VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

METODY DOLOVÁNÍ Z ČASOVÝCH ŘAD

KNOWLEDGE DISCOVERY FROM TIME SERIES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PETER KRUTÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Krutý Peter, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Inteligentní systémy
Název: **Metody dolování z časových řad**
Knowledge Discovery from Time Series
Kategorie: Data mining

Zadání:

1. Seznamte se s problematikou dolování dat, poté se podrobněji zaměřte na oblast dolování z časových řad.
2. Vyhledejte několik datasetů vhodných pro úlohy dolování časových řad.
3. Po konzultaci s vedoucím definujte vhodné úlohy a navrhněte experimentální aplikaci, která bude zvolenou úlohu provádět.
4. Navrženou aplikaci implementujte a ověřte její funkčnost.
5. Proveďte experimenty a vyhodnoťte přínos použitých metod.
6. Zhodnoťte dosažené výsledky a další možnosti pokračování v tomto projektu.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1.
- Fu, Tak-Chung. A Review on Time Series Data Mining. Engineering Applications of Artificial Intelligence, Engineering Applications of Artificial Intelligence 24(1), p. 164-181, 2011.

Při obhajobě semestrální části projektu je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 11. října 2021

Abstrakt

Táto práca sa zaoberá problematikou získavania znalostí z dát, konkrétne z časových radov. Jej cieľom je uskutočniť prieskum podpory programovacieho jazyka Python pre túto oblasť a následný návrh a implementáciu aplikácie, ktorá umožní demonštráciu a porovnanie vybraných metód dolovania. Metódy sú demonštrované v rámci experimentov, pre ktoré je vybratá vhodná dátová sada. Výstupom práce je porovnanie metód pre jednotlivé úlohy a aplikácia implementujúca zvolené metódy.

Abstract

This thesis is focused on the field of knowledge discovery from data, specifically from time series. Main objective is to research Python programming language support in this area and then design and implement an application that will allow to demonstrate and compare selected methods. Methods are demonstrated in experiments using appropriate data set. The output of the thesis is a comparison of methods for specific tasks and the application implementing selected methods.

Kľúčové slová

získavanie znalostí z databáz, dolovanie z dát, analýza dát, vizualizácia dát, časové rady, dekompozícia, reprezentácie, segmentácia, anotácia, sumarizácia, predpoveď, klasifikácia, zhlukovanie, vyhľadávanie, programovací jazyk Python, kryptomeny

Keywords

knowledge discovery from databases, data mining, data analysis, data visualization, time series, decomposition, representation, segmentation, annotation, summarization, forecasting, classification, clustering, search, Python programming language, cryptocurrencies

Citácia

KRUTÝ, Peter. *Metody dolování z časových řad*. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Metody dolování z časových řad

Prehlásenie

Prehlasujem, že túto diplomovú prácu som vypracoval samostatne pod vedením pána Ing. Vladimíra Bartíka, Ph.D. Pravdivo som uviedol všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Peter Krutý
16. mája 2022

Poďakovanie

Rád by som poďakoval vedúcemu práce, pánovi Ing. Vladimírovi Bartíkovi, Ph.D. za odbornú pomoc, trpezlivosť a cenné rady. Za pravopisnú korekciu textu ďakujem Mgr. Kataríne Krutej.

Obsah

Kapitola 1

Úvod

Informatizácia spoločnosti a automatizácia priemyslu výrazným spôsobom zvýšila množstvo generovaných a ukladaných údajov v úložiskách po celom svete. Údaje zaplavili takmer každý aspekt ľudského života. Množstvo elektronicky uložených údajov je dnes tak extrémne veľké, že ich manuálne spracovávanie a analýza sa stali časovo neefektívne a v niektorých prípadoch až nemožné. Naliehavá potreba automatizácie týchto činností viedla k vzniku novej vednej disciplíny, ktorá sa nazýva získavanie znalostí z databáz.

Získavanie znalostí je možné vykonávať z množstva rôznych typov zdrojových údajov. Jedným z nich sú časové rady. Ide o súbor pozorovaných hodnôt, ktoré sú zbierané chronologicky v časových okamihoch. Prítomnosť časových radov je možné nájsť v množstve rôznych odborov, akými sú napríklad ekonómia, spoločenské vedy, fyzikálne vedy, epidemiológia a financie. Zisk znalostí z časových radov sa odlišuje od tradičných údajov a vyžaduje si jedinečný prístup. V rámci práce je tento prístup analyzovaný v podobe prieskumu metód pre jednotlivé úlohy.

Text práce je logicky rozdelený do niekoľkých kapitol. V kapitole 2 je všeobecne popísaný teoretický úvod do problematiky získavania znalostí z databáz. Pre časové rady je vyhradená samostatná kapitola 3, kde je detailne popísaná ich reprezentácia, vizualizácia, analýza, spracovanie a metódy pre jednotlivé dolovacie úlohy. Znalosť princípu popísaných metód je potrebná pre pochopenie ďalších kapitol. Kapitola 4 predstavuje programovací jazyk *Python*, jeho vlastnosti, možnosti a podporu pre získavanie znalostí z časových radov. Kapitola 5 obsahuje podrobný popis návrhu, implementácie a použitia aplikácie implementujúcej metódy pre zvolené úlohy. V kapitole 6 sú prezentované experimenty, v rámci ktorých sú demonštrované vybrané metódy pre konkrétnu úlohu. Popis experimentov zahŕňa predstavenie dátovej sady, ukážku implementovanej dolovacej úlohy a zhodnotenie výsledkov.

Kapitola 2

Získavanie znalostí z databáz

Táto kapitola je zameraná na predstavenie problematiky získavania znalostí z databáz. Ide o základný popis teórie, ktorej vysvetlenie je potrebné pre pochopenie ďalších pokročilejších častí práce.

Úvod kapitoly je vyhradený predstaveniu oblasti získavania znalostí z databáz a vysvetlením dôležitých pojmov, ktoré s touto oblasťou súvisia. Získavanie znalostí z databáz je chápané ako proces zložený z niekoľkých krokov a tieto kroky sú v jednej z podkapitol detailne popísané. Ďalšou dôležitou časťou tejto kapitoly je vymedzenie základných zdrojov dát pre dolovanie. Koniec kapitoly je venovaný vymedzeniu a vysvetleniu typických úloh pre dolovanie z dát.

2.1 Čo je získavanie znalostí z databáz

Získavanie znalostí z databáz je extrémne rýchlo rastúci vedný odbor, ktorý poskytuje širokú podporu v analýze a v rozhodovaní. Formálne ide o proces extrakcie relevantných, netriviálnych, skrytých, vopred neznámych užitočných vzorov a modelov z obvykle veľkého množstva údajov [22]. Extrahované vzory a modely následne reprezentujú získané znalosti.

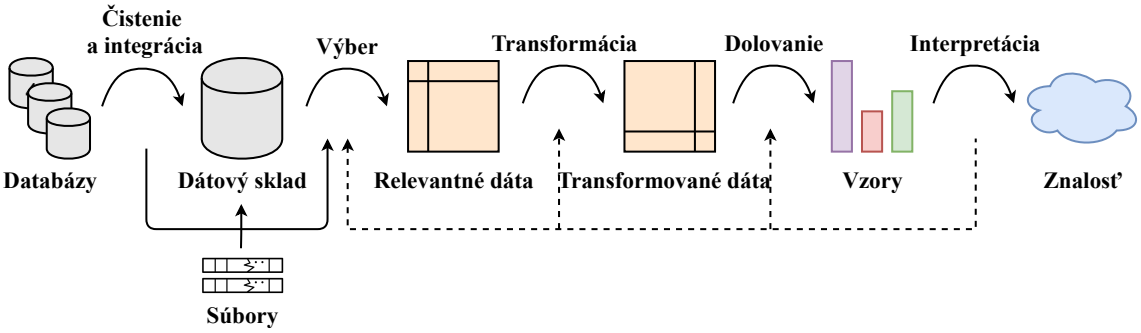
Tento vedný odbor ťaží z mnohých iných výskumných oblastí. Ide napríklad o štatistiku, strojové učenie, databázy, algoritmy alebo vizualizáciu [25]. Medzi jeho praktické aplikácie naopak patrí napríklad biznis, bioinformatika, financie, zdravotníctvo alebo vzdelávanie [6]. Všetky tieto oblasti vďaka zisku znalostí z údajov výrazným spôsobom profitujú.

Aj keď je definícia celkom jasná, pomerne často sa stretávame s tým, že niektoré operácie nad údajmi sú do tejto oblasti nesprávne zaraďované. Je teda vhodné uviesť príklady operácií, ktoré do tejto oblasti nepatria. Napríklad obyčajné vyhľadávanie údajov alebo jednoduché príkazy SELECT nad údajmi porušujú vlastnosť netriviálnosti. Hľadanie jasne viditeľných vzorov by zas porušovalo vlastnosť skrytosti. Vo všeobecnosti sa dá povedať, že ide o operácie, ktoré istým spôsobom porušujú jednu z vlastností znalosti uvedenej v definícii.

Význam pojmu získavania znalostí z databáz je v literatúre neraz zavádzajúci. Pomerne často sa stretávame s prípadom, kedy je zamieňaný s pojmom dolovanie z dát (angl. data mining). Niekedy sú tieto dva pojmy chápané ako synonymá. V tejto práci je však z dôvodu jednoznačnosti dolovanie z dát vnímané ako jedna časť procesu získavania znalostí. Podrobnejší popis procesu zisku znalostí a dolovania z dát je popísaný v nasledujúcej kapitole.

