InterQuartile Range) pracuje na principu použití rozdílové vzdálenosti mezi prvním a třetím kvartilem, kterou vynásobí 1,5x ve směru do plusu i do mínusu. Všechny ostatní body, které leží mimo tento interval, jsou označeny jako potenciální odlehlé hodnoty. [89]

3.9.1.3 Autoencoder

Autokoder je druh neuronové sítě, která se skládá z kódovací a dekodovací části. Při kódování využívá získané vzory časové řady, které následně skládá a zase rozkládá. Díky tomuto rozkladu zde vznikají chyby, které se v případě odlehlých hodnot zvyšují. [86]

3.9.1.4 PCA

Analýza hlavních komponent (angl. Principal Component Analysis) rozkládá data časových řad do více menších dimenzí a ty potom zase zpětně skládá. Podobně jako u autokodérů zde opětovným rozkladem vznikají chyby, které se v případě odlehlých hodnot mají tendenci zvyšovat. [86]

3.9.1.5 LOF

Faktor lokální odlehlé hodnoty (angl. Local Outlier Factor) je algoritmus typu učení bez učitele, který podle blízkosti sousedních bodů vypočítá skóre, které označuje hustotu sousedních bodů. Toto skóre se vždy porovnává i se skórem sousedních bodů. Je-li tato hodnota od ostatních příliš vysoká, tak je poté možné označit tento bod za odlehlou hodnotu. [90]

3.9.1.6 OCSVM

Jednotřídová metoda podpůrných vektorů (angl. One Class Support Vector Machines) je jeden z oblíbených algoritmů, které lze použít na nalezení případné odlehlé hodnoty. Jedná se o algoritmus, který pracuje na principu minimalizování prostoru dané třídy. Záznamy mimo ohraničený prostor této třídy lze považovat za odlehlé hodnoty. [91]

3.9.1.7 Isolation Forest

Samotná technika je založená na rozhodovacích stromech (angl. Decision Trees), které jsou založeny na rekurzivním zanořování a větvení náhodných částí datové sady. Cílem této techniky učení bez učitele je zanořit data podle vhodného dělení co možná nejhlouběji. Způsob dělení záleží na nastavení patřičné hodnoty, podle které se pak data člení doleva nebo doprava. Tímto způsobem se běžná data dostanou hlouběji a data obsahující odlehlé hodnoty se časem sami izolují. [92]

4 PYTHON KNIHOVNY A BALÍČKY PRO DATOVOU ANALÝZU

Pro samotnou analýzu a zpracování dat existuje nespočet různých knihoven a balíčků. Mezi některé z nich je možné zařadit ty, které jsou vypsání v následujících podkapitolách.

4.1 Numpy

Tento balíček, který byl vytvořen v roce 2005 a je volně dostupný, obecně slouží jako podpora jazyka Python pro různé matematické a vědecké výpočty. V rámci datové analýzy se nejčastěji používá pro rychlou a efektivní práci s n-dimenzionálními poli. [93]

4.2 Pandas

Vývoj této knihovny začal v roce 2008. Jedná se o knihovnu, která slouží zejména pro práci s tabulkovými daty, do kterých umí data zapisovat nebo je načítat z různých zdrojů, které zároveň umí i spojovat dohromady. Používá se také k samotným úpravám tabulek, které zahrnují čištění, agregování dat a vytváření souhrnů, změnu tvaru nebo zobrazování menších úseků v rámci práce s velkými datovými sadami. Pak je zde využití i čistě pro časové řady, u kterých jsou dostupné funkce jako je třeba posunování v čase. Tuto knihovnu však lze použít i k vizualizování dat. [94] [95]

4.3 Matplotlib

Tato grafická knihovna slouží především k jednoduché vizualizaci, která je uživatelsky přívětivá k vykreslování četných druhů grafů. Sama knihovna je postavena na základech již zmíněné knihovny Numpy. Knihovna byla vytvořena již v roce 2003. [96]

4.4 Seaborn

Seaborn je grafická knihovna pro jednoduché i komplexní vizualizace stavící na základech knihovny Matplotlib, která je charakteristická svou uživatelskou přívětivostí pro jednoduché úpravy grafů. Oproti Matplotlibu zde Seaborn vytvořena tak, aby poskytovala přehledné rozhraní pro vytváření statistických grafů a dobře spolupracovala s knihovnou Pandas. [97]

4.5 Plotly

Jedná se o další volně dostupnou knihovnu pro vizualizaci, která je oproti ostatním obohacena o možnost interakce, takže je možné si v grafech zobrazit jednotlivá místa více dopodrobna. [98]

4.6 Scipy

Knihovna Scipy, která je postavena na matematické knihovně Numpy, je dalším rozšířením jazyka Python o vestavěné možnosti, jakými lze řešit matematické rovnice a algoritmy. [99]

4.7 Scikit-learn

V této práci se jedná o velmi důležitou knihovnu, která obsahuje velké množství algoritmů strojového učení, které jsou typu učení s učitelem i typu učení bez učitele. Tato knihovna je postavena na třech dalších knihovnách, které zahrnují Numpy, Scipy a Matplotlib. Lze zde nalézt algoritmy pro regrese, klasifikace, shlukovou analýzu nebo i samotné předzpracování dat. Samotný vývoj knihovny běží od roku 2007. [100]

4.8 Statsmodels

Tato knihovna slouží k analyzování různých statistických modelů. Jedná se opět o knihovnu, která je založena na Numpy a Scipy. Zahrnuje různé statistické testy, modely lineárních regresí a způsoby, jakými lze analyzovat samotné časové řady. Kromě toho obsahuje i příklady datových sad, na kterých je možné vyzkoušet si samotnou datovou analýzu. [101]

4.9 Pmdarima

Tento balíček obsahuje spoustu různých pomůcek pro analýzu časových řad, mezi které je možné zahrnout statistické testy pro kontrolu sezonosti a stacionarity, ale zejména také automatické vyhledávání parametrů p a q pro modely, jako je třeba ARIMA. [102]

4.10 Tensor Flow

Jedná se o volně dostupnou knihovnu pro náročné matematické výpočty a strojové učení. Samotná knihovna byla vytvořena roku 2015 společností Google. Zahrnuje v sobě zejména různé modely hlubokých neuronových sítí, které je možné trénovat, testovat a dále s nimi predikovat. [103]

4.11 Keras

Jedná se o API vytvořené společností Google, které slouží pro jednoduché používaní modelů a algoritmů neuronových sítí. Keras jakožto API disponuje možností vybrat si i back end pro další výpočty. Tento back end mimochodem zahrnuje i možnost využití Tensor Flow a jiných jemu podobných knihoven. Pokud se jedná o knihovnu Tensor Flow, tak ta je zde přímo

vestavěná v rámci Keras API. Keras se ani tak moc nevyznačuje svou rychlostí jakožto spíše svou uživatelskou přívětivostí. [104]

4.12 PyTorch

Populární knihovna PyTorch byla vyvinuta společností Facebook v roce 2017. Jedná se o knihovnu podporující různé algoritmy a modely hlubokého učení, u kterých umožňuje velmi rychlý proces vytvoření. Z předchozích informací tedy vyplývá, že se zřejmě jedná o konkurenci knihoven jako je Tensor Flow od společnosti Google. [105]

4.13 XGBoost

XGBoost je balíček, který je zaměřený zejména na prediktivní úlohy, jako jsou regrese nebo klasifikace. Jedná se o velmi efektivní způsob implementace algoritmu strojového učení, který si obecně velmi dobře vede na velkém počtu různorodých datových sad, pro které je potřeba udělat predikce. [106]

PRAKTICKÁ ČÁST

5 VYUŽITÉ DATOVÉ SADY

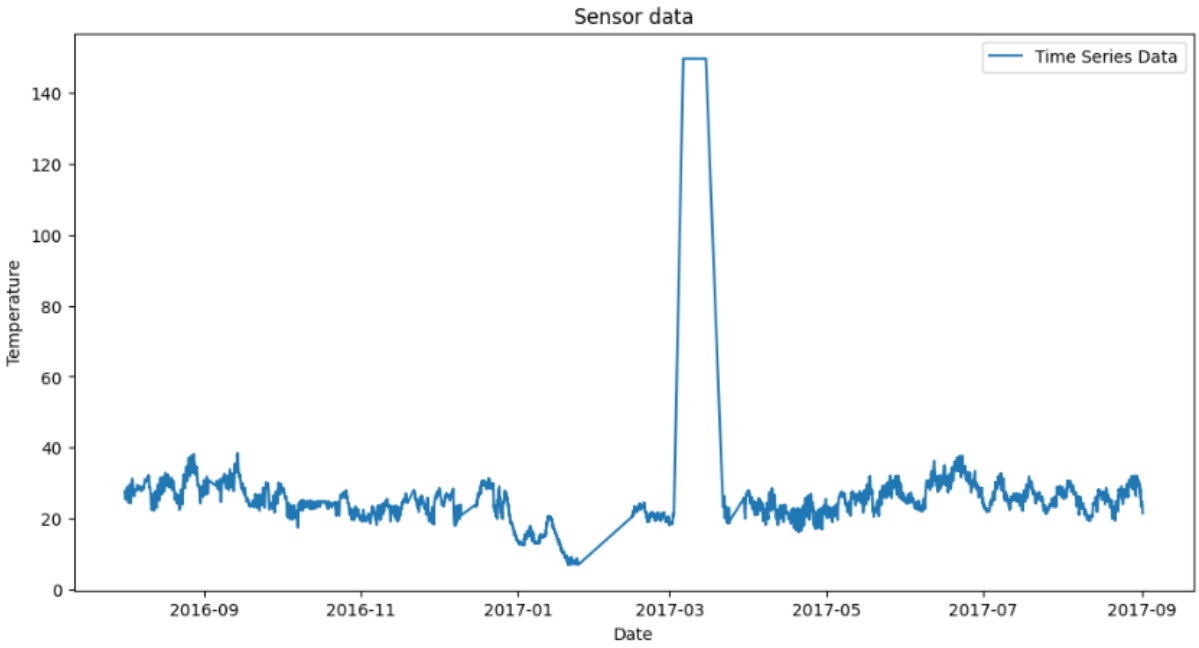
Jednotlivé typové úlohy se často hodí pro různé specifické účely nebo jsou svou povahou typické pro určité odvětví. Z toho také vychází, že se některé úlohy v těchto sektorech mohou vyskytovat frekventovaněji. Z toho důvodu byly pro jednotlivé typové úlohy vybrány datové sady, které pochází právě z různých prostředí a zdrojů, aby bylo zřetelně vidět využití těchto úloh na reálných datech. Součástí tohoto seskupení datových sad jsou zejména data z průmyslu, obchodního sektoru, energetiky a meteorologie. Samotné datové sady jsou menší velikosti, aby bylo možné typové úlohy demonstrovat v relativně krátkém časovém intervalu. Detailněji jsou využité datové sady popsány v následujících podkapitolách.

5.1 Průmyslová data z teplotních senzorů

Senzory existují z mnoha důvodů. Jedním z nich je monitorování provozních teplot, které se týkají nějakého výrobního zařízení. Senzor je zde z větší části využit pro monitorování průběhu teploty, aby bylo možné včas reagovat na případné nežádoucí chování výrobního stroje, které může zahrnovat přehřátí nebo i poškození samotného teplotního čidla. V rámci přehřátí se může jednat o extrémně vysoké teploty, které jsou signifikantně odlišné od všech normálních teplot. V případě výpadku senzoru se zde mohou např. objevit atypické teploty, které pro tyto senzory nejsou reálné. Tato datová sada schválně obsahuje několik záznamů, které jsou zaručeně globální odlehlé hodnoty způsobené nežádoucím chováním senzoru. Zároveň obsahuje i záznamy vykazující známky lokálních odlehlých hodnot, které je dále velmi vhodné analyzovat, aby se zjistilo, zda jsou tyto hodnoty normální či ne. Samotná datová sada pochází z webové stránky Kaggle, která je plná podobných volně dostupných datových sad. Konkrétně je tato datová sada velká 2,45 MB a obsahuje celkem 62 629 hodnot zaznamenané provozní teploty senzoru zařízení z výrobního prostředí za období od 1. 8. 2016 do 1. 9. 2019. Datová sada je ve formátu CSV. Sada mimo jiné obsahuje i číslo senzoru, které se však nemění, a proto není relevantní. Pokud pominu fakt, že se zde vykytuje i tento identifikátor senzoru, tak je možné data této sady považovat za data jednorozměrné. Další podrobnosti a konkrétní statistické vlastnosti jsou k vidění na obrázku č. 18, ze kterého lze jasně usoudit, že datová sada opravdu obsahuje maximální hodnoty, které se velmi liší od průměrné hodnoty časové řady. Z tohoto důvodu bude tato datová sada dále analyzována v typové úloze na detekci odlehlých hodnot skrze několik algoritmů a technik. Samotné vykreslení časové řady je poté k dispozici na obrázku č. 19. [107]

	SensorId	Value
count	62629.0	62629.000000
mean	1.0	24.203861
std	0.0	5.411599
min	1.0	6.886155
25%	1.0	21.369419
50%	1.0	24.550188
75%	1.0	27.443794
max	1.0	149.601822

Obrázek č. 18 – Statistické vlastnosti časové řady pro teplotní senzory [107]



Obrázek č. 19 – Vykreslení časové řady pro teplotní senzory [107]

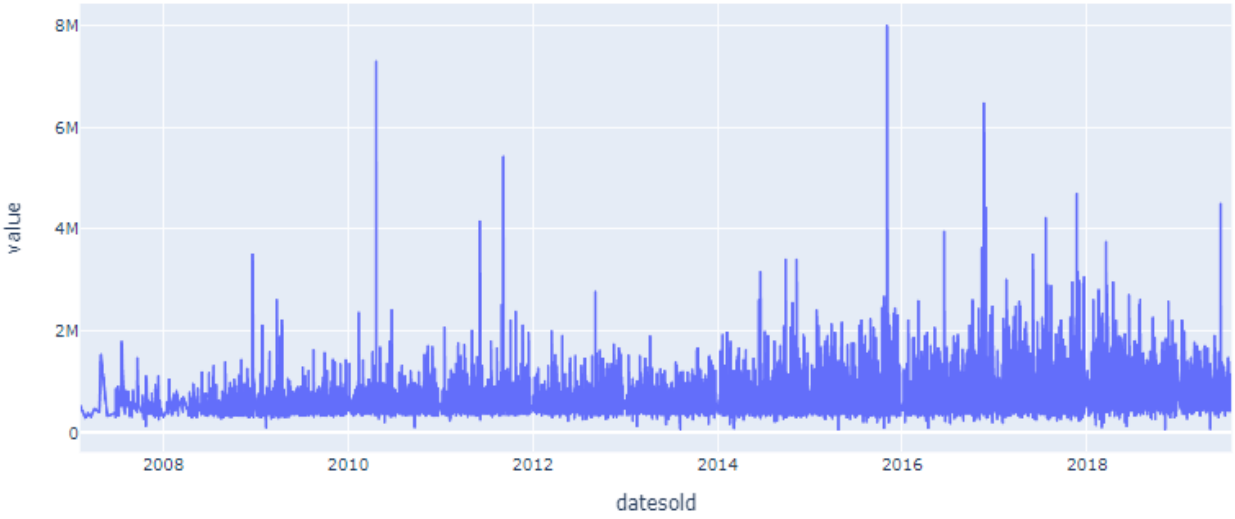
5.2 Obchodní data prodeje domů a bytů

Jedná se o datovou sadu vícerozměrné časové řady, která obsahuje záznamy o prodeji domů a bytů. V datové sadě jsou přítomné sloupce pro cenu, lokaci prodeje, počty místností určených ke spaní a klasifikování záznamu jako prodej bytu či domu. Počet místností se pohybuje od 0 do 5. Datová sada obsahuje celkem 29 580 záznamů a zabírá 1,43 MB. Stejně jako obě předchozí datové sady i tato pochází z webové stránky Kaggle a je také ve formátu CSV. Datová sada bude využita v prediktivní typové úloze pro klasifikaci časových záznamů do

dvou skupin podle počtu pokojů a ceny. Obdobně jako u předchozích datových sad je i zde souhrn statistických vlastností zobrazený na obrázku č. 20. Zobrazení průběžné ceny prodaných bytů a domů je poté znázorněno na obrázku číslo č. 21 pomocí knihovny Pandas. [108]

	postcode	price	bedrooms
count	29580.000000	2.958000e+04	29580.000000
mean	2730.249730	6.097363e+05	3.250169
std	146.717292	2.817079e+05	0.951275
min	2600.000000	5.650000e+04	0.000000
25%	2607.000000	4.400000e+05	3.000000
50%	2615.000000	5.500000e+05	3.000000
75%	2905.000000	7.050000e+05	4.000000
max	2914.000000	8.000000e+06	5.000000