2.2 Proces získavania znalostí a jeho primitíva

Tak ako bolo spomenuté v úvode kapitoly, získavanie znalostí z databáz je proces zložený z niekoľkých krokov. Kroky sa obvykle opakujú vo vopred nestanovenom poradí, pričom na predošlý krok sa vracia iba v prípade potreby. Tak ako je uvedené na obrázku 2.1, proces získavania znalostí obsahuje sedem krokov [23]:



Obr. 2.1: Proces získavania znalostí (prevzaté z [23])

1. **Čistenie dát** – úlohou čistenia dát je odstránenie chýbajúcich hodnôt, identifikácia odľahlých hodnôt, vyhladenie zašumených údajov, vyriešenie nekonzistencie a redundancie.

2. **Integrácia dát** – úlohou integrácie dát je zlúčenie dát pochádzajúcich z rôznych dátových zdrojov.

3. **Výber dát** – úlohou výberu dát je vybrať dáta, ktoré sú z pohľadu dolovacej úlohy relevantné.

4. **Transformácia dát** – úlohou transformácie dát je zmena formátu dát do konsolidovanej podoby vhodnej pre dolovanie z dát. Zmenou formátu sa rozumie napríklad vykonanie sumarizácie alebo agregácie.

5. **Dolovanie z dát** – úlohou dolovania z dát je extrakcia vzorov z transformovaných dát, resp. vytvorenie modelu dát aplikovaním určitej metódy s konkrétnou implementáciu.

6. **Hodnotenie vzorov/modelu** – úlohou hodnotenia vzorov/modelu je identifikovať skutočne zaujímavé vzory na základe miery užitočnosti.

7. **Interpretácia/Prezentácia znalostí** – úlohou interpretácie/prezentácie znalostí je prezentovať/interpretovať získané vedomosti z vytvorených vzorov/modelu. Obvykle sú využívané rôzne techniky vizualizácie a reprezentácie znalostí.

Kroky 1 až 4 sú označované ako fáza predspracovania pre dolovanie. Typicky ide z pohľadu úsilia programátora o najnáročnejšiu fázu procesu. Hlavný krok procesu získavania znalostí je dolovanie z dát. V tomto kroku je niekedy potrebné interagovať s používateľom alebo databázou znalostí. Výsledok dolovania v podobe nových znalosti môže byť použitý v ďalších iteráciách alebo iných dolovacích úlohách.

Proces získavania znalostí sa typicky v rôznych prípadoch odlišuje. Odlišnosť procesu je definovaná na základe nasledujúcich **primitív** [23]:

- množina relevantných zdrojových údajov – výber vstupu dolovacej úlohy,
- druh dolovacej úlohy – výber typu výsledného modelu/znalosti,
- algoritmus dolovacej úlohy – výber algoritmu pre špecifický druh úlohy,
- doménová znalosť – podrobnejšie špecifikovanie problému ovplyvňujúce priebeh dolovania,
- miera a prah zaujímavosti – definícia hraničných hodnôt, napr. pre asociačné pravidlá minimálna podpora a spoľahlivosť,
- spôsob prezentácie výsledkov – výber spôsobu vizualizácie znalostí.

2.3 Typy zdrojových dát pre dolovanie

Všeobecne vzaté, dolovanie z dát je možné vykonať nad akýmkoľvek typom zmysluplných údajov. Zdrojové údaje sa obvykle rozdeľujú do dvoch kategórií: Tradičné dáta a netradičné (pokročilé) dáta. [23]

Tradičnými typmi zdrojových dát, s ktorými sa môžeme stretávať, sú:

- **Relačné databázy** – úložiská, v ktorých je hlavným prvkom relácia. Relácia pozostáva z dvoch častí, a to schéma relácie a inštancia relácie. **Inštancia relácie** je tabuľka zložená z množiny niekoľkých záznamov. Záznam je chápaný ako jeden riadok (n-tica), pričom všetky riadky majú rovnaký počet stĺpcov (atribútov). Názov a doménu každého stĺpca (atribútu) špecifikuje **schéma relácie**. Každý záznam v tabuľke je identifikovaný primárnym kľúčom zloženým z neprázdnej množiny atribútov. Jednotlivé stĺpce tabuľky musia obsahovať iba atomické hodnoty. [39]
- **Dátové sklady** – úložiská, pre ktoré platí, že obvykle pozostávajú z niekoľkých rozdielnych dátových zdrojov. Ich vznik vyžaduje a predchádza procesu čistenia dát, integrácie dát, transformácie dát, načítania dát a periodického aktualizovania dát. Údaje sú uložené z historickej perspektívy, organizované okolo hlavných subjektov a typicky sumarizované. Obvykle je modelovaný štruktúrou, ktorá sa nazýva **multidimenzionálna kocka**. Takisto je vhodný pre vykonávanie OLAP (Online Analytical Processing) operácií. [39] [23]
- **Transakčné databázy** – úložiska, ktoré sú organizované na základe istého druhu časovej známky. Každý záznam reprezentuje transakciu, pričom jednotlivé záznamy sú typicky uložené v obyčajnom súbore. Transakcia obyčajne zahŕňa jednoznačný identifikátor a zoznam položiek. [23]

Okrem relačných databáz, dátových skladov a transakčných databáz existuje množstvo ďalších typov zdrojových dát, ktoré sú označované ako netradičné (pokročilé). Medzi takéto údaje patria [23]:

- **Databázy časových radov** – úložiská obsahujúce údaje zaznamenané chronologicky v časových okamihoch. Čas záznamu hodnoty musí byť explicitne uvedený. Príkladom takejto databázy môžu byť údaje meranej teploty, tlaku, zisku atď. Pre popis časových radov a dolovanie z nich je vyhradená samostatná kapitola 3.

- **Sekvenčné databázy** – úložiská obsahujúce údaje v podobe usporiadanej množiny prvkov/udalostí. Čas vykonania udalostí nemusí byť zaznamenaný, dôležité je usporiadanie. Príkladom takejto databázy môže byť postupnosť lekárskych vyšetrení pacienta.
- **Prúdy dát** – úložiská obsahujúce údaje, ktoré sa produkujú súvisle, pričom ich objem je obrovský a potencionálne nekonečný (údaje zo senzorov atď.). Údaje, ktoré sú najstaršie, sa typicky v čase zahadzujú.
- **Grafové databázy** – úložiská obsahujúce údaje vo forme matematickej reprezentácie grafu. Graf je abstraktný objekt daný množinou vrcholov V a množinou hrán E medzi dvojicami vrcholov.
- **Priestorové a časopriestorové databázy** – úložiská obsahujúce údaje vzťahujúce sa k priestorovému usporiadaniu. Obvykle ide o geografické databázy obsahujúce údaje máp. Časopriestorové údaje majú od priestorových navyše zaznamenávanú dimenziu času.
- **Multimediálne databázy** – úložiská obsahujúce audio údaje, obrazové údaje alebo video údaje.
- **Textové databázy** – úložiská obsahujúce neštruktúrované, čiastočne štruktúrované (XML, JSON atď.) alebo štruktúrované údaje (katalógy knižníc, atď.).
- **Web** – veľká, rýchlo rastúca, heterogénna, široko distribuovaná databáza dostupná cez globálnu sieť internet. Obsahuje množstvo rôznych formátov (audio, video, text atď.) a disponuje veľkým potenciálom pre zisk znalostí.
- **Databázy veľmi rozsiahlych dát (Big Data)** – obvykle distribuované úložiská obsahujúce údaje, ktoré sú charakteristické rôznorodosťou, veľkým objemom a vysokou rýchlosťou rastu objemu. Spracovanie, analýza a dolovanie z týchto databáz sú typicky veľmi náročné na výpočtové prostriedky.

Pokročilé/Netradičné údaje často v rámci prípravy transformujeme extrakciou rysov/príznakov na tradičné údaje. Transformácia je obvykle vykonaná z dôvodov zjednodušenia ďalšieho spracovania, analýzy a dolovania.

2.4 Typy dolovacích úloh

Táto podkapitola je zameraná na vymedzenie a vysvetlenie základných typov dolovacích úloh. Inými slovami sú objasnené rôzne druhy modelov, ktoré môžeme z dát získavať. Ako prvá je popísaná technika popisovania konceptu/tried, následne dolovanie frekventovaných vzorov, predikcia, zhluková analýza, analýza vývoja údajov a analýza odľahlých objektov. Okrem uvedených typov dolovacích úloh existuje množstvo ďalších, avšak menej obvyklých. Vo všeobecnosti môžeme dolovacie úlohy rozdeliť do dvoch kategórií [16]:

1. **Deskriptívne** – ich cieľom je charakterizovať všeobecné vlastnosti medzi údajmi. Typickým predstaviteľom tohto typu úloh je napríklad hľadanie asociačných pravidiel alebo zhlukovanie.

2. **Prediktívne** – ich cieľom je na základe analýzy súčasných údajov určiť, resp. predpokladať budúce správanie alebo hodnoty. Ako príklad je možné uviesť klasifikáciu a regresiu.

Pri procese dolovania je generované veľké množstvo vzorov. Typicky však platí, že iba malá časť z nich je pre koncového používateľa zaujímavá. Pred samotným popisom jednotlivých dolovacích techník je teda vhodné stanoviť si kritériá, ktoré odlišujú zaujímavé vzory od nezaujímavých [23]:

- jednoduchá zrozumiteľnosť pre človeka,
- platnosť pre nové alebo testovacie údaje s istým stupňom určitosti,
- potenciálna užitočnosť,
- novosť.

Uvedené vlastnosti sú teda kľúčové preto, aby vytvorený vzor mohol byť označený za znalosť.

2.4.1 Popis konceptu/triedy

Jedným zo základných typov dolovacej úlohy je popis konceptu/triedy (zovšeobecňovanie). Túto techniku zaraďujeme do kategórie deskriptívnych úloh. Jej cieľom je poskytnúť stručný, súhrnný a výstižný popis určitej množiny údajov [24]. Vo firme predávajúcej športové potreby môžu byť potreby rozdelené do tried: kopačky a hokejky, priestory firmy môžu byť rozdelené do tried: sklady a predajne. Tieto triedy je následne užitočné charakterizovať popisom, ktorý je možné získať jedným z nasledujúcich spôsobov [23]:

- **Charakterizácia dát** – cieľová trieda je popísaná sumarizovaním jej vlastností. Údaje analyzovanej/cieľovej triedy sú obvykle vyberané príkazom SELECT.
- **Diskriminácia dát** – cieľová trieda je popísaná rozdielmi medzi jednou alebo viacerými triedami. Triedy, s ktorými sa porovnáva, sa nazývajú rozdielové.

2.4.2 Dolovanie frekventovaných vzorov, asociácií a korelácií

Ďalším typom dolovacej úlohy je odhaľovanie vzťahov medzi atribútmi. Medzi tento typ úloh patrí dolovanie frekventovaných zdrojov, asociácií a korelácií.

Tak ako názov napovedá, **frekventované vzory** sú vzory, ktoré sa objavujú v množine údajov často. Poznáme mnoho rôznych typov frekventovaných vzorov: frekventované množiny, frekventované podpostupnosti alebo iné podštruktúry, ako napr. podgrafy, podstromy atď. Za frekventovanú množinu môžeme typicky označiť množinu položiek, ktoré sa spolu vyskytujú často. Ako príklad je vhodné uviesť lyže a lyžiarky. Tieto dve položky sa obvykle vyskytujú v jednom nákupe veľmi často. Príkladom frekventovanej podpostupnosti by zase mohol byť sled nákupov jedného zákazníka. Prvý nákup by bol bicykel, následne stojan na bicykel a posledný náhradná duša do kolesa. [23]

Dolovaním frekventovaných vzorov odhaľujeme v údajoch zaujímavé asociácie a korelácie. Proces odhaľovania zaujímavých asociácií je nazývaný ako **asociačná analýza**. Výsledkom analýzy sú tzv. **asociačné pravidlá**. Asociačná analýza býva pomerne často doplnená o hľadanie zaujímavých **korelácií** medzi dvojicami asociačných pravidiel. [16]

Formát asociačného pravidla je v nasledujúcom tvare (2.1):

$$kupuje(X,'lyže') \Rightarrow kupuje(X,'lyžiarky')[podpora = 0.5\%, spoľahlivosť = 62\%], \quad (2.1)$$

kde premenná X reprezentuje zákazníka. Toto pravidlo informuje o tom, že zákazníci, ktorí si kupujú lyže, majú tendenciu v rovnakom nákupe kupovať aj lyžiarky. Významnosť pravidla v tomto prípade udávajú dva parametre: **podpora** a **spoľahlivosť**. Podpora 0.5 % značí, že 0.5 % nákupov obsahuje spolu položky lyže a lyžiarky. Spoľahlivosť 62 % značí, že v 62 % nákupoch si zákazník v prípade kúpy lyží kúpil aj lyžiarky. Podpora a spoľahlivosť sú dva najčastejšie používané parametre, avšak existujú aj ďalšie. Asociačné pravidlá, ktoré nepresiahnu minimálnu hranicu týchto parametrov, bývajú typicky odstránené.

Uvedené asociačné pravidlo obsahuje iba jeden predikát, a preto sa nazýva **jednodimenzionálne asociačné pravidlo**. Asociačné pravidlá však môžu obsahovať aj viac predikátov, napr. pravidlo (2.2):

$$pohlavie(X,'muž') \wedge vek(X,'15..25') \Rightarrow kupuje(X,'korčule')$$
$$[podpora = 1\%, spoľahlivosť = 40\%] \quad (2.2)$$

obsahuje 3 predikáty: *pohlavie*, *vek* a *kupuje*. Takéto pravidlá sa nazývajú **multidimenzionálne asociačné pravidlá**. [23]

Dolovanie frekventovaných vzorov, asociácií a korelácií má široké uplatnenie v praxi. Ako príklad je možné uviesť hľadanie strategickej polohy skupín produktov v predajni alebo našepkávanie vhodných produktov pri nákupe v online obchode.

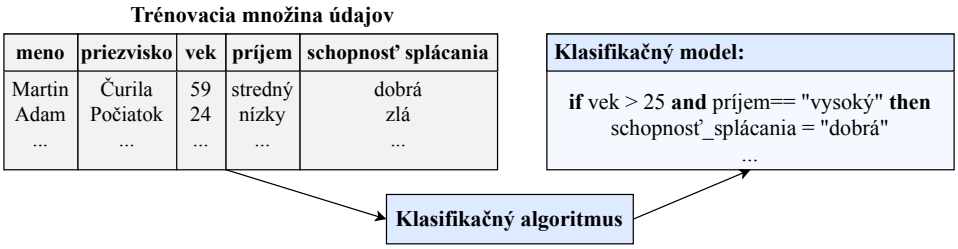
2.4.3 Klasifikácia a regresia

Medzi najčastejšie typy dolovacích úloh patrí aj klasifikácia a regresia. Úlohou a cieľom **klasifikácie** je vytvorenie modelu, ktorý popisuje a vymedzuje jednotlivé triedy dát. Model je následne využívaný pre predikciu kategorických hodnôt, t. j. hodnôt, ktoré sú diskrétne a neusporiadané. Toto je zároveň hlavná odlišnosť od regresie. **Regresia** je typ dátovej analýzy, na základe ktorej sú predikované pomocou regresnej funkcie numerické hodnoty spojitého charakteru. [16]

Klasifikácia a regresia má množstvo praktických uplatnení. Pomáhajú napríklad v oblasti priemyselnej výroby, zdravotníctva, marketingu a všade, kde je užitočné niečo identifikovať alebo predpovedať.

Proces klasifikácie sa skladá z troch krokov: trénovanie/učenie modelu, testovanie modelu a aplikácia modelu. Pred procesom klasifikácie je typicky potrebné istým spôsobom upraviť zdrojové údaje na vstupe. Medzi najzákladnejšie úpravy patrí čistenie dát, analýza významnosti a transformácia dát [23].

V prvej fáze, t. j. fáze **trénovania/učenia** prebieha analýza súboru dát. Tento súbor dát je označovaný ako **tréningová množina** objektov. Pre jednotlivé objekty tréningovej množiny platí, že poznáme ich triedy. Z toho vyplýva, že na základe tejto množiny dát je možné vytvoriť konkrétny klasifikačný model, ktorý dokáže predikovať triedy neznámych objektov. Priebeh tejto fázy je ilustrovaný na obrázku 2.2. [6]

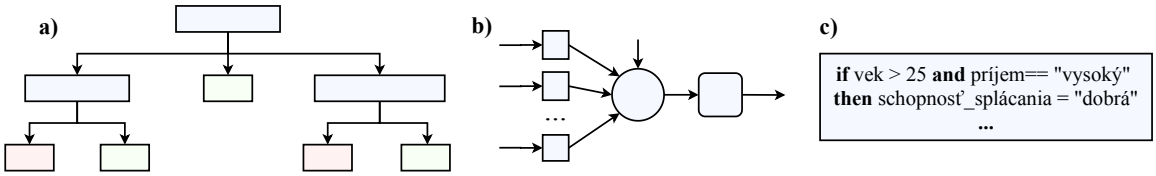


Obr. 2.2: Fáza trénovania/učenia

Klasifikačný model môže nadobúdať rôzne podoby [23]:

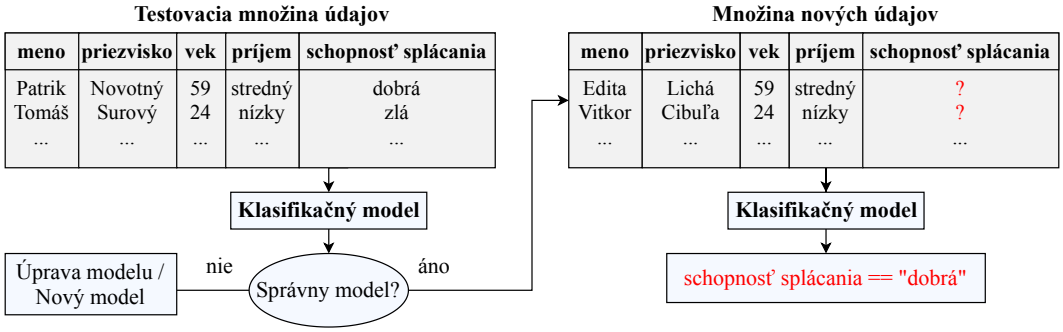
- **rozhodovacie stromy** – metóda *ID3*, *C4.5*, *Gini index*, náhodný les atď.,
- **matematické formuly** – Bayesovská klasifikácia a siete, *SVM* atď.,
- **klasifikačné pravidlá** – pravidlá v tvare podmienok,
- **neurónové siete** – *Perceptron*, *Backpropagation* atď.

Štruktúra modelu rozhodovacieho stromu, neurónu a pravidiel je ukázaná na obrázku 2.3.