Obrázek č. 20 – Statistické vlastnosti časové řady pro prodej domů a bytů [108]



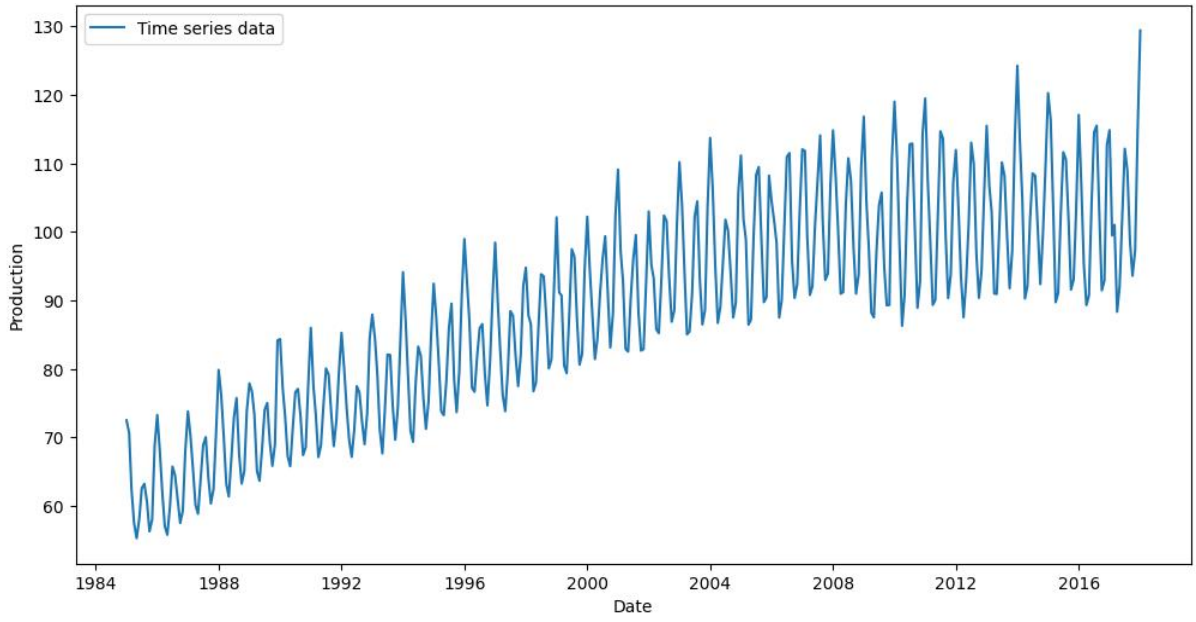
Obrázek č. 21 – Vykreslení časové řady pro prodej domů a bytů [108]

5.3 Energetická data produkce energie

Tato datová sada obsahuje data o produkci energie za období od 1.1.1985 do 1.8.2017. Celkový počet záznamů činí 397 hodnot zaznamenaných po jednotlivých měsících vždy k prvnímu dnu daného měsíce. Celková velikost sady je jen pouhých 8 kB. Původ datové sady je rovněž ze stránky Kaggle a je také ve formátu CSV. Datová sada neobsahuje vyjma hodnoty produkované energie žádné další sloupce, a proto se jedná o další případ jednorozměrné časové řady. Tato datová sada bude využita v typové úloze pro predikci výroby energie na několik nadcházejících měsíců, která by např. mohla vést k optimalizaci výrobních procesů v energetickém sektoru. Statistické vlastnosti jsou k vidění na obrázku č. 20. Samotné vykreslení časové řady je poté zobrazeno na obrázku č. 21. [109]

	Production
count	397.000000
mean	88.847218
std	15.387834
min	55.315100
25%	77.105200
50%	89.779500
75%	100.524400
max	129.404800

Obrázek č. 22 – Statistické vlastnosti pro výrobu energie [109]



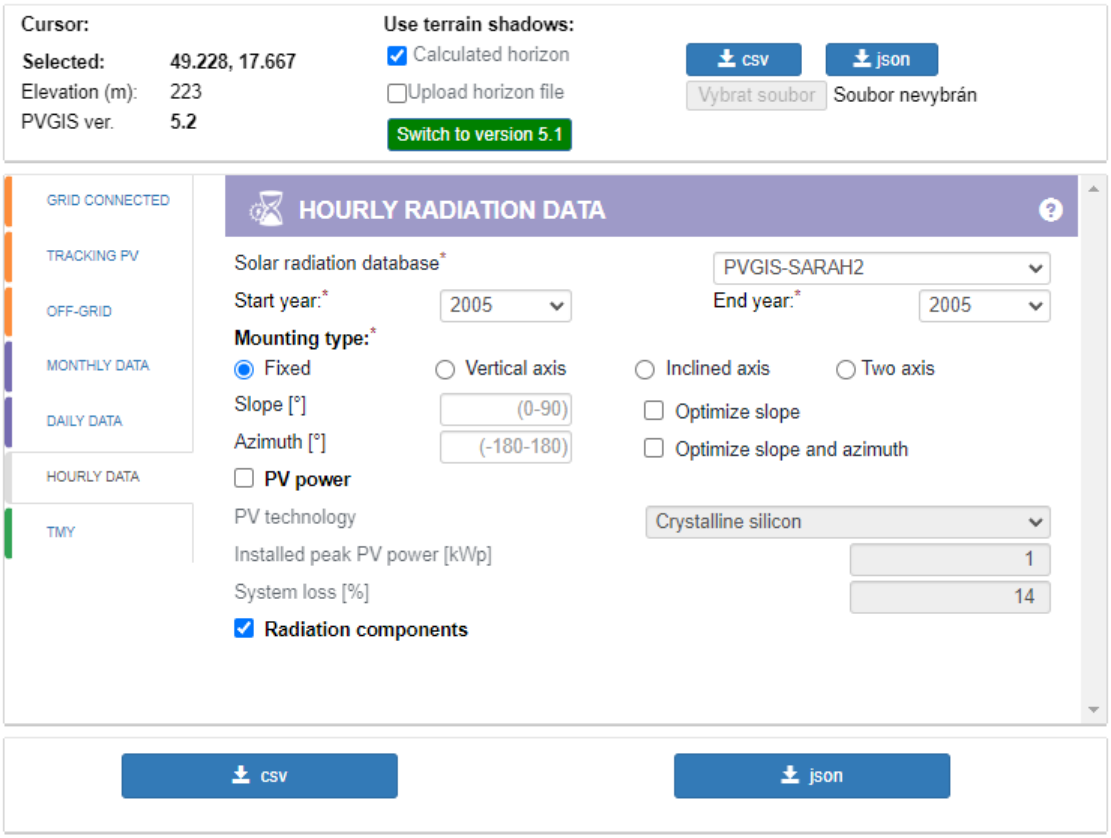
Obrázek č. 23 – Vykreslení časové řady pro výrobu energie [109]

5.4 Meteorologická data pro město Zlín

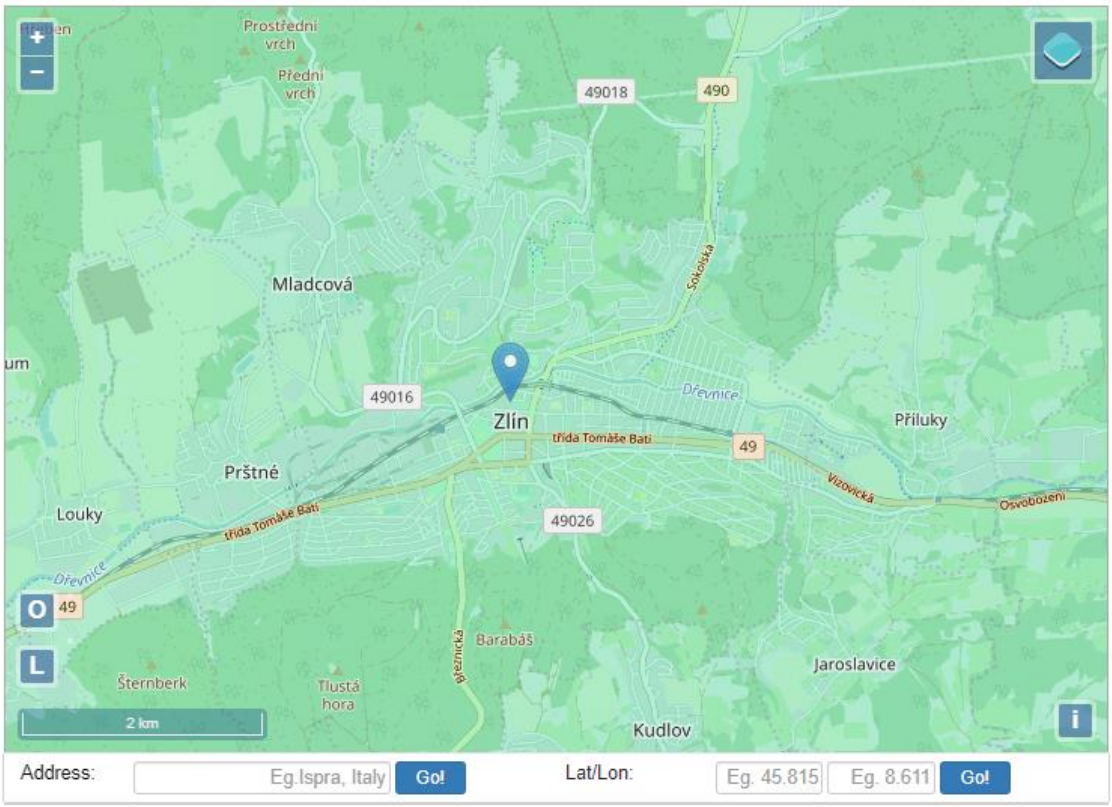
Pro oblast meteorologie je zde datová sada přímo z města Zlín, která se od předchozích liší zejména zdrojem, ze kterého pochází. Samotné data byly získána z fotovoltaického geografického informačního systému (angl. Photovoltaic Geographical Information System, dále jen PGIS), který umožňuje čerpat meteorologická data z různých lokací skoro po celém světě, které zahrnují např. data o osvitu, rychlosti větru nebo teplotě. Tyto data je možné získat za různé časové intervaly, kdy mohou být data zaznamenávána po hodinách, dnech nebo měsících. Konkrétně pro město Zlín byly vybrána data zaznamenaná po hodinách v průběhu jednoho roku. Pro tuto sadu byly vybrány právě sloupce, jako jsou již zmíněné osvity, rychlosti větru a teploty. Z toho důvodu se zde podobně jako u předchozí příkladu jedná o vícerozměrnou časovou řadu. Celá datová sada o velikosti 420 kB byla poté vyexportována do formátu CSV. Záznamy o počasí poslouží v typové úloze pro shlukovou analýzu, kde se budou shlukovat jednotlivé dny do klastrů podle toho, jestli byly spíše studené nebo teplé. [110] [111]

	Gb(i)	Gd(i)	Gr(i)	H_sun	T2m	WS10m
count	8784.000000	8784.000000	8784.000000	8784.00000	8784.000000	8784.000000
mean	96.841930	63.322110	3.336248	13.31938	9.853973	2.676358
std	206.301424	91.571914	5.459347	17.99945	7.756691	1.429691
min	0.000000	0.000000	0.000000	0.00000	-7.480000	0.000000
25%	0.000000	0.000000	0.000000	0.00000	3.177500	1.660000
50%	0.000000	0.000000	0.000000	0.00000	9.420000	2.410000
75%	47.772500	113.340000	4.490000	23.53250	16.090000	3.520000
max	1000.680000	415.930000	22.690000	63.92000	29.530000	8.760000

Obrázek č. 24 – Statistické vlastnosti časové řady pro počasí ve Zlíně [111]



Obrázek č. 25 – Ukázka nastavení systému PGIS [111]



Obrázek č. 26 – Ukázka vyhledávání na mapě pomocí systému PGIS [111]

6 IMPLEMENTACE TYPOVÝCH ÚLOH

Každá typová úloha vyžaduje několik úkonů. Některé úkony se však budou často opakovat, a proto budou popsány přímo v této kapitole. Jedná se zejména o funkce pro načtení dat a zobrazování dat. Základní příkaz např. pro načtení dat je zobrazen na následujícím řádku.

```
df = pd.read_csv('./Dataset_sensor_fault_detection.csv', delimiter=';')
```

Příkazů pro výpis dat je hned několik. Patří mezi ně příkazy pro vypsání určitého počtu záznamů, výpis statistických vlastností nebo získání obecného přehledu o datové sadě. Tyto příkazy jsou uvedeny v následujících řádcích.

```
df.info()
df.describe()
df.head()
```

Dále se zde často objevují příkazy pro úpravu dat v tabulkách, které zahrnují funkce pro přejmenování sloupců, třídění dat, formátování, práci s rozměry tabulky nebo mazání dat samotných sloupců. Některé z nich jsou k vidění níže.

```
df.rename()
df.sort_index()
pd.to_datetime()
df.values.reshape()
df.drop()
```

Dále je třeba uvést i základní knihovny, které se zaručeně objevují ve všech datových sadách. Řadíme mezi ně knihovny na následujících řádcích.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

6.1 Detekce odlehlých hodnot

Prvním krokem pro tuto úlohu je importování potřebných knihoven. Základní knihovny, které budou potřeba, jsou Numpy, Pandas, Matplotlib a Seaborn pro vykreslení dat. Další potřebná knihovna je potom Scikit-learn, ze které se budou používat modely pro detekci odlehlých hodnot. Všechny potřebné importy, které budou v této typové úloze využívány, jsou k vidění na následujících řádcích.

```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import OneClassSVM
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.decomposition import PCA
```

Dalším krokem je načtení datové sady, který se provede následujícím způsobem, kdy je potřeba zvolit cestu k souboru a způsob pro oddělování dat, aby se sada správně naformátovala.

```
df = pd.read_csv('./Dataset_sensor_fault_detection.csv', delimiter=';')
```

Po načtení následuje zobrazení dat, kdy si tato data zobrazím v tabulkovém provedení pomocí následujících příkazů. Informace o datech si následně zobrazím pomocí příkazů zmíněných v hlavní kapitole. Data si poté ještě vykreslím skrze spojnicový graf, abych získal základní přehled o tvaru časové řady. Samotný graf je vykreslen přímo na obrázku č. 19.

```
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x=df.index, y='Temp', label='Time Series Data')
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.legend()
plt.title('Sensor data')
plt.show()
```

V rámci této typové úlohy je implementován OCSVM, Isolation Forest, LOF, PAC a IQR,

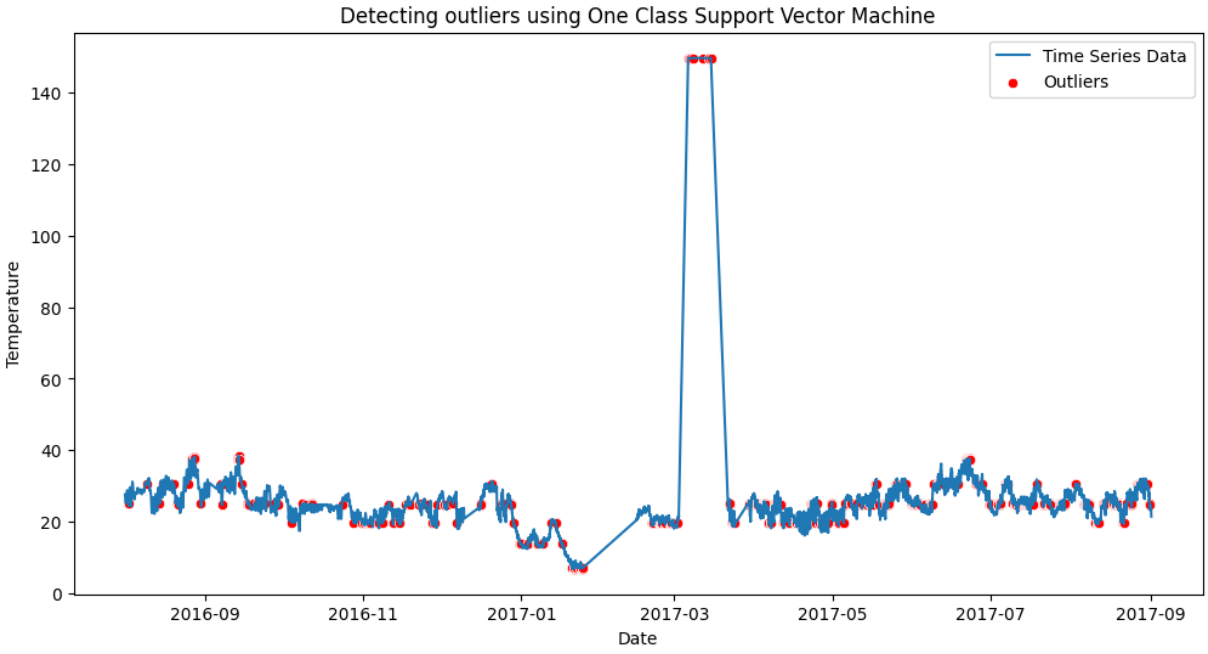
6.1.1 Příklad č. 1 – OCSVM

V rámci prvního příkladu se zde jedná o algoritmus One Class Support Vector Machines. V následujícím kódu se nejprve upraví tvar samotné tabulky a uloží se do proměnné X. Poté dojde k vytvoření modelu, u kterého se nastaví parametr *nu*, který označuje maximální hranici počtu odlehlých hodnot. V dalším řádku se modelu předloží data z upravené tabulky. Následuje samotná predikce odlehlých hodnot. Tyto hodnoty je následně nutné vyfiltrovat a uložit si je do další tabulky, která bude sloužit pro vykreslení odlehlých hodnot v původním grafu. Filtrování probíhá tak, že odlehlé hodnoty jsou označeny hodnotou -1. Samotný počet odlehlých hodnot si ještě dodatečně zobrazím v procentech, abych věděl, jakou část v datové sadě zabírají.

```
X = df.values.reshape(-1, 1)
OSVM_model = OneClassSVM(nu=0.01)
OSVM_model.fit(X)
y_pred = OSVM_model.predict(X)
df_outliers = df[y_pred == -1]
print('Percent of outliers: ' + str(round(len(df_outliers)/len(df) * 100,
2)) + '%')
```