Obr. 2.3: Reprezentácie modelov

Účelom druhej fázy je **testovanie** a **vyhodnotenie** klasifikačného modelu, inými slovami určenie miery úspešnosti/neúspešnosti klasifikácie. V tejto fáze sa pracuje s tzv. **validačnou množinou** objektov. Pre túto množinu údajov však platí, že ich triedy poznáme. Práve vďaka tejto vedomosti, dokážeme určovať mieru úspešnosti/neúspešnosti na základe porovnania referenčných tried s výsledkami klasifikátora. [6]



Obr. 2.4: Fáza testovania a klasifikácie

Na základe výsledkov z predošlého kroku je následne potrebné rozhodnúť, či je alebo nie je možné model použiť. V prípade, že je model dostatočné presný, je možné klasifikovať reálne údaje s neznámymi triedami. V opačnom prípade je potrebné model upraviť, prípadne vytvoriť nový. Priebeh fázy testovania a klasifikácie je uvedený na obrázku 2.4.

Postup pri regresnej analýze je veľmi podobný ako pri klasifikácii. Takisto sú realizované kroky trénovania/učenia, testovania a aplikácie modelu. V kroku trénovania/učenia je realizovaný odhad typu regresnej funkcie a odhad koeficientov regresnej funkcie. V rámci regresnej funkcie rozlišujeme rôzne typy závislosti:

- **jednoduchá regresia** – závislosť dvoch numerických premenných,
- **viacnásobná regresia** – závislosť jednej premennej na skupine iných numerických premenných,
- **viacrozmerná regresia** – závislosť skupiny viacnumerických premenných na skupine iných numerických premenných.

Závislosť premenných môže byť rôzna: lineárna, polynomiálna, hyperbolická, exponenciálna atď. Zároveň je vhodné uviesť, že regresná analýza je citlivá na odľahlé hodnoty a zložitosť výpočtu rastie s počtom závislých premenných. [16]

2.4.4 Zhlukovanie

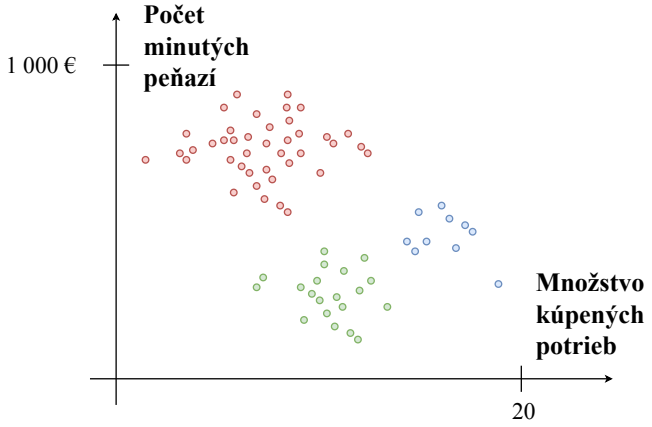
Zhluková analýza je ďalším obvyklým typom dolovacej úlohy. Jej cieľom je rozdeľovanie objektov do skupín na základe vopred definovaného kritériá podobnosti. Výsledkom zhlukovania sú teda homogénne skupiny. Skupiny sú na rozdiel od klasifikácie vopred neznáme. Objekty v jednej skupine by mali byť v ideálnom prípade maximálne podobné a zároveň minimálne podobné s objektmi inej skupiny. Podobnosť objektov sa určuje pomocou **vzdialenostnej funkcie**. Jej výber spolu s implementáciou konkrétnej metódy závisí od výslednej kvality zhlukovej analýzy. [54]

Existuje viacero metód pre zhlukovanie a možno ich rozdeliť do viacerých kategórií [38]:

- **Hierarchické metódy** – vytvárajú hierarchický rozklad množiny objektov zhora-nadol alebo zdola-nahor na základe vzdialenostnej matice. Nie je nutné stanoviť počet tried, ale je vhodné uviesť ukončujúcu podmienku. Konkrétnymi metódami sú *Diana*, *Agnes* atď.
- **Metódy založené na rozdeľovaní/vzdialenosti** – realizujú rozklad n objektov do vopred stanoveného počtu k tried. Platí, že $k \leq n$, pričom každý zhluk obsahuje aspoň jeden objekt a každý objekt patrí iba do jednej triedy. Konkrétnymi metódami sú K-means, K-medoids atď.
- **Metódy založené na hustote** – realizujú rozklad množiny objektov na základe hustoty. Zhluk je zväčšovaný, pokiaľ hustota objektov v susedstve neklesne pod vopred stanovenú hranicu. To znamená, že oblasti s veľkou hustotou objektov sú oddelené oblasťami s malou hustotou objektov. Konkrétnymi metódami sú *DenClue*, *DBSCAN* atď.
- **Metódy založené na mriežke** – vytvárajú rozklad priestoru do buniek tvoriacich mriežku. Samotné zhlukovanie nie je realizované nad objektami, ale nad bunkami mriežky. Konkrétnymi metódami sú *WaveCluster*, *Clique* atď.

- **Metódy založené na modeloch** – snažia sa o optimalizáciu zhody medzi zdrojovými údajmi a vhodným matematickým modelom. Hľadanie zhlukov je následne realizované tak, aby maximálne zodpovedali danému modelu. Konkrétnymi metódami sú *Expectation-Maximization*, *COBWEB* atď.

Ako ukážku zhlukovania je vhodné uviesť obrázok 2.5, kde sú zhlukovaní zákazníci firmy predávajúcej športové potreby. Zhlukovanie prebieha na základe počtu minutých peňazí a množstva kúpených potrieb za jeden konkrétny mesiac.



Obr. 2.5: Zhlukovanie – Zákazníci športového obchodu

2.4.5 Vyhľadávanie podobných/odľahlých údajov

Vyhľadávanie podobných údajov je úloha, v ktorej sa snažíme odhaliť záznamy / hodnoty, ktoré sú maximálne podobné vybraným záznamom/hodnotám. Nájdené maximálne podobné záznamy/hodnoty je možné využiť napr. pre dodatočné trénovanie modelov s cieľom optimalizácie výsledkov. [23]

Vyhľadávanie odľahlých údajov je opakom predošlej úlohy. Už názov napovedá, že v tejto úlohe z pohľadu tvorby modelu nie sú zaujímave obvyklé záznamy/hodnoty. Práve naopak, zaujímajú nás záznamy/hodnoty, ktoré sa významne odlišujú. Vznik odľahlého záznamu/hodnoty môže byť zapríčinený chybou merania, absenciou merania alebo netypickým správaním. Záznam/hodnota, ktorý vznikol chybou/absenciou merania, je obvykle potrebné **doplniť, upraviť alebo odstrániť**, nakoľko negatívnym spôsobom ovplyvňuje analýzu dát. Odľahlé hodnoty sa snažíme **analyzovať** v prípade, že značia nezvyčajné správanie. Nezvyčajné správanie totiž často môže byť zapríčinené podvodom. [23]

2.4.6 Analýza vývoja údajov

Ďalším typom dolovacej úlohy, ktorý je v tejto podkapitole popísaný, je analýza vývoja údajov. Jej cieľom je popisovať a modelovať trend, pravidelnosť a odchýlku vývoja objektov, ktorých správanie sa mení v čase. Konkrétnym príkladom takejto úlohy je **analýza podobností** alebo **analýza periodicity** v čase. S úlohou **analýzy sekvenčných vzorov** sa stretávame v prípade, že záznamy neobsahujú informácie o čase. Výsledky analýz bývajú využité typicky pri **predpovedi budúcich hodnôt**. Podrobnejší popis úlohy je uvedený v ďalšej kapitole. [16]

Kapitola 3

Získavanie znalostí z časových radov

Úlohou tejto kapitoly je detailne predstaviť problematiku získavania znalostí z časových radov. V prvej časti kapitoly je definovaný pojem časový rad, jeho vlastnosti a oblasti, v ktorých sa s ním stretávame. Predspracovanie, analýza, dolovanie a všeobecne práca s časovými radmi sa odlišuje od práce s tradičnými údajmi a vyžaduje si jedinečný prístup. Tento prístup je popísaný v podkapitolách analýzy, predspracovania a dolovania z časových radov. V rámci podkapitoly analýzy a predspracovania sú diskutované prevody reprezentácií, segmentácia, dekompozícia, stacionarita, chýbajúce hodnoty, odľahlé hodnoty atď. V podkapitole dolovania z časových radov sú popísané úlohy: predpoveď, klasifikácia, zhluková analýza, regresia, vyhľadávanie podobností/odľahlostí atď. V rámci každej úlohy sú diskutované jej základné metódy realizácie.