V další fázi se data původního grafu a zjištěné odlehlé hodnoty zobrazí v jednom grafu, aby bylo vidět, kde se tyto body nachází. K vykreslení grafu posloužil kód, který je kombinací bodového a spojnicového grafu. Samotný vykreslený graf s objevenými odlehlými hodnotami je na obrázku č. 27.

```
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x=df.index, y='Temp', label='Time Series Data')
sns.scatterplot(data=outliers, x=outliers.index, y='Temp', color='red',
label='Outliers')
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.title('Detecting outliers using Isolation Forest')
plt.legend()
plt.show()
```



Obrázek č. 27 – Detekce odlehlých hodnot pomocí OSVC [107]

Z obrázku je patrné, že datová sada obsahuje několik globálních odlehlých hodnot. Zároveň je zde i spousta lokálních odlehlých hodnot. Celkový počet odlehlých hodnot zahrnuje v rámci celé datové sady 2,88 %.

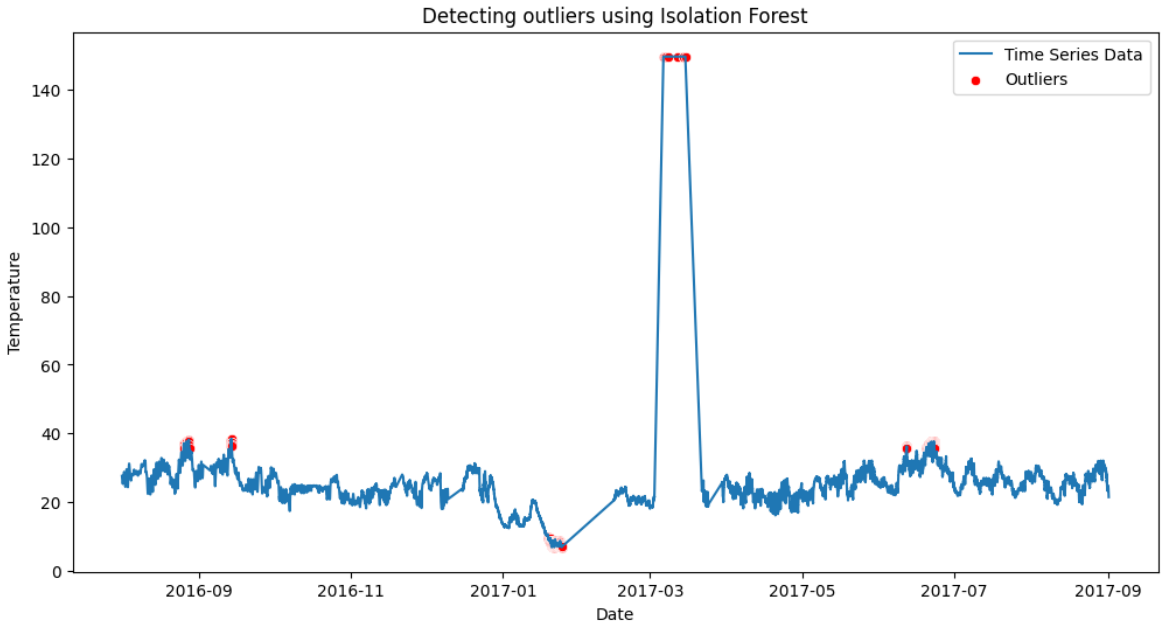
6.1.2 Příklad č. 2 – Isolation Forest

Dalším algoritmem je Isolation Forest. V první řadě je třeba vytvořit model. Do modelu zadám parametr *n_estimators*, který slouží k určení tzv. odhadců. Tento počet nastavím na 100. Dále se po nastavení modelu musí zvolit hranice pro maximální počet odlehlých hodnot, která byla nastavena na 2,5.

```
isolation_forest = IsolationForest(n_estimators=100)
isolation_forest.fit(df)
outlier_scores = isolation_forest.decision_function(df)
outlier_threshold = np.percentile(outlier_scores, 2.5)
```

V rámci Isolation Forest bylo detekováno celkem 2,38 % odlehlých hodnot.

Samotné vykreslení detekovaných hodnot je obdobně zobrazeno v následujícím grafu na obrázku č. 28.



Obrázek č. 28 – Detekce odlehlých hodnot pomocí Isolation Forest [107]

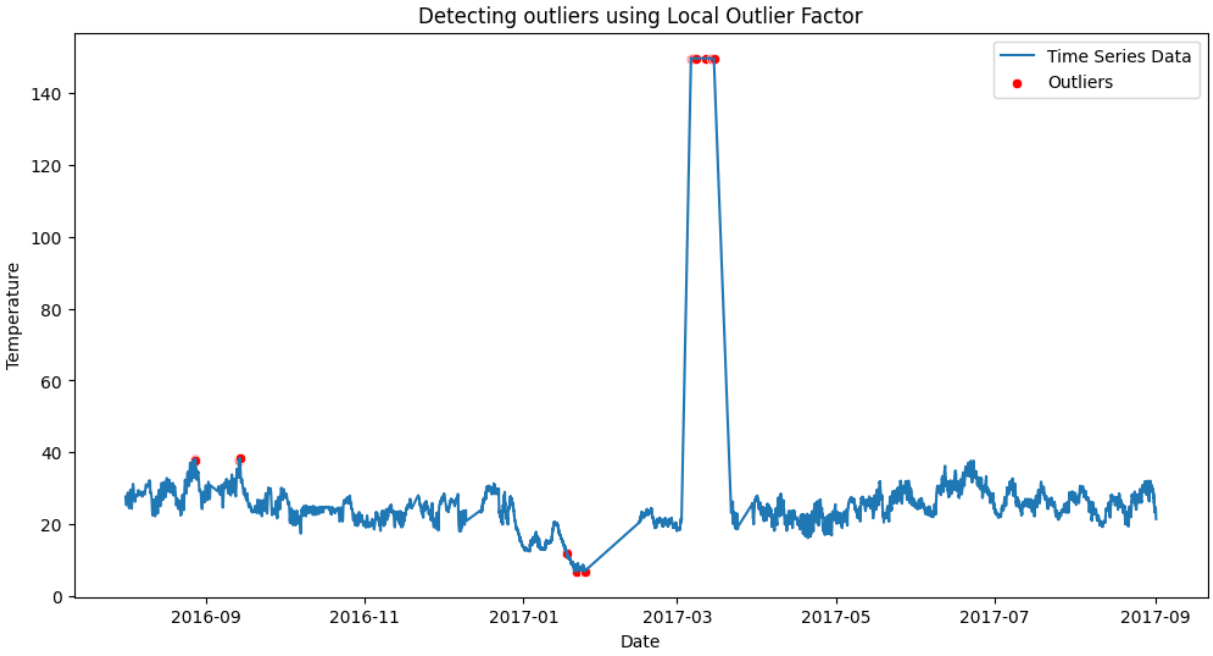
6.1.3 Příklad č. 3 – LOF

U algoritmu LOF je nejdříve potřeba změnit formát tabulky obdobně jako v prvním příkladu. Následně vytvořím LOF model, pro který specifikuji parametr *n_neighbours* pro určení počtu sousedních bodů. Tento parametr nastavím na hodnotu 10. Z toho vyplývá, že označení záznamu jako odlehlé hodnoty bude záviset právě na 10 nejbližších prvcích. Následně si vypíšu jednotlivé hodnoty s již vypočítaným skórem pro určení, zda se jedná o odlehlou hodnotu.

```
X = [[x] for x in df.Temp.values]
lof = LocalOutlierFactor(n_neighbors=10)
outliers = lof.fit_predict(X)
for i, score in enumerate(outliers):
    print(f'Data point: {df.Temp.values[i]}, Outlier Score: {score}')
```

Dále si do původní tabulky přidám sloupec s hodnotami označenými 1 a -1. Tuto modifikovanou tabulku následně obdobně vyfiltruji podle hodnoty -1, aby obsahovala jen záznamy

s odlehlými hodnotami a celý výsledek uložím do nové tabulky. Poté si už jen zobrazím procenta vyskytnutých odlehlých hodnot a vykreslím graf, který je na obrázku č. 29. Samotný výsledek detekovaných hodnot je pouhých 0,06 %.

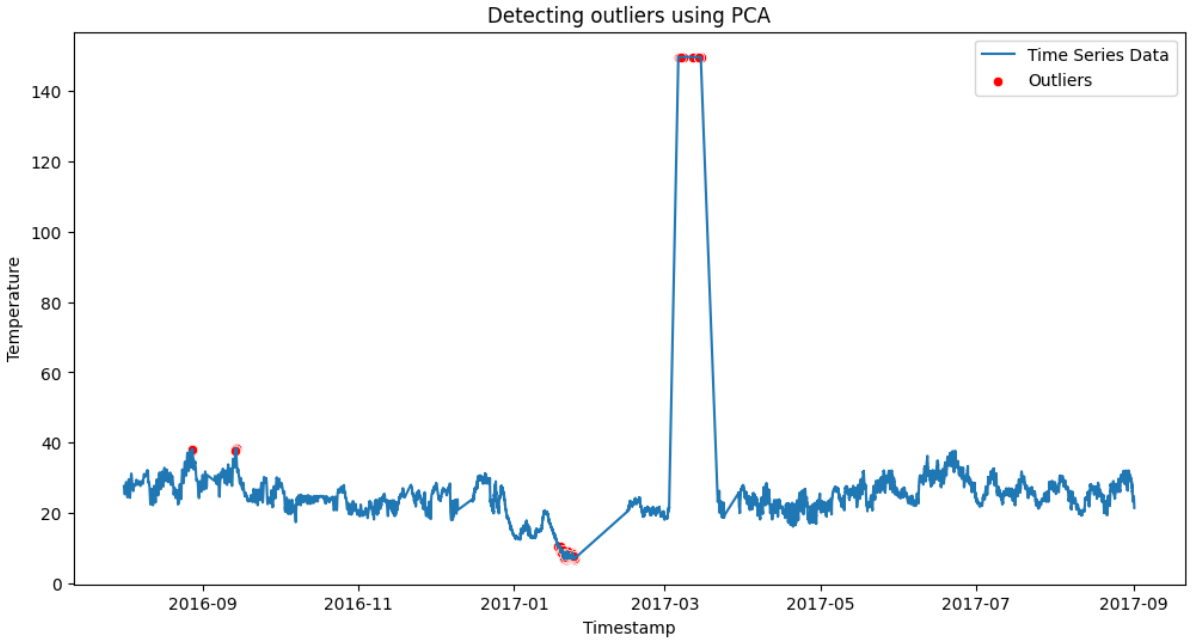


Obrázek č 29 – Detekce odlehlých hodnot pomocí LOF [107]

6.1.4 Příklad č. 4 – PCA

Pro PCA je nejdříve nutné si data standardizovat skrze StandardScaler() z knihovny Scikit-learn. Poté si vytvořím model PCA, který zahrnuje parametr *n_components*, který ponechám na hodnotě 0. Samotná data poté ještě transformuji do vhodné podoby. Dále je třeba vypočítat vzdálenost, která se měří od jednotlivých datových bodů až k samotnému prostoru PCA. Tyto vzdálenosti se poté využijí k porovnání s maximální hranicí pro počet detekovaných hodnot. Hranice byla vzhledem předchozím příkladům nastavena také na hodnotu 2,5. Hodnoty s vyšší vzdáleností jsou tedy vyfiltrovány do listu, který se využije pro vykreslení grafu s odlehlými hodnotami na obrázku č. 30

```
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df)
pca = PCA(n_components=0)
pca.fit(data_scaled)
components = pca.transform(data_scaled)
distances = np.linalg.norm(data_scaled - pca.inverse_transform(components),
axis=1)
```



Obrázek č. 30 – Detekce odlehlých hodnot pomocí PCA [107]

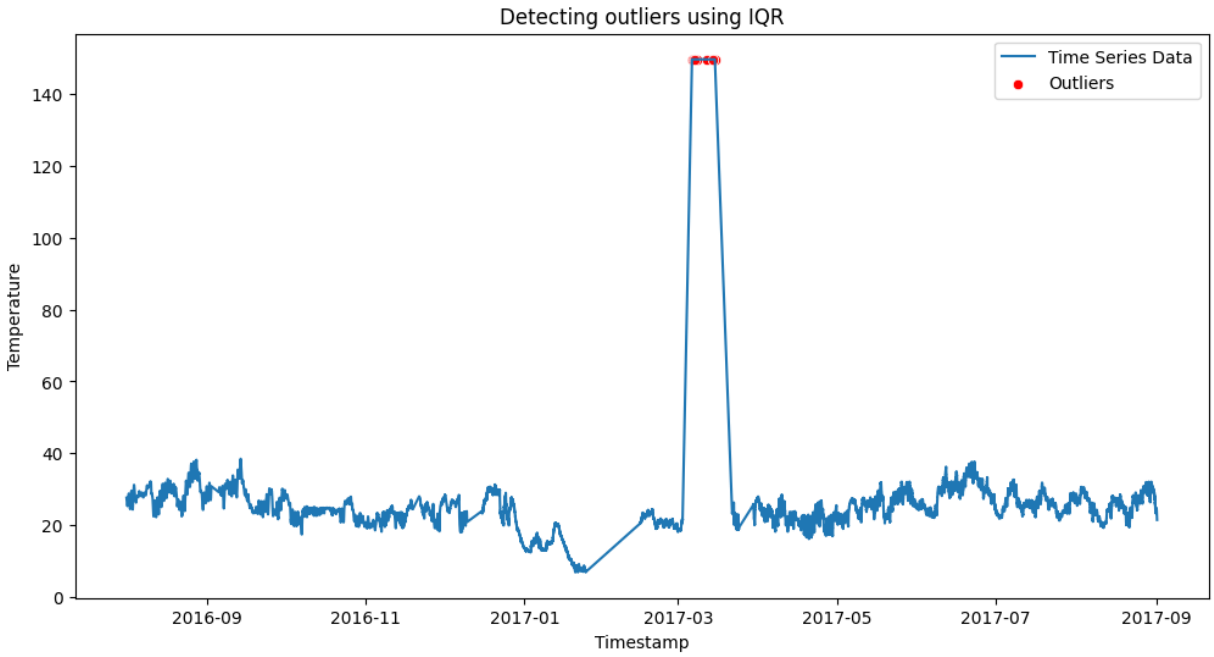
V rámci PCA došlo k detekci 2,19 % odlehlých hodnot z celé datové sady.

6.1.5 Příklad č. 5 – IQR

U této statistické metody není třeba provádět žádné speciální úkony jako změnu měřítka nebo transformace. Pouze se využije np.percentile() k získání 25 % a 75 % kvartilu. Tyto kvartily potom využiju pro výpočet prostoru, který je dán právě těmito kvartily. Nastavím si hranici pro maximální počet odlehlých hodnot a ten dále využiju pro výpočet vrchní a spodní hranice celého prostoru. Jako odlehlé hodnoty jsou následně označeny veškeré hodnoty, které se v tomto prostoru nenachází. K vyfiltrování se využije funkce np.where().

```
q1 = np.percentile(df.values, 25)
q3 = np.percentile(df.values, 75)
iqr = q3 - q1
threshold = 2.5
lower_point = q1 - threshold * iqr
upper_point = q3 + threshold * iqr
outliers = np.where((df < lower_point) | (df > upper_point))[0]
```

IQR detekovala pouze globální odlehlé hodnoty, které zahrnují jen 0,01 %. Vykreslený graf je poté k dispozici na obrázku č. 31.