3.1 Čo je časový rad

Časový rad je definovaný ako **postupnosť pozorovaných hodnôt meniacich sa v čase**. Hodnoty časového radu sú zbierané chronologicky v pravidelných časových intervaloch, pričom majú numerickú a spojitú povahu. Charakteristickými vlastnosťami časového radu je vysoký počet zaznamenaných položiek a ich nepretržitá aktualizácia. Pre každú položku musí zároveň platiť, že má explicitne uvedený čas zaznamenania hodnoty. [19]

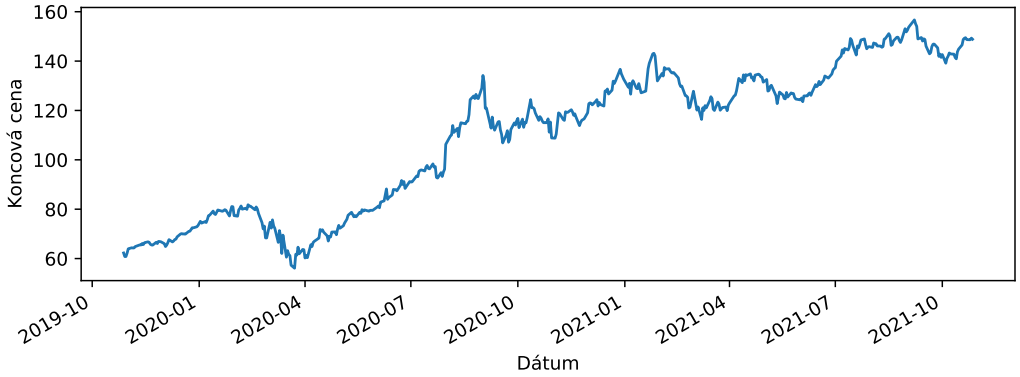
Časový interval, v ktorom sa údaje zbierajú, sa označuje ako **frekvencia časového radu**. Frekvencia sa vo všeobecnosti v rámci časových radov môže líšiť. Rovnako sa môže líšiť aj počet zaznamenávaných hodnôt. Ak je zaznamenávaná iba jedna hodnota, časový rad sa označuje ako **jednodimenzionálny**. V prípade, že sa v čase zaznamenáva viac hodnôt, časový rad je označovaný ako **multidimenzionálny**. [15]

Zdrojom generovaných údajov môžu byť senzory alebo iné zariadenia pre zber dát. Prítomnosť časových radov je možné nájsť v množstve rôznych odborov s množstvom konkrétnych príkladov [47]:

- **Ekonómia** – hrubý domáci produkt (HDP), index spotrebiteľských cien (CPI), nezamestnanosť atď.
- **Spoločenské vedy** – pôrodnosť, populácia, migrácia, politické ukazovatele.
- **Fyzikálne vedy** – teplota, vlhkosť, veternosť, tlak, znečistenie atď.

- **Medicína** – sledovanie krvného tlaku, sledovanie hmotnosti, meranie cholesterolu, sledovanie tepovej frekvencie atď.
- **Epidemiológia** – miera nákazy, úmrtnosť, vyliečenosť, zaočkovanosť atď.
- **Financie** – akcie, kryptomeny, nehnuteľnosti atď.

Konkrétny jednodimenzionálny časový rad je možné vidieť na obrázku 3.1. Ide o vizualizáciu ceny akcií spoločnosti *Apple Inc.* za posledné 2 roky pre dátum *28.10.2021*.



Obr. 3.1: Časový rad ceny akcií spoločnosti *Apple Inc.* za posledné 2 roky pre dátum *28.10.2021*

3.2 Analýza a predspracovanie časových radov

Táto podkapitola predstavuje základné techniky analýzy a predspracovania časových radov. Konkrétne sú pre časové rady popísané kroky redukcie, segmentácie, dekompozície, analýzy stacionarity, analýzy autokorelácie a čiastočnej autokorelácie. V závere je tiež vysvetlený proces analýzy a spracovania chýbajúcich a zašumených hodnôt.

3.2.1 Redukcia

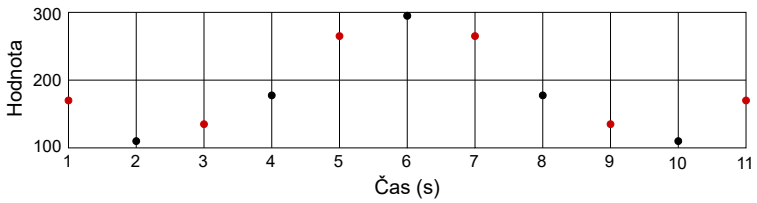
Už v predošlej podkapitole bolo uvedené, že charakteristickými vlastnosťami časových radov sú: numerická/spojitá povaha, veľké množstvo a vysoká dimenzionalita dát. Spracovanie údajov s takýmito vlastnosťami je typicky veľmi náročné a drahé. Ide teda o problém, ktorý je potrebné riešiť. Riešením je redukcia/zníženie množstva zaznamenaných údajov. Pre redukciu existuje viacero metód a výsledkom ich aplikácie sú jedinečné reprezentácie. Reprezentácie je všeobecne možné rozdeliť do troch kategórií [19]:

1. **Reprezentácie časovej domény** – reprezentácie, ktoré zachovávajú časový pohľad nad údajmi.
2. **Reprezentácie spektrálnej/frekvenčnej domény** – reprezentácie, ktoré premietajú časový pohľad nad údajmi do frekvenčného/spektrálneho.
3. **Ostatné reprezentácie** – reprezentácie, na ktoré je aplikovaná nejaká iná transformácia, ktorá sa nezaraďuje ani do jednej z predošlých dvoch kategórií.

Konkrétne reprezentácie spolu s metódami ich zisku sú popísané nasledovne.

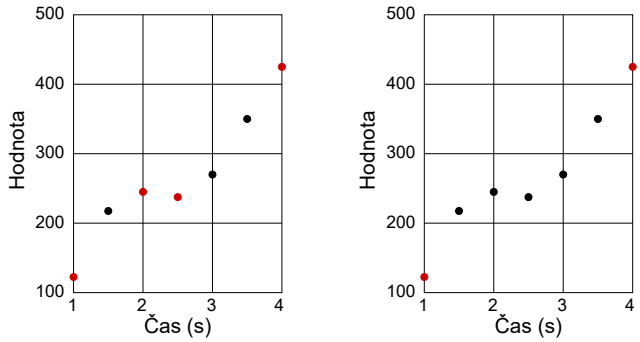
Reprezentácie časovej domény

Najjednoduchšou metódou redukcie údajov je **vzorkovanie**. Vzorkovanie je metóda, ktorá spočíva v extrakcii stanoveného počtu fixne vzdialených objektov. Vstupom metódy môže byť požadovaný počet extrahovaných objektov alebo dĺžka kroku medzi extrahovanými objektami. Výsledkom metódy je reprezentácia, ktorá zachováva časovú doménu. Nevýhodou vzorkovania je, že v prípade stanovenia príliš nízkeho počtu vzorkov môže dôjsť ku skresleniu časového radu. Ilustrácia metódy vzorkovania časového radu s krokom 2 je zobrazená na obrázku 3.2. [56]



Obr. 3.2: Vzorkovanie časového radu

Ďalším obvyklým prístupom je metóda **extrakcie extrémov**. Ide o metódu zachovávajúcu časovú doménu. Jej princíp spočíva v zisku všetkých lokálnych extrémov časového radu. Po extrakcii sa obvykle vykonáva zoradenie extrémov podľa dôležitosti, pričom sa odstránia najmenej dôležité hodnoty. Miera odstraňovaných a uchovávaných hodnôt je definovaná prahovou hodnotou dôležitosti. Ilustrácia extrakcie extrémov je uvedená na obrázku 3.3 vľavo. V pravej časti obrázka je demonštrované odstránenie najmenej dôležitých extrémov. [17]



Obr. 3.3: Extrakcia najdôležitejších extrémov časového

Metóda **prevodu na úroveň bitu** opäť zachováva časovú doménu. Oproti predošlým dvom metódam sa však odlišuje tým, že neredukuje počet zaznamenávaných objektov, ale množinu zaznamenávaných hodnôt objektu. Objekty môžu v rámci tejto reprezentácie nadobúdať iba dve hodnoty 0 alebo 1. Hodnota je rovná 1, ak bola pôvodná hodnota väčšia ako stredná hodnota časového radu. Inak je hodnota rovná 0. Táto metóda sa dá zovšeobecniť na akýkoľvek počet množiny hodnôt. Nevýhodou metódy je, že sa z časového radu stráca veľké množstvo informácie. [3]

Existuje ešte veľké množstvo ďalších metód redukcie údajov zachovávajúcich časovú doménu. Ich odlišnosti obvykle spočívajú v spôsobe interpretácie dôležitosti extrahovaných a

redukovaných bodov. Rovnako existuje aj množstvo reprezentácií využívaných segmentáciu viď podkapitola 3.2.2.

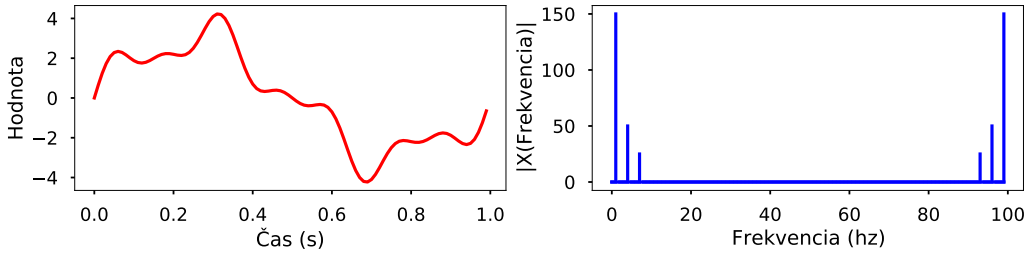
Reprezentácie spektrálnej/frekvenčnej domény

Aj v prípade redukcií časových radov do spektrálnej/frekvenčnej domény existuje viacero metód. V tejto práci sú vysvetlené dve najzákladnejšie metódy: Diskrétna Fourierova transformácia (DFT) a Diskrétna vlnková transformácia (DWT).

Diskrétna Fourierova transformácia (DFT) je metóda redukcie, ktorá premieta časovú doménu do frekvenčnej/spektrálnej domény. Vychádza z myšlienky, že akákoľvek spojitá periodická funkcia môže byť transformovaná na kombináciu sínusových a kosínusových vĺn označovaných ako **Fourierova séria**. Každá sínusová/kosínusová vlna je reprezentovaná komplexným číslom označovaným ako **Fourierov koeficient**. Diskrétna Fourierova transformácia konkrétne mapuje diskrétnu periodickú sekvenciu f na diskrétnu sekvenciu koeficientov F. Formálny zápis rozloženia časového radu je uvedený vo vzorci 3.1, pričom N je rovné veľkosti f. [48]

$$f[t] = \frac{1}{N} \sum F[j] e^{\frac{2\pi i t j}{N}} \tag{3.1}$$

Vizualizáciu rozloženia časového radu na Fourierovu sériu je možné vidieť na obrázku 3.4. V ľavej časti je uvedený časový rad, v pravej časti koeficienty Fourierovej série.