Obrázek č. 31 - Detekce odlehlých hodnot pomocí IQR [107]

6.1.6 Zhodnocení

Každý z algoritmů byl schopný detekovat všechny globální odlehlé hodnoty. V rámci porovnání byly hranice maximálního počtu odlehlých hodnot některých algoritmů nastaveny schválně na stejnou úroveň a byly provedeny procentuální výpočty k zjištění celkového počtu odlehlých hodnot v datové sadě. Z uvedených je třeba zmínit IQR, který byl schopný zaznamenat pouze globální hodnoty, ale lokální hodnoty nikoliv. Tento výsledek by mohl být ovlivněn do značné míry právě nastavenou hranicí maximálního počtu odlehlých hodnot. Výsledky jsou k dispozici v tabulce č. 1.

OCSVM	Isolation Forest	LOF	PCA	IQR
2,88%	2,38%	0,06%	2,19%	0,01%

Tabulka č. 1 – Procentuální vyhodnocení detekce odlehlých hodnot [107]

6.2 Shluková analýza

Kromě základních knihoven je zde třeba importovat části z knihovny Scikit-learn, které pomohou se správným zvolením počtu klastrů.

```
from sklearn.metrics import silhouette_score
```

Dále je třeba importovat již zmíněnou funkci pro předzpracování z také knihovny Scikit-learn.

```
from sklearn.preprocessing import StandardScaler
```

Samotná datová sada je specifická tím, že byla získána z jiného zdroje. Pro získání správně naformátované tabulky je potřeba provést trochu odlišní způsob než u ostatních datových sad. Při načítání je třeba zadat parametr, který označuje počet sloupců. Způsob, jakým data správně načtu a naformátuju je na dalším řádku.

```
df          = pd.read_csv("./Dataset_Zlin_weather.csv", on_bad_lines='skip',
delimiter = ',', header = 6 )
```

Samotné data je poté potřeba taky naformátovat. S tím souvisí vyřazení NaN hodnot, vymazání nepotřebných sloupců, přejmenování, přetypování, nastavení správného formátu datumu a jeho indexu.

```
df          = df.dropna()
df          = df.drop(columns = 'Int')
df['time'] = pd.to_datetime(df['time'], format = '%Y%m%d:%H10')
df          = df.set_index('time')
df.rename(columns = {'time':'Date'}, inplace=True)
df['Gb(i)'] = df['Gb(i)'].astype(float)
```

Dále je třeba vytvořit rysy, které budou dodatečně pomáhat při samotném klastrování. Tyto rysy budou hodnoty za dny během celého roku s jejich hodinovými hodnotami pro každý sloupec. Číslování těchto rysu je provedeno od 0.

```
df['Day_of_year'] = df.index.dayofyear - 1
df['hour']        = df.index.hour
```

Data je poté třeba dále transformovat a upravit do vhodného tvaru.

```
scaler = StandardScaler()
scaler.fit(df['T2m'].values.reshape(-1,1))
```

```
df['T2m_scale'] = scaler.transform(df['T2m'].values.reshape(-1,1))
```

Dalším krokem je vytvoření tabulky podle rysů, která se skládá ze všech hodinových hodnot teplot zaznamenaných za každý jednotlivý den v roce.

```
df_pivot = df.pivot_table(index = 'Day_of_year', columns = 'hour', values =
'T2m')
```

Výsledkem je poté tabulka, která je zobrazena na následujícím obrázku č. 32.

hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
Day_of_year																					
0	0.71	0.46	0.11	-0.26	-0.63	-0.39	-0.77	-1.51	-0.32	1.28	...	2.39	1.19	0.70	0.93	0.66	0.57	-1.72	-2.34	-2.78	-2.94
1	-2.94	-2.64	-2.43	-2.76	-2.85	-2.80	-2.97	-4.76	-3.58	-0.64	...	3.17	1.31	-1.00	-2.49	-3.20	-3.51	-3.48	-3.33	-3.82	-3.86
2	-3.69	-3.75	-4.06	-4.27	-4.43	-4.55	-4.59	-3.97	-2.85	-1.68	...	0.43	-0.86	-1.48	-1.71	-1.88	-2.32	-2.31	-1.96	-1.48	-1.15
3	-1.03	-0.45	0.19	1.03	1.88	2.70	3.49	3.34	3.10	3.13	...	2.38	1.72	1.17	0.74	0.57	1.64	1.72	1.67	1.23	0.66
4	0.15	-0.23	-0.24	-0.40	-0.55	-0.68	-0.76	-0.84	-0.47	0.04	...	1.05	0.08	-0.83	-1.36	-1.98	-2.01	-2.62	-2.67	-1.71	-1.84

Obrázek č. 32 – Hodinové teploty pro každý den v roce [111]

6.2.1 Příklad č. 1 – K-Means

K-Means algoritmus je třeba importovat z modulu knihovny Scikit-learn.

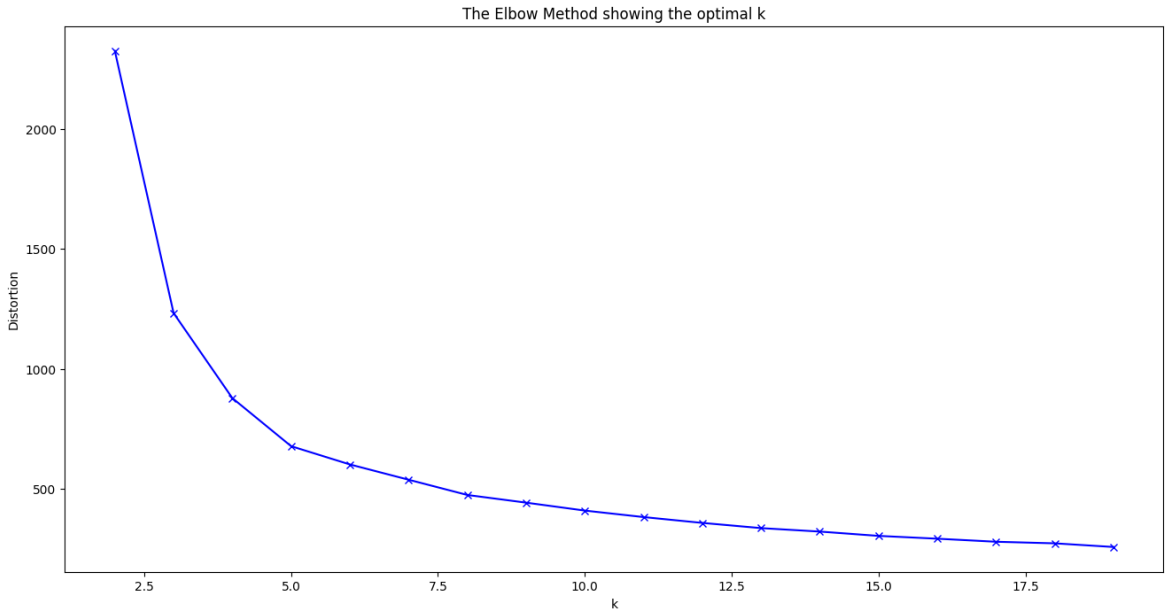
```
from sklearn.cluster import KMeans
```

První věc, která se musí udělat, je zjistit K počet klastrů. To se provede za pomoci loktového pravidla a Silhouette skóre. Jedná se o vykreslení různých počtu klastrů, u kterých je pak možné pomocí výše zmíněných dvou způsobů vybrat optimální počet klastrů.

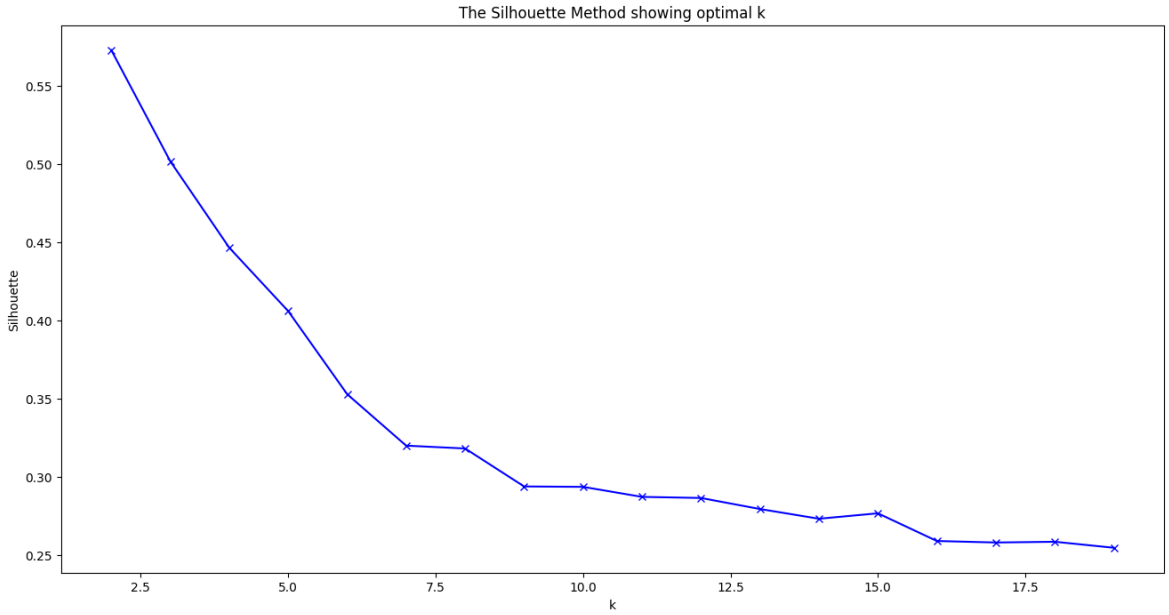
```
K = range(2,20)
distortions = []
sill = []

for k in K:
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(df_pivot_scaled.values)
    distortions.append(kmeans.inertia_)
    sill.append(silhouette_score(df_pivot_scaled.values, labels=kmeans.pre-
dict(df_pivot_scaled.values), metric='euclidean'))
```

Vykreslením získám dva grafy, které jsou ukázány na obrázku č. 33 a č. 34. Z obrázku pro loktové pravidlo není úplně jasné, který počet klastrů zvolit. V rámci Silhouette se však většinou oplatí vzít hodnotu, která dosáhla nejvyššího skóre. V tomto případě se jedná teda o 2 klastry.



Obrázek č. 33 – Loktové pravidlo pro K-Means [111]



Obrázek č. 34 – Silhouette skóre pro K-Means [111]

Po zjištění optimálního počtu klastrů je možné vytvořit samotný model, pro který teda nastavím počet na 2 klastry. Z toho vyplývá, že dny v roce se budou dělit podle teploty do dvou shluků, kdy jeden bude obsahovat teplejší dny a druhý ty studenější. Pro tyto klastry je však ještě třeba přidat sloupec, který bude obsahovat 1 a 0 podle toho, o který klastr se jedná.

```
kMeansModel = KMeans(n_clusters=2)
kMeansModel.fit(df_pivot_scaled.values)
clusters = kMeansModel.predict(df_pivot_scaled.values)
```

```
df_pivot_scaled['cluster'] = clusters
```

Samotné data je poté ještě potřeba dostat zpátky do původní podoby, kdy se aplikuje maska, která tento proces zapříčiní.

```
for day in df['Day_of_year'].values:
    mask = (df['Day_of_year'] == day)
    df.loc[mask, 'cluster'] = int(df_pivot_scaled.loc[day]['cluster'])
```

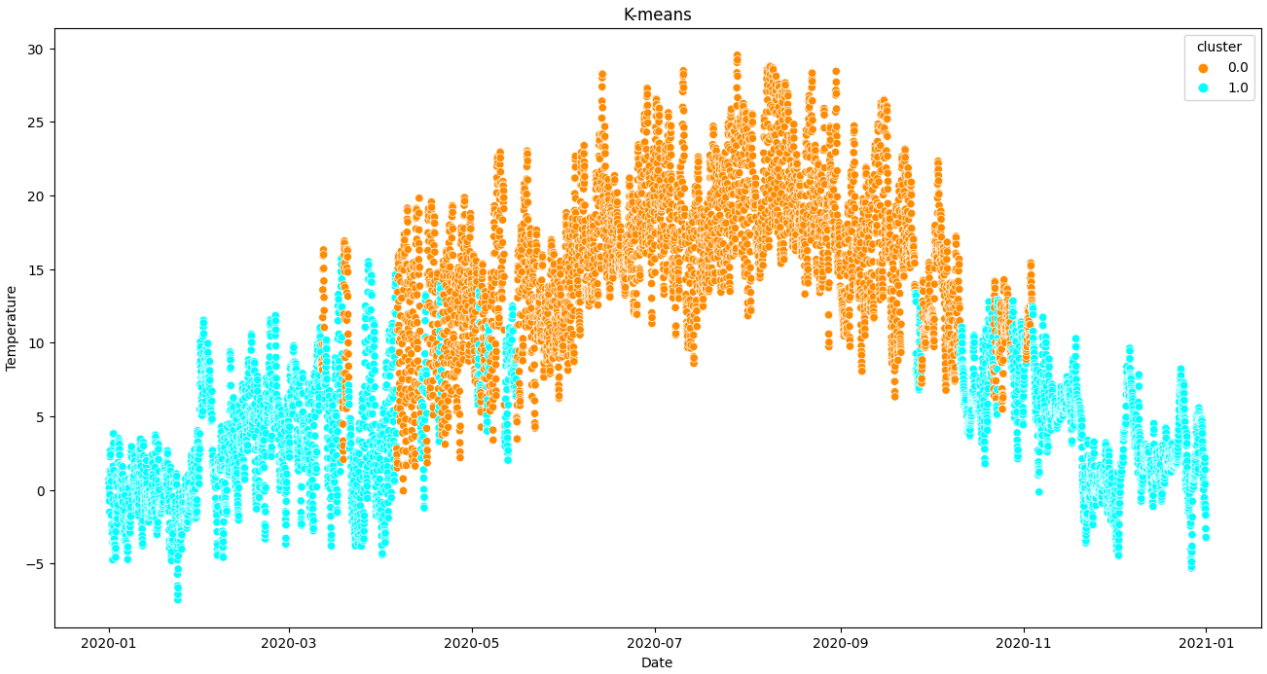
Pro zjištění a lepší porovnání jsem také vytvořil výpočet pro procentuální znázornění hodnot v každém klastru.

```
cold_days = df.cluster.loc[(df.cluster == 0)]
warm_days = df.cluster.loc[(df.cluster == 1)]

cold_days_percents = len(cold_days) / len(df.cluster) * 100
warm_days_percents = len(warm_days) / len(df.cluster) * 100

print('Cold days: ' + str(round(cold_days_percents, 2)) + '%')
print('Warm days: ' + str(round(warm_days_percents, 2)) + '%')
```

V rámci algoritmu K-Means se podařilo vytvořit dva přirozené shluky dat. Shluk dat pro teplejší dny tvoří 48,91 % zatím co ty chladnější tvoří v druhém shluku zbylých 51,09 %. Samotné rozdělení dnů na teplejší a chladnější je na obrázku č. 35.