Obr. 3.4: Diskrétna Fourierova transformácia

Diskrétna vlnková transformácia (DWT) je metóda premietajúca časovú doménu do frekvenčnej/spektrálnej domény. Vychádza z myšlienky, že každá spojitá funkcia môže byť rozložená na základe vlnkových funkcií. Vlnky sú lokálne funkcie, ktoré umožňujú väčšiu presnosť lokálnych detailov. Ide teda o presnejší variant DFT, nakoľko poskytuje informácie o frekvencii v lokálnych častiach radu v jasne definovanom čase. [35]

Ostatné reprezentácie

V tejto sekcii sú v stručnosti uvedené niektoré reprezentácie, ktoré nepatria do predošlých dvoch kategórií.

Prvým typom, ktorý je vhodné uviesť, sú **symbolické reprezentácie**. Ide o reprezentácie, kde je časový rad prevádzaný na postupnosť symbolov. Výhodou týchto reprezentácií je, že môžeme vyžiť techniky dolovania z textu. Nevýhodou a problémom je, že časové rady sú potencionálne nekonečné a k dispozícii je iba obmedzená abeceda symbolov. Konkrétnym príkladom metódy redukcie časového radu na postupnosť symbolov je metóda *MCB (Multiple Coefficient Binning)* alebo *SFA (Symbolic Fourier Approximation)*. [45]

Ďalším typom reprezentácie časového radu môže byť **Skrytý Markovov model (*HMM*)**. Redukcia v tejto reprezentácii spočíva v rozdelení časového radu na časti, pričom podobné časti reprezentujú 1 stav v modeli. Časový rad je teda v tejto reprezentácii zastúpený postupnosťou stavov *HMM* modelu. [2]

3.2.2 Segmentácia

Segmentácia je proces, ktorý spočíva v rozdelení časových radov na množinu neprekrývajúcich sa segmentov. **Segment**, inak nazývaný časové okno, je následne definovaný ako podsekvencia časového radu. Segmentácia časového radu môže byť chápaná dvomi spôsobmi buď ako krok predspracovania pri redukcii, alebo ako technika analýzy trendu. Cieľom segmentácie je, aby vopred definovaný počet segmentov čo najlepšie reprezentoval pôvodný časový rad. [27]

Segmentačné algoritmy je možné rozdeliť do troch tried [27]:

- **Algoritmy kĺzavého okna** – princíp algoritmov spočíva v zväčšovaní jedného segmentu začínajúc prvým údajovým bodom, kým nie je prekročená určitá hranica chyby. Nevýhodou a problémom je absencia globálneho pohľadu, z čoho vyplýva nízka presnosť.
- **Algoritmy zhora nadol** – princíp algoritmov spočíva v rekurzívnom rozdeľovaní segmentov, kým nie sú splnené vopred stanovená kritériá zastavenia. Algoritmy začínajú s najväčším možným segmentom. Nevýhodou je nutnosť opakovania procesu nad všetkými údajmi pri pridaní nových bodov.
- **Algoritmy zdola nahor** – princíp algoritmov spočíva v rekurzívnom zlučovaní segmentov, kým nie sú splnené vopred stanovené kritériá zastavenia. Algoritmy začínajú s najmenším možným segmentom. Nevýhoda je zhodná s predošlým typom algoritmov.

Všetky tri triedy teda majú isté nevýhody. Riešením nevýhod je **algoritmus SWAB (Sliding-Window And Bottom-Up)**, ktorý už ako názov napovedá, je kombináciou algoritmov kĺzavého okna a algoritmov zdola nahor. Algoritmus dokáže aproximovať segment novopridaných hodnôt bez nutnosti opätovnej aproximácie predošlých segmentov a dosahuje presnosť približnú algoritmov zdola nahor. [27]

Reprezentáciu údajov v rámci segmentu je možné realizovať niekoľkými spôsobmi:

- **Agregačná aproximácia (*PAA*)** – aproximácia segmentu je vykonaná aplikáciou agregačných funkcií, ako napríklad funkciou 3.2.

$$\overline{x}_i = \frac{M}{n} \sum_{j=n/M(i-1)+1}^{(n/M)i} x_j, \tag{3.2}$$

  kde je $X = (x_1, \ldots, x_n)$ časový rad dĺžky n, $\overline{X} = (\overline{x}_1, \ldots, \overline{x}_M)$ je segment dĺžky M a $M \leq n$. Okrem uvedeného vzorca existuje viacero ďalších prístupov pre výpočet. [29]

- **Adaptívna konštantná aproximácia (*APCA*)** – aproximácia segmentu je vykonaná rovnako ako pri *PAA*. Hlavným rozdielom od *PAA* je, že *APCA* dokáže pracovať so segmentami s premenlivou dĺžkou. [10]
- **Aproximácia polynómom** alebo akoukoľvek **spojitou** a **diferencovateľnou funkciou** [46].

3.2.3 Dekompozícia

Vykonanie rozkladu časového radu je označované ako dekompozícia. Motiváciou pre realizáciu dekompozície je jednoduchšia identifikácia pravidelností v porovnaní s nerozloženým stavom časového radu. Identifikácia pravidelností je následne prínosná pre vykonanie krátkodobej a dlhodobej predikcie v rozumnej kvalite. Dekompozíciu časového radu je možné vykonať podľa jedného z nasledovných modelov [12]:

- **Aditívny model** – hodnoty časového radu sú funkciou súčtu základných zložiek ($Y = T + S + C + N$).
- **Multiplikatívny model** – hodnoty časového radu sú funkciou súčinu základných zložiek $Y = T \times S \times C \times N$.

Jednotlivé zložky znamenajú [21]:

- **Trend *T*** – zachytáva smer/zmenu vývoja časového radu z dlhodobého hľadiska v podobe priamky/krivky (dlhodobý rast/pokles). **Odhad** je možné realizovať:
  - manuálne – cez hodnoty radu je preložená krivka,
  - aproximáciou využitím metódy najmenších štvorcov – pre hodnoty radu je nájdená krivka, ktorá minimalizuje súčet štvorcov odchyliek od krivky,
  - metódou pohyblivého priemeru radu n – počíta priemer z n hodnôt rady.

  Jeho **odstránenie** je následne možné vykonať:
  - odčítaním komponenty trendu, priamky najlepšieho odhadu alebo priemeru,
  - aplikáciou filtra (*Baxter-King* filter, *Hodrick-Prescott* Filter...).
- **Sezónna zložka *S*** – zachytáva takmer identické vzory, ktoré sa vyskytujú v rovnakých úsekoch obdobia. Odhad z časového radu Y je možné vykonať napr. výpočtom empirického sezónneho indexu podľa vzťahu 3.3.

$$I_j = 1/k \sum_i Y_{ij} T_{ij}, \tag{3.3}$$

  kde T_{ij} sú trendové zložky (i-tý rok, j-tý mesiac), k je počet období.

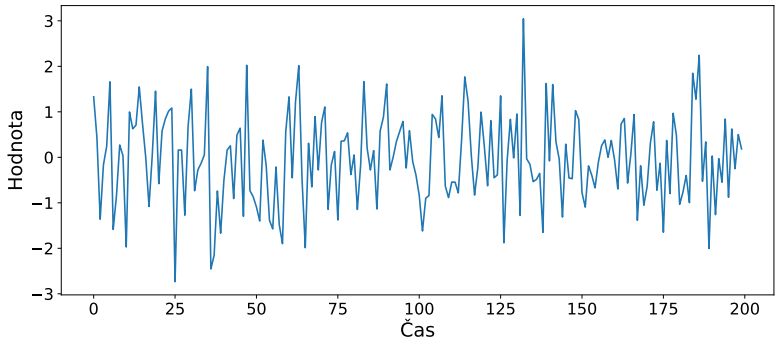
  Odstránenie je následne možné vykonať:
  - podelením sezónnym indexom – rad podelený hodnotou sezónneho indexu,
  - sezónnym rozdielom – rozdiel hodnoty predchádzajúcej a aktuálnej sezóny,
  - vyhladením s využitím kĺzavého priemeru (dĺžka okna rovná dĺžke sezóny).
- **Cyklická zložka *C*** – zachytáva dlhodobé oscilácie okolo priamky/krivky trendu. Ak sa vyskytuje v časovom rade aspoň približná periodicita, je možné skonštruovať cyklický index. Výpočet je realizovaný podobne ako sezóny index.
- **Náhodná/reziduálna zložka *N***– zachytáva nepravidelné alebo náhodné/reziduálne pohyby. Nepravidelnosti je možné získať odstránením trendu, sezónnosti a cyklickosti.

3.2.4 Analýza stacionarity

Prieskum stacionarity je nepovinným, ale zároveň jedným z najdôležitejších krokov predspracovania časových radov. Nad stacionárnym časovým radom je totiž možné vytvárať jednoduchšie a presnejšie modely.

Striktne stacionárny časový rad je rad, ktorého hodnoty nie sú funkciou času. Formálne povedané, hodnoty radu $(y_t, \ldots, y_{t-h})$ nie sú závislé na časovej perióde t, ale iba na oneskorení h. Všetky štatistické charakteristiky ako stredná hodnota, rozptyl a autokorelácia sú konštantné v čase a časový rad je naviac bez sezónnych vplyvov. **Nestacionárnym časovým radom** je časový rad, ktorý nespĺňa podmienky slabej (kovariančnej) stacionarity. **Slabá stacionarita** vyžaduje rovnakú konečnú nepodmienenú strednú hodnotu a rozptyl vo všetkých časových okamihoch a autokovarianciu nezávislú od času. [34]

Pre niektoré modely platí, že sú validné iba za predpokladu slabej stacionarity. Za takýchto okolností sa stáva prieskum stacionarity a následný prevod povinnosťou. Príkladom demonštrujúcim výhodnosť stacionarity sú prediktívne modely. Prediktívne modely sú lineárne regresné modely a tie lepšie pracujú nad údajmi, ktoré nie sú korelované. Stacionárny časový rad spĺňajúci vyššie uvedené podmienky je vizualizovaný na obrázku 3.5.



Obr. 3.5: Stacionárny časový rad

Test stacionarity časového radu je možné vykonať rôznymi spôsobmi [31]:

- **Vykreslenie a analýza** časového radu.
- **Štatistické testy** – časový rad je rozdelený na spojité časti, pre ktoré sú vypočítané štatistiky: stredná hodnota, rozptyl a autokorelácia. Ak sú hodnoty pre jednotlivé časti odlišné, tak je časový rad pravdepodobne nestacionárny. Pre tento test existuje viacero implementácií: *ADH* Test, *KPSS* test, *PP* test atď.

Takmer každý časový rad je možné transformovať na stacionárny. Spôsobov prevodu nestacionárneho radu na stacionárny je viacero a je možné ich kombinovať [31]:

- **Rozdiel predošlej hodnoty od aktuálnej hodnoty** – cez celý časový rad je vykonaný rozdiel $y_t = y_t - y_{t-1}$. Proces je možné opakovať viacnásobne.
- **Výpočet logaritmu** – každá hodnota časového radu je prevedená na jej logaritmus.
- **Výpočet *n*-tej odmocniny** – každá hodnota časového radu je prevedená na jej n-tú odmocninu.

3.2.5 Analýza korelácie

V časových radoch existuje potenciál pre koreláciu (podobnosť/závislosť) medzi pozorovanými hodnotami. Na základe prítomnosti korelácie v časových radoch je možné odhadnúť vhodnosť predpovede. Všeobecne platí, že čím je väčšia korelácia, tým je jednoduchšie vykonať predpoveď budúcich hodnôt. V rámci časových radov sú rozlišované tri typy korelácie [40]:

- **Krížová korelácia** – korelácia hodnôt dvoch odlišných časových radov X a Y dĺžky T. Výpočet je možné vykonať podľa vzorca 3.4.

$$r_{xy} = \frac{\sum_{t=k+1}^{T}(x_t - \overline{x})(y_{t-k} - \overline{y})}{\sqrt{\sum_{t=1}^{T}(x_t - \overline{x})^2 \sum_{t=1}^{T}(y_t - \overline{y})^2}}, \tag{3.4}$$

  kde $\overline{x}$, $\overline{y}$ sú stredné hodnoty časových radov.

- **Autokorelácia** – korelácia aktuálnej hodnoty na predošlých hodnotách časového radu s jasne definovaným oneskorením k, pričom sú brané do úvahy aj korelácie s oneskorením 1 až $k-1$. Autokoreláciu je možné vypočítať podľa vzorca 3.5.

$$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \overline{y})(y_{t-k} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2}, \tag{3.5}$$

  kde $\overline{y}$ je stredná hodnota časového radu dĺžky T.

- **Čiastočná autokorelácia** – korelácia aktuálnej hodnoty na predošlých hodnotách časového radu s jasne definovaným oneskorením k, pričom nie sú brané do úvahy korelácie s oneskorením 1 až $k-1$. Získaná je ako koeficienty α_k autoregresnej funkcie Y_t vo vzorci 3.6.

$$Y_t = \alpha + \alpha_1 Y_{t-1} + \cdots + \alpha_k Y_{t-k} \tag{3.6}$$

Koreláciu medzi hodnotami je možné okrem uvedených výpočtov analyzovať aj vizualizáciou s využitím rôznych grafov, ako napr. graf oneskorení atď. Rovnako je vhodné uviesť, že analýza korelácie je len jednou z mnohých analýz, ktoré sa vykonávajú pre odhad vhodnosti predpovede.

3.2.6 Analýza chýbajúcich údajov

Chýbajúce údaje sú v časovom rade, tak ako vo väčšine prípadov, brané ako negatívum a nedostatok, ktorý je potrebné odstrániť. Najjednoduchšou technikou vysporiadania sa s chýbajúcimi hodnotami je ich odstránenie. Ide však o techniku, pri ktorej sa stráca významné množstvo informácie. Častejším prístupom je teda technika doplnenia. Metód doplnenia chýbajúcich hodnôt v časových radoch je viacero a do istej miery sa odlišujú od tradičných prístupov. Ako príklady metód je vhodné uviesť [37]:

- **Dopredné/Spätné doplnenie** – chýbajúca hodnota je doplnená predošlou/nasledovnou hodnotou.

- **Lineárna/Kvadratická/Kubická interpolácia** – chýbajúca hodnota je doplnená metódou lineárnej/kvadratickej/kubickej interpolácie

- **Priemer najbližších susedov** – chýbajúca hodnota je doplnená priemerom susedných hodnôt.
- **Priemer sezónnych/cyklických náprotivkov** – chýbajúca hodnota je doplnená priemerom sezónnych/cyklických náprotivkov.
- **Prediktívny model** – chýbajúca hodnota je doplnená tradičným prediktívnym modelom (metóda *náhodný les*, metóda *k-najbližších susedov* atď.).
- **Predpoveď/Spätná predpoveď** – chýbajúca hodnota je doplnená modelmi predpovede časových radov. Detailnejší popis týchto metód je uvedený v podkapitole 3.3.1.

Konkrétny prístup je vybratý v závislosti na vlastnostiach zdrojových údajov a požadovanej presnosti. Ak sú prítomné doplňajúce atribúty, je vhodné použiť tradičný prediktívny model. Ak je dostatok predošlých hodnôt, odporúča sa doplniť chýbajúce hodnoty modelom predpovede časových radov. Ak je dostatok nasledovných hodnôt, je vhodné chýbajúce hodnoty doplniť modelom spätnej predpovede časových radov. Ak neplatí nič z predošlo uvedeného, je vhodné vykonať predpoveď náprotivkov z predchádzajúcich cyklov.

3.2.7 Analýza odľahlých údajov

V rámci časových radov je často možné z globálneho alebo lokálneho hľadiska nájsť prítomnosť odľahlých údajov. Identifikácia a spracovanie odľahlých dát je veľmi dôležitá, nakoľko tieto údaje môžu mať negatívny vplyv na presnosť ďalších analýz a dolovacích úloh.

V časových radoch sú rozlišované dva typy odľahlých údajov: odľahlý bod a odľahlá podsekvencia bodov. **Odľahlý bod** je jeden neobvyklý záznam v špecifickom čase [50]. **Odľahlá sekvencia** je súvislá postupnosť bodov v špecifickom čase, ktorá je z pohľadu ostatných sekvencií neobvyklá [28]. Samotné objekty, ktoré sú jej súčasťou, však nemusia byť odľahlé. Vo zvyšku tejto sekcie je venovaná pozornosť na identifikáciu a spracovanie odľahlých bodov. Existuje niekoľko spôsobov, ako **identifikovať** odľahlé body [50]:

- **Štatistickým modelom** – za odľahlý bod je považovaný bod, ktorý patrí mimo rozsah vypočítaných štatistík priemeru, mediánu, štandardnej odchýlky atď. Najvyššia a najnižšia hranica je obvykle definovaná manuálne na základe analýz.
- **Modelom predpovede** – za odľahlý bod je považovaný bod, ktorý patrí mimo interval spoľahlivosti netrénovaného modelu predpovede.
- **Modelom zhlukovania** – za odľahlý bod je považovaný bod, ktorý je ďaleko vzdialený od vytvorených zhlukov.

Spracovanie odľahlých bodov sa môže v jednotlivých situáciách odlišovať. Časový rad sa od odľahlých bodov snažíme **očistiť**, ak ich prítomnosť interpretujeme ako negatívum. Existujú však aj situácie, kedy sú odľahlé body oblasťou záujmu. V takom prípade sú odľahlosti podrobené ďalšej **analýze** podľa potreby.

3.3 Dolovanie z časových radov

Úlohou tejto podkapitoly je predstaviť základné úlohy dolovania z časových radov. Konkrétne sú popísané úlohy predpovede, klasifikácie, zhlukovania a ďalších obvyklých úloh. Pozornosť v popise jednotlivých úloh je venovaná hlavne na vysvetlenie najpoužívanejších metód.

3.3.1 Predpoveď

Predpoveď budúcich hodnôt časového radu je najobvyklejšou úlohou dolovania z časových radov. Využitie je prínosné v takmer všetkých oblastiach v podobe podpory plánovania, rozhodovania, výroby a množstva ďalších činností. Je však veľmi dôležité, aby boli prognózy presné, akékoľvek chyby v prognózach môžu mať fatálne následky na úspech. Predpoveď časového radu je z pohľadu počtu premenných delená do dvoch kategórií [47]:

- **Jednopremenná predpoveď** – predpoveď hodnôt je vykonaná iba na základe predošlých hodnôt časového radu.
- **Viacpremenná predpoveď** – predpoveď hodnôt časového radu je vykonaná na základe viacerých prediktorov (dekomponované zložky, paralelne zbierané premenné, viaceré časové rady atď.).