Obrázek č. 35 – K-Means klastrování dnů na teplejší a chladnější [111]

6.2.2 Příklad č. 2 – DBSCAN

Pro implementaci algoritmu DBSCAN je třeba importovat následující.

```
from sklearn.cluster import DBSCAN
```

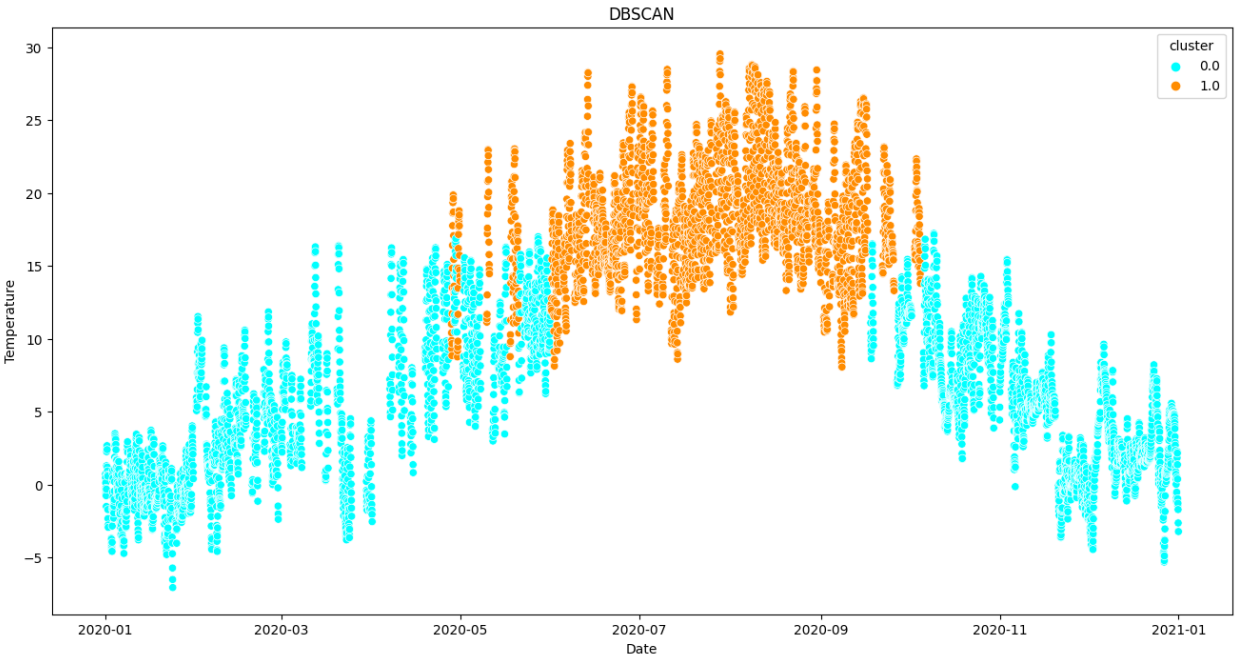
Obdobně jako u předchozího typu klastrování je i zde třeba vykreslit graf pro Silhouette skóre. V rámci DBSCANU se však nejedná o optimální počet klastrů, ale o vhodnou kombinaci parametrů *eps* a *min_samples*. Vykreslení tohoto grafu je na obrázku č. 36.

Obrázek č. 36 – Silhouette skóre pro kombinaci parametrů *eps* a *min_samples* [111]

Z obrázku je zřejmé, že nejvyšší skóre dosáhla kombinace parametrů *eps* s hodnotou 0,9 a *min_samples* s hodnotou 9. Tyto hodnoty poslouží jako vstup pro nejlepší možný model.

```
dbscan = DBSCAN(eps=eps, min_samples=min_samples)
clusters = dbscan.fit_predict(df_pivot_scaled.values)
```

Výsledkem shlukové analýzy jsou dva klastry, kdy chladnější dny zahrnují 63,05 % a teplejší dny zbylých 36,95 %. Samotný graf je pak k dispozici na obrázku č. 37.



Obrázek č. 37 – DBSCAN klastrování dnů na teplejší a chladnější [111]

6.2.3 Příklad č. 3 – GMM

V prvním kroku si importuju z modulu mixture knihovny Scikit-learn model GMM

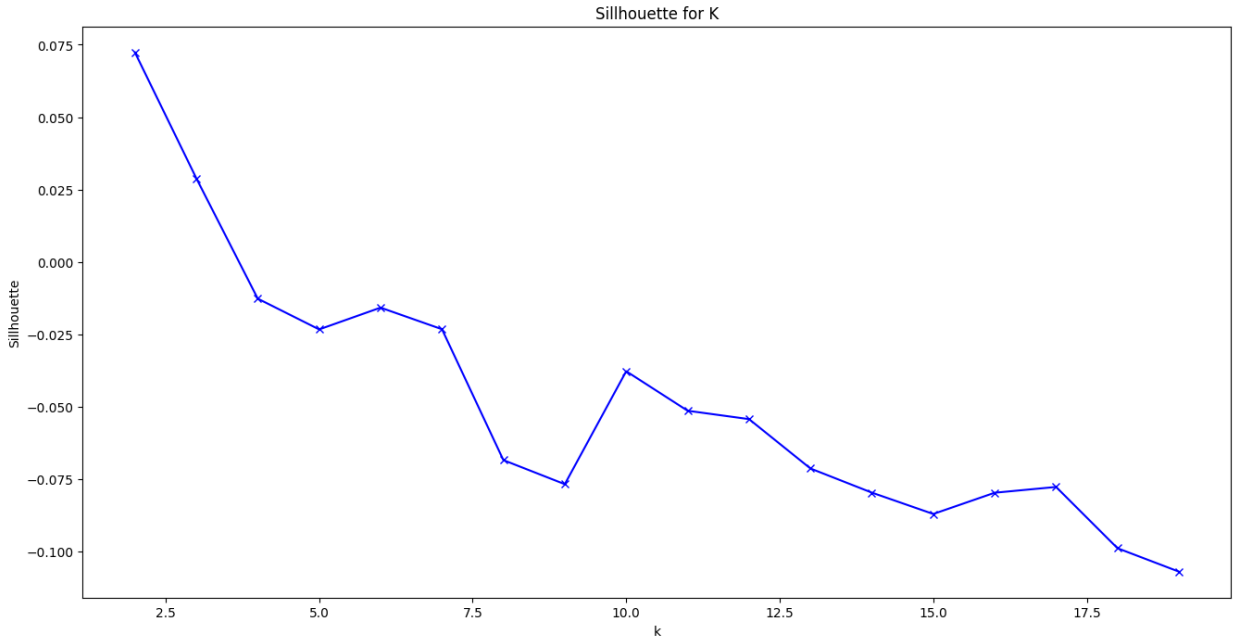
```
from sklearn.mixture import GaussianMixture
```

Dalším krokem je zjištění optimálního počtu klastrů obdobně jako v prvním příkladě. Jediný rozdíl je zde v tom, že pro GMM využiji jinou vzdálenost, kterou je Mahalanobisova. Optimální počet klastrů byl znova zvolen jako číslo 2. Graf se Silhouette skóre je zobrazeno na obrázku č. 38. Samotný graf pro GMM pak na obrázku č. 39.

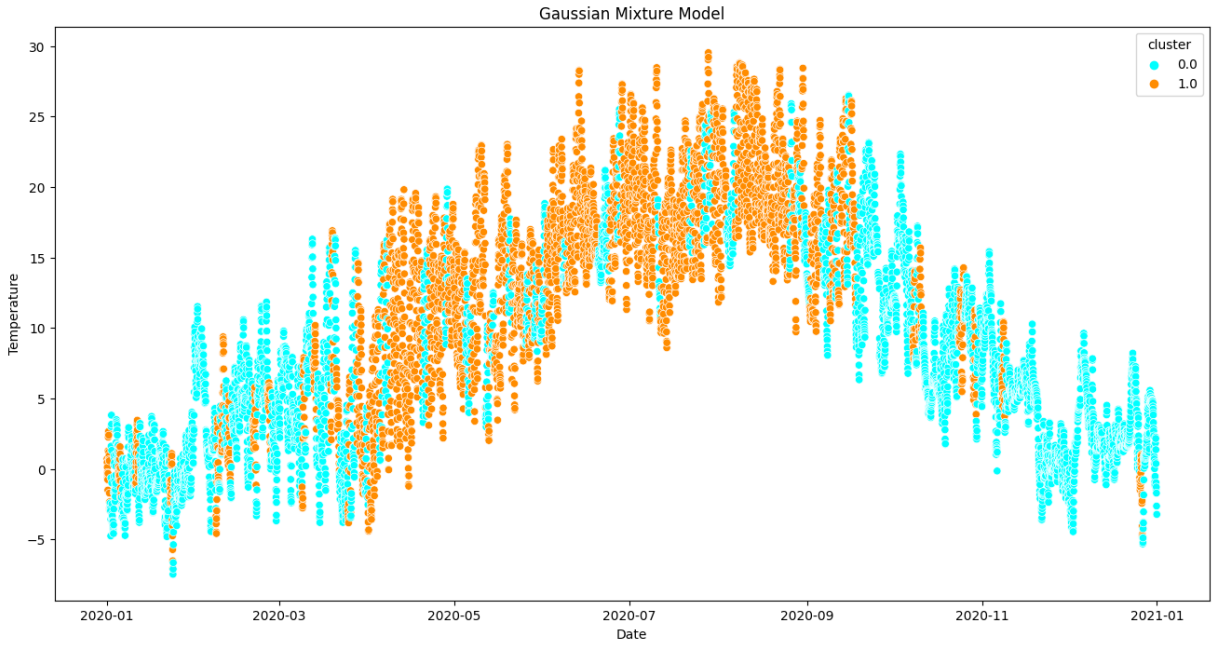
```
K     = range(2,20)
sill  = []

for k in K:
    gmm = GaussianMixture(n_components=k)
    gmm.fit(df_pivot_scaled.values)
    sill.append(silhouette_score(df_pivot_scaled.values, labels = gmm.pre-
dict(df_pivot_scaled.values), metric='mahalanobis'))
```

V rámci GMM byly dny rozděleny do shluků na spíše teplé dny zahrnujících 43,17 % a spíše chladnější dny, které tvoří zbylých 56,83 %.



Obrázek č. 38 - Silhouette skóre pro GMM [111]



Obrázek č. 39 - GMM klastrování dnů na teplejší a chladnější []

6.2.4 Zhodnocení

Z jednotlivých modelů se podobně projevily zejména K-Means a DBSCAN, které téměř striktně oddělily chladné dny od teplých. Na druhou stranu GMM zvládlo rozlišit i některé chladnější dny v letních měsících a teplejší dny v těch zimních.

	K-Means	DBSCAN	GMM
Warm days	51,09%	36,95%	43,17%
Cold days	48,91%	63,05%	56,33%

Tabulka č. 2 - Procentuální vyhodnocení klastrování teplých a studených dnů [111]

6.3 Predikce – klasifikace

Stejně jako vždy je prvním krokem načtení datové sady. V tomto případě není potřeba specifikovat žádné další parametry jako třeba oddělovače nebo počty sloupců.

```
df = pd.read_csv("./Dataset_sales.csv")
```

Dále si obdobným způsobem jako v minulých příkladech upravím tabulku. Tyto úpravy zahrnují zvolení indexu a nastavení správného formátu pro datum.

```
df.index          = pd.to_datetime(df['datesold'])
```

Poté si vytvořím dodatečné vlastnosti dat, které klasifikace využije pro lepší zařazení, protože prodeje domů nejsou stejné pro každý den. Tyto vlastnosti teda zahrnují roky, měsíce, dny v týdnu a také to, jestli je zrovna víkend. Výsledkem je tabulka na obrázku č. 40.

```
df['year']        = df.index.year
df['month']       = df.index.month
df['day_of_week'] = df.index.day_of_week
df['is_weekend']  = (df['day_of_week'] >= 5).astype(int)
```

	datesold	postcode	price	propertyType	bedrooms	year	month	day_of_week	is_weekend
datesold									
2007-02-07	2007-02-07 00:00:00	2607	525000	house	4	2007	2	2	0
2007-02-27	2007-02-27 00:00:00	2906	290000	house	3	2007	2	1	0
2007-03-07	2007-03-07 00:00:00	2905	328000	house	3	2007	3	2	0
2007-03-09	2007-03-09 00:00:00	2905	380000	house	4	2007	3	4	0
2007-03-21	2007-03-21 00:00:00	2906	310000	house	3	2007	3	2	0
...	...	...	...	...	...	...	...	...	...
2019-07-25	2019-07-25 00:00:00	2620	172500	unit	1	2019	7	3	0
2019-07-25	2019-07-25 00:00:00	2603	297500	unit	2	2019	7	3	0
2019-07-25	2019-07-25 00:00:00	2606	321000	unit	1	2019	7	3	0
2019-07-25	2019-07-25 00:00:00	2603	380000	unit	1	2019	7	3	0
2019-07-25	2019-07-25 00:00:00	2612	475000	unit	2	2019	7	3	0

Obrázek č. 40 – Tabulka obohacená o dodatečné vlastnosti [108]

V rámci tabulky jsou k vidění informace, jestli se jedná o byt či dům. Tyto informace je dále třeba převést do numerické podoby, aby se mohly používat ke klasifikaci.

```
le = preprocessing.LabelEncoder()
le.fit(df['propertyType'].values)
df['property_type_encoded'] = le.transform(df['propertyType'].values)
```

Předchozí tabulka obsahuje sloupec s vlastnostmi, které jsou plné různých hodnot. Následně si vytvořím novou verzi této tabulky, kdy jednotlivé hodnoty vlastností budou brány jako sloupce jejichž hodnoty budou nabývat 1 nebo 0 podle toho, zda je vlastnost přítomna nebo nikoli.

```
df_dummed = pd.concat([pd.get_dummies(df['bedrooms']), pd.get_dum-
mies(df['postcode']), pd.get_dummies(df['is_weekend']), pd.get_dum-
mies(df['year']) ,df['price'], df['property_type_encoded']], axis = 1)
```

Tato tabulka by však mohla být nepřehledná, protože některé sloupce by se pak mohly jmenovat stejně. Proto si sloupce této tabulky ještě očísluji, aby byly sloupce jednoznačně identifikovatelné. Celá tabulka je poté ještě normalizována od 0 do 1. Výsledná tabulka je pak k vidění na obrázku č. 41.

```
ordered_names      = df_dummed.columns.astype(str) + '_' + np.arange(df_dum-
med.shape[1]).astype(str)
df_dummed.columns = ordered_names

price_column = df_dummed.columns[df_dummed.columns.str.contains('price')][0]
df_dummed[price_column] = df_dummed[price_column]/df_dummed[price_co-
lumn].max()
```

datesold	0_0	1_1	2_2	3_3	4_4	5_5	2600_6	2601_7	2602_8	2603_9	...	2012_40	2013_41	2014_42	2015_43	2016_44	2017_45	2018_46	2019_47	price_48	property_type_encoded_49
2007-02-07	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0.065625	0
2007-02-27	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0.036250	0
2007-03-07	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0.041000	0
2007-03-09	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0.047500	0
2007-03-21	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0.038750	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2019-07-25	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0.021562	1
2019-07-25	0	0	1	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	1	0.037187	1
2019-07-25	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0.040125	1
2019-07-25	0	1	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	1	0.047500	1
2019-07-25	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0.059375	1

Obrázek č. 41 – Normalizovaná tabulka s očíslovanými vlastnostmi [108]

Posledním krokem v přípravě dat před samotnou aplikací algoritmů je rozdělení dat na trénovací a testovací množinu.

```
X_train, X_test, y_train, y_test =
train_test_split(df_dummed
[df_dummed.columns[0 : df_dummed.shape[1] - 1]].values,
df_dummed['property_type_encoded' + '_'
 + str(df_dummed.shape[1] - 1)].values, test_size = 0.2)
```

V rámci implementace byly vybrány algoritmy Logistic Regression, SVN, KNN, Decision Tree, Random Forest a GNB.

6.3.1 Příklad č. 1 – Logistic Regression

Prvně si importuji z knihovny Scikit-learn modul pro logistickou regresy.

```
from sklearn.linear_model import LogisticRegression
```

Trénovací data mám připravené, takže je už je jen vložím do modelu, který tak natrénuji. Následně vyzkouším predikci na testovacích datech a vypíšu výstup z matice zmatení, která mi zobrazí počty správně a špatně zařazených záznamů pro každou jednotlivou třídu.

```
logistic_regression = LogisticRegression().fit(X_train, y_train)
logreg_prediction = logistic_regression.predict(X_test)
conf_matrix = confusion_matrix(y_test, logreg_prediction)
print(conf_matrix)
```

Samotnou matici využiji k vypočítání procentuální přesnosti zařazení do jednotlivých tříd. Zprůměrováním výsledků každé třídy zjistím i celkovou přesnost klasifikace.

```
TP = conf_matrix[0][0]
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TN = conf_matrix[1][1]

houseAccuracy = TP / (TP + FP) * 100
unitAccuracy = TN / (FN + TN) * 100
overallAccuracy = (houseAccuracy + unitAccuracy) / 2

print('House classification accuracy: ' + str(round(houseAccuracy, 2)) +
'%')
print('Unit classification accuracy: ' + str(round(unitAccuracy, 2)) + '%')
print('Overall classification accuracy: ' + str(round(overallAccuracy, 2)) +
'%')
```

Procentuálně vyšlo zařazení do skupiny domů s přesností 97,5 %. Pro byty byla poté přesnost 84,54 %. Celková přesnost klasifikace pomocí logistické regrese je 91,02 %.

6.3.2 Příklad č. 2 – SVM

Zápis kódu pro SVM je obdobný jako u přechozí varianty s logistickou regresí.

```
svm     = svm.SVC().fit(X_train, y_train)
svm_prediction = svm.predict(X_test)
```

6.3.3 Příklad č. 3 – KNN

KNN se lehce liší tím, že je třeba zadat počet sousedů v okolí pro klasifikování každého záznamu. Tento počet byl nastaven na 20.

```
knn = KNeighborsClassifier(n_neighbors = 20).fit(X_train, y_train)
knn_prediction = knn.predict(X_test)
```

6.3.4 Příklad č. 4 – Decision Tree

Zápis rozhodovacích stromů je téměř totožný s většinou variant.

```
decision_tree = tree.DecisionTreeClassifier().fit(X_train, y_train)
tree_prediction = decision_tree.predict(X_test)
```

6.3.5 Příklad č. 5 – Random Forest

Zápis náhodného lesu se liší zejména tím, že je potřeba specifikovat počet odhadců. Tento počet byl nastaven na hodnotu 100.

```
random_forest = RandomForestClassifier(n_estimators=100)
.fit(X_train, y_train)
forest_prediction = random_forest.predict(X_test)
```

6.3.6 Příklad č. 6 – GNB

Zápis GNB je totožný jako pro logistickou regresy.

```
gnb = GaussianNB().fit(X_train, y_train)
gnb_prediction = gnb.predict(X_test)
```

6.3.7 Zhodnocení

Pro přehlednost byly data matice zmatení uvedeny pro všechny případy až na samotném konci podkapitoly. Celkové shrnutí je následně v tabulce č. 3.

		Logistic Regression	SVM	KNN	Decision Tree	Random Forest	GNB
Houses	True	4759	4816	4777	4769	4781	4080
	False	122	121	113	132	120	858
Houses accuracy		97,50%	97,50%	97,69%	97,31%	97,55%	82,60%
Units	True	875	806	873	849	862	931
	False	160	173	153	166	153	47
Units accuracy		84,54%	82,33%	85,09%	83,65%	84,93%	95,19%
Overall accuracy		**91,02%**	**89,94%**	**91,39%**	**90,48%**	**91,24%**	**88,91%**

Tabulka č. 3 – Vyhodnocení klasifikace [108]

6.4 Predikce – regrese

Stejně jako u předchozí typové úlohy je i zde první krok importování potřebných knihoven. Obdobně si načtu i datovou sadu, pro kterou si zobrazím informace v tabulkách. V rámci datové sady provedu několik úprav sloupců, které jsou zobrazeny níže v kódu. Ty zahrnují přejmenování sloupců, nastavení indexů a správného formátu datumu.

```
df.rename(columns={'IPG2211A2N':'Production'}, inplace = True)
df.rename(columns={'DATE':'Date'}, inplace = True)
df['Date'] = pd.to_datetime(df['Date'])
df = df.set_index(['Date'])
```

Data si poté vhodně vykreslím. Samotný graf je k dispozici na již uvedeném obrázku č. 23.

Zejména pro srovnání jednotlivých modelů bude také potřeba importovat jednu důležitou funkci z knihovny Scikit-learn, která je zobrazena níže.

```
from sklearn.metrics import mean_squared_error
```

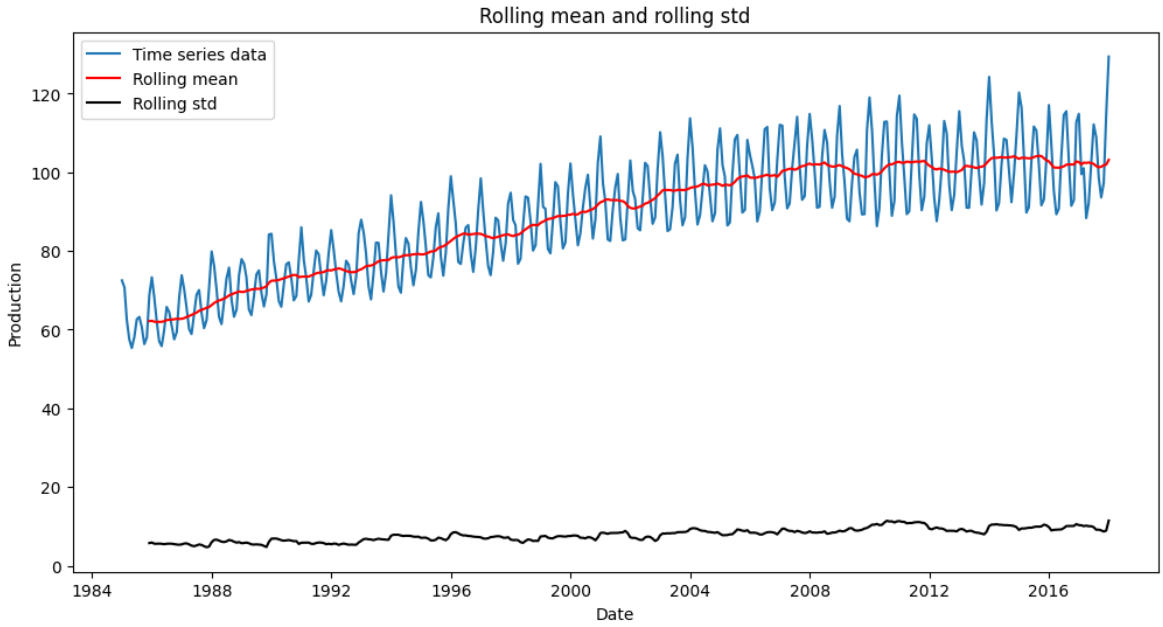
6.4.1 Příklad č. 1 – SARIMA

Čistě v rámci modelu SARIMA je zprvu nutné importovat další dodatečné knihovny a balíčky. Jedná se zejména o knihovnu Statsmodels a její součásti. Pro samotné testování parametrů modelu SARIMA bude potřeba ještě přidat i knihovnu Pmdarima.

```
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from pmdarima.arima import auto_arima
```

Dalším krokem je poté vykreslení klouzavého průměru a směrodatné odchylky, aby se jsem zjistil, jestli je třeba řadu převést na řadu stacionární. Výsledek je k vidění na obrázku č. 42.

```
roll_mean = df.rolling(window=12).mean()
roll_std = df.rolling(window=12).std()
print(roll_mean, roll_std)
```



Obrázek č. 42 – Zobrazení klouzavého průměru a klouzavé směrodatné odchylky [109]

Z obrázku č. 42 je možné usoudit, že časová řada nebude možná potřebovat diferenciování pro odstranění nestacionarity. Samotným pohledem do grafu ale toto přesvědčení nic nepotvrzuje. Proto je nutné provést Dickey-Fullerův test, který zjistí kritické hodnoty a hodnotu *p-value*.

```
adf_test_result = sm.tsa.adfuller(df)
test_statistic = adf_test_result[0]
p_value = adf_test_result[1]
critical_values = adf_test_result[4]

print("Test Statistic:", test_statistic)
print("P-value:", p_value)
print("Critical Values:")

for key, value in critical_values.items():
    print("\t{}: {}".format(key, value))
```

Samotná hodnota *p-value* indikuje, že by bylo vhodné provést diferenciování. Kritické hodnoty jsou nicméně velmi blízko testované hodnotě, a proto je potřeba využít funkcí knihovny Pmdarima, která automaticky určí nejen parametr pro diferenciování, ale také všechny další parametry. S tím také souvisí, že automaticky také zvolí nejvhodnější model.

Ještě před využitím této knihovny je třeba rozdělit si data na trénovací a testovací část.

```
split_index = int(len(df) * 0.8)
train_data = df[:split_index]
test_data = df[split_index:]
```

Poté už lze využít funkce auto_arima(), do které jako parametry vložím trénovací data, nastavení sezonosti a samotný počet sezón. Výsledkem je poté report o celém testování modelu, který zahrnuje i vstupní parametry pro nejlepší model i s jeho chybovostí. Tento report je k dispozici na obrázku č. 43.

SARIMAX Results

Dep. Variable:	y	No. Observations:	317
Model:	SARIMAX(1, 0, 2)x(0, 1, [1], 12)	Log Likelihood	-674.200
Date:	Wed, 24 May 2023	AIC	1360.401
Time:	18:46:15	BIC	1382.723
Sample:	01-01-1985 - 05-01-2011	HQIC	1369.329
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.1109	0.065	1.696	0.090	-0.017	0.239
ar.L1	0.9301	0.041	22.423	0.000	0.849	1.011
ma.L1	-0.3393	0.079	-4.315	0.000	-0.493	-0.185
ma.L2	-0.3423	0.067	-5.111	0.000	-0.474	-0.211
ma.S.L12	-0.6949	0.048	-14.534	0.000	-0.789	-0.601
sigma2	4.7444	0.286	16.581	0.000	4.184	5.305

Ljung-Box (L1) (Q):	0.05	Jarque-Bera (JB):	56.94
Prob(Q):	0.83	Prob(JB):	0.00
Heteroskedasticity (H):	2.48	Skew:	-0.32
Prob(H) (two-sided):	0.00	Kurtosis:	5.02

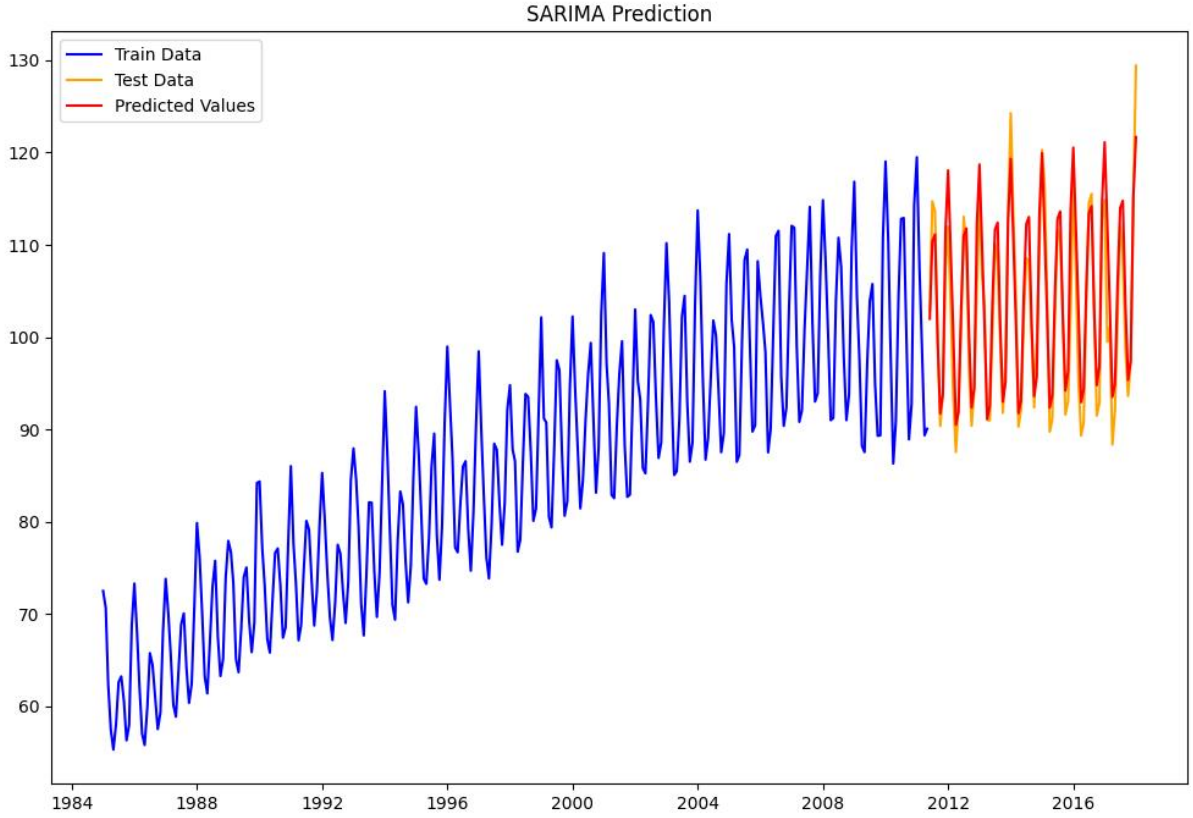
Obrázek č. 43 – Report o modelu SARIMAX [109]

Z reportu je vidět, že se jedná dokonce o model SARIMAX, který zahrnuje externí vlivy. Pro naše potřeby však dobře poslouží i jeho varianta bez X. Základní model ARIMA vyšel s parametry (1, 0, 2). To znamená, že model ARIMA bude mít v rámci složky AR závislost na jedné hodnotě zpátky, nebude potřeba standardní diferencování a MA bude záviset na dvou předchozích hodnotách. V rámci modelu SARIMA se zde vyskytují i parametry pro sezónní chování zahrnující i četnost sezón, které jsou v následujícím pořadí (0, 1, 1, 12).

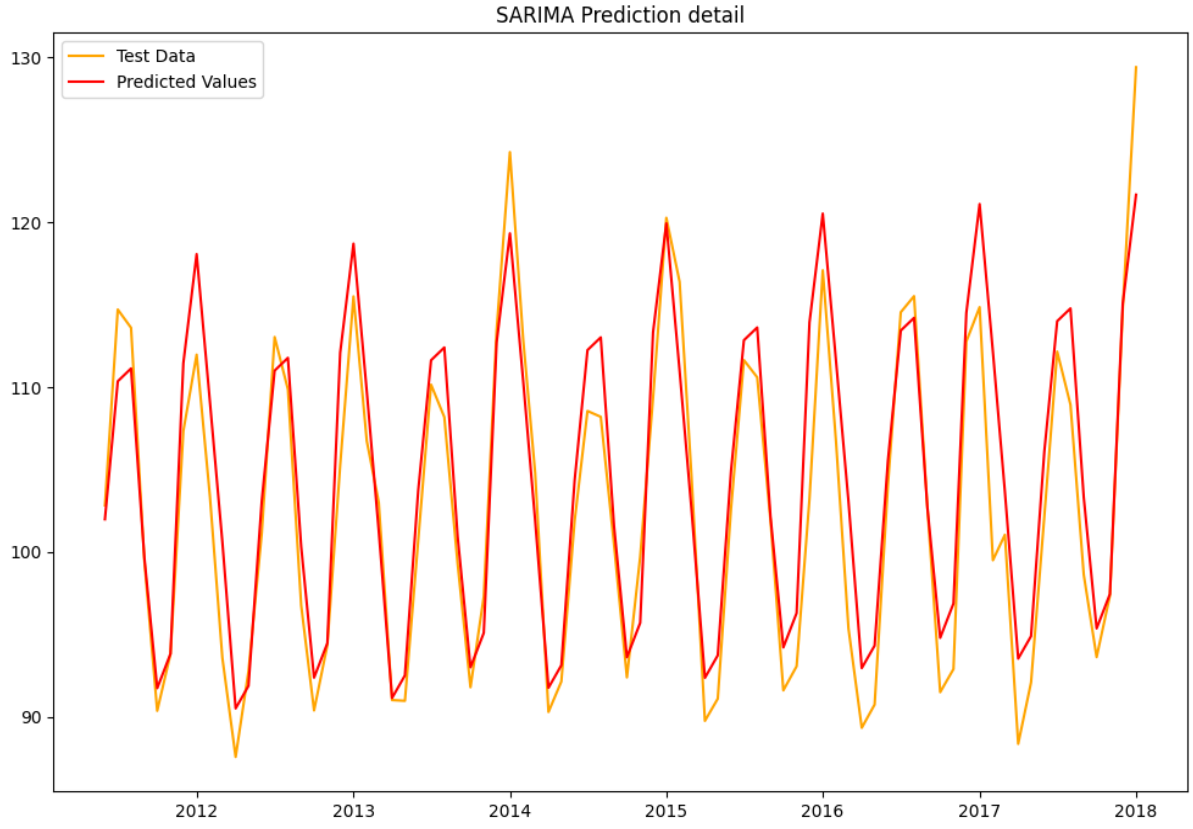
Následně už jen stačí vzít tyto parametry a vložit je do modelu společně s trénovacími daty a model nechám trénovat.

```
model = ARIMA(train_data, order=(1, 0, 2), seasonal_order=(0, 1, 1, 12))
model_fit = model.fit()
```

Poté samotný model odzkouším na testovacích datech a následně vykreslím graf, který zahrnuje trénovací data, testovací data a predikované hodnoty. Tento graf je k vidění na obrázku č. 44.



Obrázek č. 44 - Predikce s využitím modelu SARIMA [109]



Obrázek č. 45 – Detail predikce s využitím modelu SARIMA [109]

Aby však bylo možné tento model dále posoudit s jinými modely, tak je třeba vypočítat i chybovost modelu, kterou zjistím pomocí funkce mean_squere_error(), která je součástí knihovny Scikit-learn.

```
rmse = np.sqrt(mean_squared_error(test_data, prediction))
```

Chybovost modelu se zaokrouhlením na dvě desetinná místa vyšla s hodnotou 3,81.

ZÁVĚR

Na začátku teoretické části byly uvedeny oblasti využití datové analýza a časových řad. Poté byly vysvětleny jednotlivé druhy časových řad, dekompozice časové řady na jednotlivé složky a důležitý jev zvaný stacionarita. V rámci stacionarity byly představeny i její druhy a jak jí u časové řady dosáhnout. Dále byla pozornost věnována samotnému datamining, kdy došlo k rozebrání dat podle struktury a jejich původu, uvedení a rozdělení metodologií a zejména pak zpracování všech jednotlivých fází pro metodologii CRISP-DM. Během popisování jednotlivých fází byla zdůrazněna a podrobněji rozebrána především část o předzpracování dat. V hlavní teoretické části došlo k vytvoření komplexního přehledu o jednotlivých typových úlohách. Na začátku bylo uvedeno rozdělení podle účelu analýzy a dále pak rozdělení podle typu učení. Kromě hledání motivů a hledání asociačních pravidel bylo u všech typových úloh zobrazeno jejich podrobnější rozdělení a také několik příkladů, jak danou typovou úlohu řešit. V rámci další kapitoly o Python knihovnách byly povrchově zmíněny základní knihovny pro matematické výpočty, práci s daty a jejich následnou vizualizaci. Také došlo ke zmínění několika knihoven, které obsahují již implementované základní verze algoritmů.

V praktické části došlo k úvodu do použitých datových sad. V rámci výběru datové sady byla pozornost věnována především účelu analýzy těchto dat. Byly vybrány datové sady z obchodního, průmyslového, energetického a meteorologického sektoru. Většina datových sad byla použita ze stránky Kaggle. V rámci ozvláštnění byla přidána i datová sada o počasí, která byla získána z fotovoltaického geografického informačního systému přímo pro město Zlín. V rámci samotných ukázek byly vytvořeny praktické příklady v Jupyter notebooku využívající Python knihovny pro hledání odlehlých hodnot u teplotních senzorů, shlukovou analýzu pro identifikování teplých a studených dnů, klasifikování časových záznamů obchodních dat a predikování budoucích hodnot produkce energie. V rámci prvních tří případů bylo provedeno i srovnání ve formě tabulek se základní statistikou.

SEZNAM POUŽITÉ LITERATURY

[1] PALMA, Wilfredo. *Time series analysis: Wiley Series in Probability and Statistics*. United States, New Jersey, Hoboken: Wiley, 2016 [cit. 2023-05-18]. ISBN 1118634322. 978-1-118-63432-5.

[2] ERIC. Introduction to the Fundamentals of Time Series Data and Analysis. APTECH [online]. United States: APTECH, 2019-09-13 [cit. 2023-05-23]. Dostupné z: https://www.aptech.com/blog/introduction-to-the-fundamentals-of-time-series-data-and-analysis/

[3] BERG, Christian. The Complete Guide to Time Series Data. *Clarify* [online]. Norway: Clarify, 3.11.2021 [cit. 2023-05-18]. Dostupné z: https://www.clarify.io/learn/time-series-data

[4] TYAGI, Neelam. 5 Applications of Time Series Analysis. *Analytics Steps* [online]. India: Analytics Steps, 2020, 19.7.2021 [cit. 2023-05-18]. Dostupné z: https://www.analyticssteps.com/blogs/5-applications-time-series-analysis

[5] SOTO, Carlota. The unexploited power of industrial time-series data. *CrateDB* [online]. CrateDB, 2020-06-02 [cit. 2023-05-18]. Dostupné z: https://crate.io/blog/the-unexploited-power-of-industrial-time-series-data

[6] CALZON, Bernardita. 21 Examples Of Big Data Analytics In Healthcare That Can Save People. *Datapine* [online]. Germany: datapine, 2022-08-02 [cit. 2023-05-18]. Dostupné z: https://www.datapine.com/blog/big-data-examples-in-healthcare/

[7] ÖCAL, Göktuğ. Forecasting Energy Consumption With AI. *Faradai* [online]. United Kingdom: faradai [cit. 2023-05-18]. Dostupné z: https://faradai.ai/wp/forecasting-energy-consumption-with-ai/

[8] BOIXEDA, Pablo. Optimizing the Energy Sector with Data Analytics. *CLOUDERA Blog* [online]. United States: CLOUDERA Blog, 2022-12-20 [cit. 2023-05-18]. Dostupné z: https://blog.cloudera.com/optimizing-the-energy-sector-with-data-analytics/

[9] Measuring and Forecasting Weather: 2019-08-21. *Let's Talk Science* [online]. Canada [cit. 2023-05-18]. Dostupné z: https://letstalkscience.ca/educational-resources/backgrounders/measuring-and-forecasting-weather

[10] Using Data Analytics for Weather Forecasting. *Classes Near Me* [online]. United States: Noble Desktop, 2022-03-09 [cit. 2023-05-18]. Dostupné z: https://www.nobledesktop.com/classes-near-me/blog/data-analytics-for-weather-forecasting

[11] Concept | Time series data types and formats. *Dataiku* [online]. Dataiku [cit. 2023-05-18]. Dostupné z: https://knowledge.dataiku.com/latest/ml-analytics/time-series/concept-data-types-formats.html

[12] ZANGRE, Andrew. Discrete vs. Continuous Data: What's the Difference?. *G2* [online]. G2, 2023-04-12 [cit. 2023-05-18]. Dostupné z: https://www.g2.com/articles/discrete-vs-continuous-data

[13] KRISHNAN, Sandhya. What is Stationarity in Time Series? How it can be detected?. *Medium* [online]. Medium, 2021-10-18 [cit. 2023-05-18]. Dostupné z: https://medium.com/codex/what-is-stationarity-in-time-series-how-it-can-be-detected-7e5dfa7b5f6b

[14] CHAUDHARI, Satyapriya. Stationarity in Time Series Analysis Explained using Python. *QuantInsti Blog* [online]. QuantInsti Blog, 2021-02-11 [cit. 2023-05-18]. Dostupné z: https://blog.quantinsti.com/stationarity/

[15] BROWNLEE, Jason. How to Check if Time Series Data is Stationary with Python. *Machine Learning Mastery* [online]. Portoriko: Machine Learning Mastery, 2016-12-30 [cit. 2023-05-18]. Dostupné z: https://machinelearningmastery.com/time-series-data-stationary-python/

[16] PALACHY AFFEK, Shay. Detecting stationarity in time series data. *Towards Data Science* [online]. Towards Data Science, 2019-07-21 [cit. 2023-05-18]. Dostupné z: https://towardsdatascience.com/detecting-stationarity-in-time-series-data-d29e0a21e638

[17] SINGH, Aishwarya. A Gentle Introduction to Handling a Non-Stationary Time Series in Python. *Analytics Vidhya* [online]. Analytics Vidhya, 2018-09-13 [cit. 2023-05-18]. Dostupné z: https://www.analyticsvidhya.com/blog/2018/09/non-stationary-time-series-python/

[18] MITRANI, Alex. Achieving Stationarity With Time Series Data. *Towards Data Science* [online]. Towards Data Science, 2020-01-10 [cit. 2023-05-18]. Dostupné z: https://towardsdatascience.com/achieving-stationarity-with-time-series-data-abd59fd8d5a0

[19] J HYNDMAN, Rob a George ATHANASOPOULOS. *Forecasting: Principles and Practice* [online]. 3rd Edition. Melbourne, Australia: Otexts, 2021 [cit. 2023-04-21]. ISBN 0987507133. 978-0987507136. Dostupné z: https://otexts.com/fpp3/decomposition.html

[20] PLUMMER, Andrew. Different Types of Time Series Decomposition: And why you're probably using the wrong one. *Towards Data Science* [online]. Canada: Towards Data Science, 2020 [cit. 2023-04-24]. Dostupné z: https://towardsdatascience.com/different-types-of-time-series-decomposition-396c09f92693

[21] BROWNLEE, Jason. How to Decompose Time Series Data into Trend and Seasonality. Machine Learning Mastery [online]. Machine Learning Mastery, 2017-01-30 [cit. 2023-05-23]. Dostupné z: https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/

[22] J HYNDMAN, Rob a George ATHANASOPOULOS. *Forecasting: Principles and Practice* [online]. 3rd Edition. Melbourne, Australia: Otexts, 2021 [cit. 2023-04-21]. ISBN 0987507133. 978-0987507136. Dostupné z: https://otexts.com/fpp3/tspatterns.html

[23] BANERJEE, Probir. Time Series Analysis: Definition and Components. *Tutorials Point* [online]. India: Tutorials Point, 2006, 15.4.2022 [cit. 2023-04-21]. Dostupné z: https://www.tutorialspoint.com/time-series-analysis-definition-and-components

[24] YENIGÜN, Okan. Time Series Analysis: Mastering the Concepts of Stationarity: Unpacking the Mysteries of Stationarity. *Medium* [online]. USA: Medium, 2023, 7.2.2023 [cit. 2023-04-28]. Dostupné z: https://python.plainenglish.io/time-series-analysis-mastering-the-concepts-of-stationarity-c9fc489893cf

[25] What is Data Mining?. *Talend* [online]. Talend [cit. 2023-05-18]. Dostupné z: https://www.talend.com/resources/what-is-data-mining

[26] TWIN, ALEXANDRA. What Is Data Mining? How It Works, Benefits, Techniques, and Examples. Investopedia [online]. United States: Investopedia, 1999, 2023-4-15 [cit. 2023-05-18]. Dostupné z: https://www.investopedia.com/terms/d/datamining.asp

[27] GRAMLICH, Michael. What is structured, semi structured and unstructured data?. *Michael Gramlich* [online]. 2020-09-09 [cit. 2023-05-18]. Dostupné z:

https://www.michael-gramlich.com/what-is-structured-semi-structured-and-un-structured-data/

[28] YUSOV, Kirill. Structured vs Unstructured Data – What is the Difference?. Jelvix [online]. Jelvix [cit. 2023-05-23]. Dostupné z: https://jelvix.com/blog/structured-vs-unstructured-data

[29] VISHWAKARMA, Ashish. Difference between Structured, Semi-structured and Unstructured data. GeeksforGeeks [online]. India: GeeksforGeeks [cit. 2023-05-18]. Dostupné z: https://www.geeksforgeeks.org/difference-between-structured-semi-structured-and-unstructured-data/

[30] LABA, Yurii. What methodologies to use for data-rich projects?. *Medium* [online]. Medium, 2021-02-25 [cit. 2023-05-18]. Dostupné z: https://medium.com/intelliarts-ai/article-title-what-methodologies-to-use-for-data-rich-projects-3234ab4aac41

[31] What Is Data Mining: Definition, Examples, Tools, and Techniques (For Beginners). Georgia Tech Boot Camps [online]. United States: Georgia Tech Boot Camps [cit. 2023-05-23]. Dostupné z: https://bootcamp.pe.gatech.edu/blog/what-is-data-mining/

[32] DISRUPTIVENEXT. Applying CRISP-DM to Financial Time-series. *Medium* [online]. Medium, 2021-02-26 [cit. 2023-05-18]. Dostupné z: https://disruptivenext.medium.com/crisp-dm-for-financial-time-series-b4e01fcb4e8b

[33] LUNA, Zipporah. CRISP-DM Phase 1: Business Understanding. *Medium* [online]. Medium, 2021-07-26 [cit. 2023-05-18]. Dostupné z: https://medium.com/analytics-vidhya/crisp-dm-phase-1-business-understanding-255b47adf90a

[34] PETERSEN, Rob. 6 essential steps to the data mining process. *BarnRaisers* [online]. BarnRaisers, 2018-10-01 [cit. 2023-05-18]. Dostupné z: https://barnraisersllc.com/2018/10/01/data-mining-process-essential-steps/

[35] ANUNAYA, Sadhvi. Data Preprocessing in Data Mining – A Hands On Guide (Updated 2023). *Analytics Vidhya* [online]. Analytics Vidhya, 2021-08-10 [cit. 2023-05-18]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/08/data-preprocessing-in-data-mining-a-hands-on-guide/

[36] SINGH, Harshita. Understanding Data Preprocessing. *Towards Data Science* [online]. Towards Data Science, 2020-05-13 [cit. 2023-05-18]. Dostupné z: https://towardsdatascience.com/data-preprocessing-e2b0bed4c7fb

[37] KEERTHANA. DATA PREPROCESSING TECHNIQUES. *Medium* [online]. Medium, 2021-07-06 [cit. 2023-05-18]. Dostupné z: https://medium.com/almabetter/data-preprocessing-techniques-6ea145684812

[38] TANUSHREE7252. Data Mining Models. *GeeksforGeeks* [online]. India: GeeksforGeeks [cit. 2023-05-18]. Dostupné z: https://www.geeksforgeeks.org/data-mining-models/

[39] Descriptive and Predictive Data Mining Comparison: 6 Critical Differences. *Hevo* [online]. Hevo, 2022-04-06 [cit. 2023-05-18]. Dostupné z: https://hevodata.com/learn/descriptive-and-predictive-data-mining/

[40] KUMAR PANIGRAHI, Kiran. Difference Between Descriptive and Predictive Data Mining. *Tutorials Point* [online]. Tutorials Point, 2023-02-22 [cit. 2023-05-18]. Dostupné z: https://www.tutorialspoint.com/difference-between-descriptive-and-predictive-data-mining

[41] ALI, Moez. Supervised Machine Learning. Datacamp [online]. United States: datacamp, 2022-08 [cit. 2023-05-23]. Dostupné z: https://www.datacamp.com/blog/supervised-machine-learning

[42] KUMAR SAHU, Deepak. Supervised and Unsupervised Learning in Data Mining. *DigitalVidya* [online]. India: DigitalVidya, 2009 [cit. 2023-05-18]. Dostupné z: https://www.digitalvidya.com/blog/supervised-and-unsupervised-learning/

[43] Types of Learning. *Tutorials Point* [online]. Tutorials Point [cit. 2023-05-18]. Dostupné z: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_types_of_learning.htm

[44] OPPERMANN, Artem. What Is Unsupervised Learning?. *Built In* [online]. Built In, 2011, 2023-01-06 [cit. 2023-05-18]. Dostupné z: https://builtin.com/machine-learning/unsupervised-learning

[45] KLEIST, Caroline. *Time Series Data Mining Methods: A Review*. Berlin, 2015. Master Thesis. Humboldt-Universitat zu Berlin School of Business and Economics Ladislaus von Bortkiewicz Chair of Statistics. Vedoucí práce Prof. Dr. Wolfgang Karl Hardle.

[46] RUGGLES, Tyler, David FARNHAM, Dan TONG a Ken CALDEIRA. Developing reliable hourly electricity demand data through screening and imputation.

Scientific Data [online]. Scientific Data, 2020 [cit. 2023-05-23]. Dostupné z: https://www.nature.com/articles/s41597-020-0483-x

[47] FILIPPO. Creating a temporal range time-series spiral plot. In: STACK OVERFLOW [online]. STACK OVERFLOW, 2018-05-18 [cit. 2023-05-23]. Dostupné z: https://stackoverflow.com/questions/46575723/creating-a-temporal-range-time-series-spiral-plot

[48] ESLING, PHILIPPE a CARLOS AGON. *Time-Series Data Mining* [online]. Institut de Recherche et Coordination, 2012-11-22 [cit. 2023-05-18]. Dostupné z: http://sites.music.mcgill.ca/orchestration/files/2013/08/Esling_2012_ACMComput-Surv.pdf

[49] EDINE YAGOUBI, Djamel-Edine. *Indexing and analysis of very large masses of time series*. France, 2018. Dostupné také z: https://theses.hal.science/tel-01945348/document. Doctoral thesis. Université Montpellier.

[50] SHAH, Aashish. Time-series segmentation in python. STACK OVERFLOW [online]. STACK OVERFLOW, 2020-02-01 [cit. 2023-05-23]. Dostupné z: https://stackoverflow.com/questions/60020521/time-series-segmentation-in-python

[51] BHARATHWAJ, M. Clustering Techniques. *Towards Data Science* [online]. Towards Data Science, 2020-09-22 [cit. 2023-05-19]. Dostupné z: https://towardsdatascience.com/clustering-techniques-hierarchical-and-non-hierarchical-b520b5d6a022

[52] TAVENARD, Romain. Time Series Clustering. Tslearn [online]. tslearn, 2017 [cit. 2023-05-23]. Dostupné z: https://tslearn.readthedocs.io/en/stable/user_guide/clustering.html

[53] HEKA.AI. Time series clustering. *Medium* [online]. Medium, 2022-07-09 [cit. 2023-05-19]. Dostupné z: https://heka-ai.medium.com/time-series-clustering-b84bcaaa63ac

[54] SPARSHI, Kapil. Difference between Hierarchical and Non Hierarchical Clustering. *GeeksforGeeks* [online]. India: GeeksforGeeks [cit. 2023-05-19]. Dostupné z: https://www.geeksforgeeks.org/difference-between-hierarchical-and-non-hierarchical-clustering/

[55] GINNI. What is model-based clustering?. *Tutorials Point* [online]. Tutorials Point, 2022-02-15 [cit. 2023-05-19]. Dostupné z: https://www.tutorialspoint.com/what-is-model-based-clustering

[56] EDUCATION ECOSYSTEM (LEDU). Understanding K-means Clustering in Machine Learning. *Towards Data Science* [online]. Towards Data Science, 2018-09-12 [cit. 2023-05-19]. Dostupné z: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[57] K-Medoids clustering-Theoretical Explanation. *Java T point* [online]. java T point [cit. 2023-05-19]. Dostupné z: https://www.javatpoint.com/k-medoids-clustering-theoretical-explanation

[58] GUPTA, Aman. Fuzzy C-Means Clustering (FCM) Algorithm. *Medium* [online]. Medium, 2021-06-02 [cit. 2023-05-19]. Dostupné z: https://medium.com/geekculture/fuzzy-c-means-clustering-fcm-algorithm-in-machine-learning-c2e51e586fff

[59] SALTON DO PRADO, Kelvin. How DBSCAN works and why should we use it?. *Towards Data Science* [online]. Towards Data Science, 2017-04-02 [cit. 2023-05-19]. Dostupné z: https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80

[60] SINGH, Jay. When to use the Gaussian mixture model?. *Tutorials Point* [online]. Tutorials Point, 2023-02-27 [cit. 2023-05-19]. Dostupné z: https://www.tutorialspoint.com/when-to-use-the-gaussian-mixture-model

[61] RALHAN, Abhinav. Self Organizing Maps. *Medium* [online]. Medium, 2018-02-18 [cit. 2023-05-19]. Dostupné z: https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4

[62] SIEBERT, Julien. Overview of time series analysis Python packages: A review of python packages dedicated to time series analysis. Cornell University [online]. United States: Cornell University [cit. 2023-05-23]. Dostupné z: https://siebert-julien.github.io/time-series-analysis-python/

[63] Time Series Regression. *MathWorks* [online]. United States: MathWorks [cit. 2023-05-19]. Dostupné z: https://www.mathworks.com/discovery/time-series-regression.html

[64] SABLE, Anuj. Introduction to Time Series Forecasting: Regression and LSTMs. Paperspace [online]. Paperspace, 2021 [cit. 2023-05-23]. Dostupné z: https://blog.paperspace.com/time-series-forecasting-regression-and-lstm/

[65] ARMA model. *Statistics How To MENU* [online]. Statistics How To MENU [cit. 2023-05-19]. Dostupné z: https://www.statisticshowto.com/arma-model/

[66] ARTLEY, Brendan. Time Series Forecasting with ARIMA , SARIMA and SARIMAX. *Towards Data Science* [online]. Towards Data Science, 2022-04-26 [cit. 2023-05-19]. Dostupné z: https://towardsdatascience.com/time-series-forecasting-with-arima-sarima-and-sarimax-ee61099e78f6

[67] VERMA, Yugesh. A Guide to VARMA with Auto ARIMA in Time Series Modelling. *AIM* [online]. AIM, 2021-09-28 [cit. 2023-05-19]. Dostupné z: https://analyticsindiamag.com/a-guide-to-varma-with-auto-arima-in-time-series-modelling/

[68] BROWNLEE, Jason. How to Model Volatility with ARCH and GARCH for Time Series Forecasting in Python. *Machine Learning Mastery* [online]. United States: Machine Learning Mastery, 2019-08-24 [cit. 2023-05-19]. Dostupné z: https://machinelearningmastery.com/develop-arch-and-garch-models-for-time-series-forecasting-in-python/

[69] IBRAHIM, Mostafa. How to Use XGBoost and LGBM for Time Series Forecasting?. *365DataScience* [online]. 365DataScience, 2022-07-21 [cit. 2023-05-19]. Dostupné z: https://365datascience.com/tutorials/python-tutorials/xgboost-lgbm/

[70] The Ultimate Guide to Random Forest Regression. *Keboola* [online]. Keboola, 2020-09-17 [cit. 2023-05-19]. Dostupné z: https://www.keboola.com/blog/random-forest-regression

[71] RUPAK ROY - II, Bob. LSTMs for regression. *Medium* [online]. Medium, 2021-06-25 [cit. 2023-05-19]. Dostupné z: https://bobrupakroy.medium.com/lstms-for-regression-cc9b6677697f

[72] NADEEM. Time Series Forecasting using TBATS Model. *Medium* [online]. Medium, 2021-11-21 [cit. 2023-05-19]. Dostupné z: https://medium.com/analytics-vidhya/time-series-forecasting-using-tbats-model-ce8c429442a9

[73] DEVBAY. Time series classification – an overview. *DEVELOPERSBAY* [online]. DEVELOPERSBAY, 2020-11-09 [cit. 2023-05-19]. Dostupné z: https://developersbay.se/time-series-classification-an-overview/

[74] STOLL, Martin. An Empirical Study of Graph-Based Approaches for Semi-supervised Time Series Classification. ResearchGate [online]. Technische Universität Chemnitz, 2022-02-20 [cit. 2023-05-23]. Dostupné z: https://www.researchgate.net/publication/357989846_An_Empirical_Study_of_Graph-Based_Approaches_for_Semi-supervised_Time_Series_Classification

[75] Classification Algorithm in Machine Learning. *Java T point* [online]. java T point [cit. 2023-05-19]. Dostupné z: https://www.javatpoint.com/classification-algorithm-in-machine-learning

[76] R. DINGER, Timothy, Yuan-chi CHANG, Raju PAVULURI a Shankar SUBRAMANIAN. What is time series classification?. *IBM* [online]. IBM, 2022-01-25 [cit. 2023-05-19]. Dostupné z: https://developer.ibm.com/learningpaths/get-started-time-series-classification-api/what-is-time-series-classification/

[77] GROOTENDORST, Maarten. 9 Distance Measures in Data Science: The advantages and pitfalls of common distance measures. Towards Data Science [online]. Towards Data Science, 2021-02-01 [cit. 2023-05-23]. Dostupné z: https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa

[78] K-Nearest Neighbor(KNN) Algorithm for Machine Learning. *Java T point* [online]. java T point [cit. 2023-05-19]. Dostupné z: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

[79] Support Vector Machine Algorithm. *Java T point* [online]. java T point [cit. 2023-05-19]. Dostupné z: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[80] RAJ, Ashwin. Perfect Recipe for Classification Using Logistic Regression. *Towards Data Science* [online]. Towards Data Science, 2020-11-07 [cit. 2023-05-19]. Dostupné z: https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592

[81] ISMAIL FAWAZ, Hassan. *Deep learning for time series classification*. Mulhouse, 2020. Dostupné také z: https://theses.hal.science/tel-03715016/document. Doctoral thesis. Université de Haute Alsace - Mulhouse.

[82] TIOZZO, Annagiulia. Dynamic Time Warping for Time Series Classification. *Medium* [online]. Medium, 2022-11-09 [cit. 2023-05-19]. Dostupné z:

https://medium.com/eni-digitalks/dynamic-time-warping-for-time-series-classification-6482cf9d7c3b

[83] SRUTHI E R. Understand Random Forest Algorithms With Examples (Updated 2023). *Analytics Vidhya* [online]. Analytics Vidhya, 2021-06-17 [cit. 2023-05-25]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[84] Decision Tree. *GeeksforGeeks* [online]. GeeksforGeeks, 2023-05-08 [cit. 2023-05-25]. Dostupné z: https://www.geeksforgeeks.org/decision-tree/

[85] VATS, Rohan. Gaussian Naive Bayes: What You Need to Know?. *UpGrad* [online]. upGrad, 2015, 2021-02-22 [cit. 2023-05-25]. Dostupné z: https://www.upgrad.com/blog/gaussian-naive-bayes/

[86] SHABOU, Saif. Outlier detection in Time: Chapter 5. *Time Series with R* [online]. 2020-12-10 [cit. 2023-05-19]. Dostupné z: https://s-ai-f.github.io/Time-Series/outlier-detection-in-time-series.html

[87] KHANIN, Artur. Time Series and How to Detect Anomalies in Them: Part I. Akvelon [online]. United States: Medium [cit. 2023-05-23]. Dostupné z: https://akvelon.com/time-series-and-how-to-detect-anomalies-in-them-part-i/

[88] HESSING, Ted. Z Scores (Z Value) & Z Table. *Six Sigma Study Guide* [online]. Six Sigma Study Guide [cit. 2023-05-19]. Dostupné z: https://sixsigmastudyguide.com/z-scores-z-table-z-transformations/

[89] BHANDARI, Pritha. How to Find Interquartile Range (IQR) | Calculator & Examples. *Scribbr* [online]. Scribbr, 2020-09-25 [cit. 2023-05-19]. Dostupné z: https://www.scribbr.com/statistics/interquartile-range/

[90] KUMAR, Pramod. Understanding LOF (Local Outlier Factor) —perspective for implementation. *Medium* [online]. Medium, 2020-07-06 [cit. 2023-05-19]. Dostupné z: https://medium.com/@pramodch/understanding-lof-local-outlier-factor-for-implementation-1f6d4ff13ab9

[91] KILARU, Vasudeva. One Class Classification Using Support Vector Machines. *Analytics Vidhya* [online]. Analytics Vidhya, 2022-06-03 [cit. 2023-05-19]. Dostupné z: https://www.analyticsvidhya.com/blog/2022/06/one-class-classification-using-support-vector-machines/

[92] AKSHARA_416. Anomaly detection using Isolation Forest – A Complete Guide. *Analytics Vidhya* [online]. Analytics Vidhya, 2021-07-26 [cit. 2023-05-19]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/

[93] What is NumPy?. *NumPy* [online]. [cit. 2023-05-19]. Dostupné z: https://numpy.org/doc/stable/user/whatisnumpy.html

[94] About pandas. *Pandas* [online]. [cit. 2023-05-19]. Dostupné z: https://pandas.pydata.org/about/

[95] Matplotlib vs. seaborn vs. Plotly vs. MATLAB vs. ggplot2 vs. pandas. *Ritza* [online]. Ritza [cit. 2023-05-19]. Dostupné z: https://ritza.co/articles/matplotlib-vs-seaborn-vs-plotly-vs-MATLAB-vs-ggplot2-vs-pandas/

[96] KUMAR, Bijay. What is Matplotlib and how to use it in Python. *PythonGuides* [online]. 2021-08-06 [cit. 2023-05-19]. Dostupné z: https://pythonguides.com/what-is-matplotlib/

[97] ALI, Moez. Python Seaborn Tutorial For Beginners: Start Visualizing Data. *Datacamp* [online]. 2023-03 [cit. 2023-05-19]. Dostupné z: https://www.datacamp.com/tutorial/seaborn-python-tutorial

[98] KOUSHIK222. Getting Started with Plotly-Python. *GeeksforGeeks* [online]. [cit. 2023-05-19]. Dostupné z: https://www.geeksforgeeks.org/getting-started-with-plotly-python/

[99] TYAGIKARTIK4282. Data Analysis with SciPy. *GeeksforGeeks* [online]. [cit. 2023-05-19]. Dostupné z: https://www.geeksforgeeks.org/data-analysis-with-scipy/

[100] Scikit Learn - Introduction. *Tutorials Point* [online]. [cit. 2023-05-19]. Dostupné z: https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm

[101] How to install statsmodels in Python?. *Java T point* [online]. [cit. 2023-05-19]. Dostupné z: https://www.javatpoint.com/how-to-install-statsmodels-in-python

[102] G SMITH, Taylor. User Guide: About the project - pmdarima. Pmdarima [online]. pmdarima [cit. 2023-05-23]. Dostupné z: https://alkaline-ml.com/pmdarima/about.html#about

[103] YEGULALP, Serdar. What is TensorFlow? The machine learning library explained. InfoWorld [online]. United States: InfoWorld, 2022-06-03 [cit. 2023-05-

23]. Dostupné z: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

[104] SIMPLILEARN. What Is Keras: The Best Introductory Guide To Keras. Simplilearn [online]. 2023-04-01 [cit. 2023-05-23]. Dostupné z: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras

[105] SIMPLILEARN. What is PyTorch, and How Does It Work: All You Need to Know. Simplilearn [online]. Simplilearn, 2023-04-31 [cit. 2023-05-23]. Dostupné z: https://www.simplilearn.com/what-is-pytorch-article

[106] BROWNLEE, Jason. How to Use XGBoost for Time Series Forecasting. Machine Learning Mastery [online]. Machine Learning Mastery, 2021-05-19 [cit. 2023-05-23]. Dostupné z: https://machinelearningmastery.com/xgboost-for-time-series-forecasting/

[107] MOBIUS. Sensor Fault Detection Data. Kaggle: Help to limit the consequences of failures in automation systems [online]. Australia: Kaggle, 2020-11-05 [cit. 2023-05-24]. Dostupné z: https://www.kaggle.com/datasets/arashnic/sensor-fault-detection-data

[108] HTAG HOLDINGS. House Property Sales Time Series: What model will produce the most accurate forecast?. Kaggle [online]. Kaggle, 2019-08-12 [cit. 2023-05-24]. Dostupné z: https://www.kaggle.com/datasets/htagholdings/property-sales?fbclid=IwAR0_B9rVZIMFpu8-TIoEMpUWvGT-FBm9U9WZB6c87v_wIaCbiLfMcvti77Dg&select=raw_sales.csv

[109] SHENBAGAKUMARS. Time Series Datasets. Kaggle [online]. India: Kaggle, 2018-09-25 [cit. 2023-05-24]. Dostupné z: https://www.kaggle.com/datasets/shenba/time-series-datasets?select=Electric_Production.csv

[110] EUROPEAN COMMISSION, JOINT RESEARCH CENTRE. PVGIS Online Tool. EU Science Hub [online]. European Commission, Joint Research Centre [cit. 2023-05-24]. Dostupné z: https://joint-research-centre.ec.europa.eu/pvgis-online-tool_en

[111] EUROPEAN COMMISSION, JOINT RESEARCH CENTRE. PHOTOVOLTAIC GEOGRAPHICAL INFORMATION SYSTEM. EU Science Hub [online]. Italy: European Commission, Joint Research Centre, 2022-03-01 [cit. 2023-05-24]. Dostupné z: https://re.jrc.ec.europa.eu/pvg_tools/en/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AR	AutoRegression
ARCH	Autoregressive Conditional Heteroskedasticity
ARIMA	AutoRegressive Integrated Moving Average
ARMA	AutoRegressive Moving Average
BATS	Box-Cox Transformation, ARMA residuals, Trend and Seasonality
CRISP-DM	CRoss-Industry Standard Process for Data Mining
DBSCAN	Density-based spatial clustering of applications with noise
DFT	Discrete Fourier Transformation
DM	Data mining
DSA	Derivative Segment Approximation
DTW	Dynamic Time Warping
DWT	Discrete Wawelet Transformation
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GMM	Gaussian Mixture Model
GNB	Gaussian Naive Bayes
HMM	Hidden Markov Models
IQR	InterQuartile Range
KDD	Knowledge Discovery in Databases
K-NN	K-Nearest Neighbours
LOF	Local Outlier Factor
LSTM	Long Short Term Memory
MA	Moving Average
MDL	Minimum Description Length
OCSVM	One Class Support Vector Machines
PAA	Piecewise Aggregate Approximation

PCA	Principal Component Analysis
PGIS	Photovoltaic Geographical Information System
SARIMA	Seasonal AutoRegressive Integrated Moving Average
SARIMAX	Seasonal AutoRegressive Moving Average with eXogenous factors
SAX	Symbolic Aggregate Approximation
SEMMA	Sample, Explore, Modify, Model, Assess
SOM	Self Organizing Maps
SVD	Singular Values Decomposition
SVM	Support Vector Machines
TBATS	Trigonometric Seasonal, Box-Cox Transformation, ARMA residuals, Trend and Seasonality
VAR	Vector AutoRegression
VARIMA	Vector AutoRegressive Integrated Moving Average
VARIMAX	Vector AutoRegressive Moving Average with eXogenous factors
VARMA	Vector AutoRegressive Moving Average
VMA	Vector Moving Average

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

P1 - CD

PŘÍLOHA P I: CD

prilohy.zip

fulltext.pdf