Pred samotnou predpoveďou časového radu je obvykle realizovaný **odhad vhodnosti predpovede** časového radu. Zrejme najobvyklejšou technikou odhadu vhodnosti je analýza korelácie, viď sekcia 3.2.5. Nejde však o jediný spôsob, nad časovými radmi je typicky takisto realizovaná analýza stacionarity alebo analýza pravidelnosti a nepredvídateľnosti kolísania hodnôt pomocou metrík *Approximate entropy* a *Sample entropy* [41]. Vhodnosť iného časového radu pre predpoveď je možné zistiť pomocou *Grangerovho* testu [20].

Na konci procesu predpovede časového radu býva obvykle realizovaná **validácia predpovede**, ktorú je opäť možné vykonať viacerými spôsobmi. Jedným z nich je krížová validácia, kde je časový rad rozdelený na trénovaciu a testovaciu podsekvenciu. Rozdelenie je typicky vykonané viacnásobne pre rôzny pomer dĺžky podsekvencií. Trénovacie údaje sú použité pre natrénovanie modelu. Testovacie údaje sú použité pre zistenie presnosti na základe porovnania s predikovanými hodnotami. [5]

Naivné metódy

Ide o najzákladnejšie a najjednoduchšie metódy predpovede časového radu. Typicky sú využívané iba ako referenčné modely pre porovnanie presnosti s pokročilejšími modelmi.

Naivná metóda predpovedá budúce hodnoty Y_t na základe poslednej pozorovanej hodnoty Y_T. Ide o veľmi nepresné riešenie, ktoré je možné rozšíriť a spresniť na sezónny naivný model. [11]

Sezónna naivná metóda predpokladá, že časový rad má sezónnu zložku s dĺžkou periódy S. Predpoveď je realizovaná podľa vzťahu $Y_t = Y_{t-S}$. Ide teda o predpoveď na základe hodnôt posunutých o S časových krokov dozadu. [11]

Metódy spriemerovania

Skupina týchto metód predpovedá budúce hodnoty na základe priemeru minulých hodnôt. Obvykle dosahujú o niečo lepšie výsledky ako naivné metódy, avšak stále sú považované za pomerne nepresné. Spôsob výberu hodnôt pre spriemerovanie sa môže líšiť. Najzákladnejší realizuje **jednoduchá metóda spriemerovania**. Budúca hodnota Y_t je predpovedaná ako priemer všetkých hodnôt časového radu. Výpočet je realizovaný podľa vzťahu 3.7.

$$Y_t = \frac{Y_1 + Y_2 + ... + Y_T}{T} \tag{3.7}$$

Metóda kĺzavého spriemerovania uvažuje v spriemerovaní iba posledných n hodnôt. Výpočet je vykonaný podľa vzorca 3.8.

$$Y_t = \frac{Y_{T-(n-1)} + ... + Y_{T-1} + Y_T}{N} \tag{3.8}$$

Metóda kĺzavého sezónneho spriemerovania kombinuje kĺzavé spriemerovanie dĺžky n so sezónnosťou dĺžky m. Budúce hodnoty sú teda počítané obdobne ako v prípade predošlého vzorca. Zmenou je, že je do úvahy braných posledných n hodnôt s rozstupom m.

Metódy exponenciálneho vyhladzovania

Metódy exponenciálneho vyhladzovania sú jednou z najviac úspešných metód predpovede. Základným variantom je metóda ***SES (Simple Exponential Smoothing)***, ktorá modeluje ďalšiu hodnotu v sekvencii ako exponenciálne váženú lineárnu funkciu pozorovaných hodnôt v predchádzajúcich časových krokoch, viď vzorec 3.9:

$$Y_{t+1} = \alpha Y_t + \alpha(1-\alpha)Y_{t-1} + \alpha(1-\alpha)^2 Y_{t-2} + \ldots, \tag{3.9}$$

kde $0 < \alpha < 1$ je koeficient udávajúci vplyv minulých hodnôt na predpoveď. Zo vzorca je možné vidieť, že váhy sa smerom do minulosti znižujú exponenciálne. Použitie metódy je vhodné pre jednorozmerné časové rady bez trendových a sezónnych zložiek. [7]

Ak časový rad obsahuje trend, je vhodné využiť vylepšenú metódu ***HES (Holt Exponential Smoothing)***. Ak časový rad obsahuje trend a sezónnosť, je vhodné využiť vylepšenú metódu ***HWES (Holt Winter's Exponential Smoothing)***. Metódy pri predpovedi totiž zohľadňujú trendovú, resp. trendovú a sezónnu zložku. [7]

Metódy z rodiny *SARIMAX*

Tak ako názov napovedá, pod pojmom *SARIMAX* sa skrýva niekoľko metód pre predpoveď časových radov. Jednotlivé metódy je možné použiť samostatne alebo nabaľovať a spájať tak, že vznikne finálny najkomplexnejší model *SARIMAX*.

Metóda ***AR (Auto Reggresion)*** modeluje ďalšiu hodnotu v sekvencii ako lineárnu funkciu pozorovaných hodnôt v predchádzajúcich časových krokoch. Výpočet je vykonaný podľa vzorca 3.10:

$$Y_t = \alpha + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \cdots + \alpha_p Y_{t-p} + e_t, \tag{3.10}$$

kde $0 < \alpha < 1$ je koeficient udávajúci vplyv minulých hodnôt na predpoveď, p definuje, koľko predchádzajúcich hodnôt je braných do úvahy pri odhade lineárnej funkcie a e_t je chyba modelu. [47]

Metóda **MA (Moving Average)** modeluje ďalšiu hodnotu v sekvencii ako lineárnu funkciu chýb modelu AR v predchádzajúcich časových krokoch. Výpočet je vykonaný podľa vzorca 3.11:

$$Y_t = \beta + \beta_1 e_{t-1} + \beta_2 e_{t-2} + \cdots + +\beta_p e_{t-q} + e_t, \tag{3.11}$$

kde $0 < \beta < 1$ je koeficient udávajúci vplyv minulých chýb na predpoveď a q definuje, koľko predchádzajúcich hodnôt je braných do úvahy pri odhade lineárnej funkcie. [47]

Metóda ***ARMA (Autoregressive Moving Average)*** spája modely *AR* a *MA*. Použitie týchto metód je vhodné pre jednorozmerné časové rady bez trendových a sezónnych zložiek. [47]

Metóda ***ARIMA (Autoregressive Integrated Moving Average)*** rozširuje predošlú metódu o prevod časového radu na stacionárny označovaný ako integrácia. Vstupom metódy sú opäť rovnaké parametre s pridaním parametru *d*. Parameter *d* je minimálny počet rozdielov potrebných na to, aby sa časový rad stal stacionárnym. Použitie metódy je vhodné pre jednorozmerné časové rady s trendom a bez sezónnych zložiek. [47]

Metóda ***SARIMA (Seasonal Autoregressive Integrated Moving-Average)*** aplikuje metódu *ARIMA* aj na sezónne hodnoty. Metóda má šesť vstupných parametrov, pričom prvé tri sú zhodné s predošlým modelom a ostatné špecifikujú použitie *ARIMA* na sezónne hodnoty. Použitie metódy je vhodné pre jednorozmerné časové rady s trendovými aj sezónnymi zložkami. [51]

Metóda ***SARIMAX (Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors)*** je rozšírením metódy *SARIMA*, pričom navyše berie do úvahy aj exogénne premenné. Exogénne premenné sú vstupné sekvencie, ktoré sú pozorované v rovnakých časových okamihoch ako pôvodný rad. Primárny časový rad je označovaný ako endogénna premenná. Pozorované hodnoty exogénnych premenných sú zahrnuté priamo v každom časovom kroku modelu a nie sú modelované rovnakým spôsobom ako primárne časové rady. Použitie metódy je vhodné pre jednorozmerné časové rady s trendovými, sezónnymi zložkami a exogénnymi premennými. [51]

Ostatné varianty metód

V rámci tejto sekcie sú popísané ďalšie pokročilejšie metódy predpovede, ktoré stavajú na vyššie uvedených. Predchádzajúce metódy predpokladajú, že chybové členy majú konštantný rozptyl. Metóda ***GARCH (Generalized Autoregressive Conditional Heteroskedasticity)*** predpokladá, že rozptyl chýb sa môže meniť v čase, čo je užitočné najmä pri predpovedi časových radov vo financiách, kde sa volatilita v čase mení. [4]

Metóda ***TBATS (Trigonometric, Box-Cox transform, ARMA errors, Trend and Seasonal components)*** je metóda predpovede, ktorá je založená na exponenciálnom vyhladzovaní. Hlavnou črtou modelu *TBATS* je jeho schopnosť vysporiadať sa s viacerými sezónnosťami v rámci jedného časového radu. Každú sezónnosť modeluje pomocou trigonometrickej reprezentácie založenej na Fourierovom rade. Alternatívnou metódou pre predpoveď časového radu s viacerými sezónnosťami je metóda ***Prophet***. [53]

Ak je snaha vykonať predpoveď pomocou neurónových sietí, je možné využiť siete *LSTM* alebo *CNN*. ***LSTM*** **(Long-Short Term Memories)** je rekurentný typ siete, kde je stav reprezentovaný vektorom. Sieť umožňuje sledovať závislosti aj so staršími hodnotami. ***CNN*** **(Convolutional Neural Network)** je neurónová sieť, ktorej odlišnosť spočíva v implementácii operácie konvolúcie v jednej z vrstiev. Na konvolučnú vrstvu nadväzuje redukčná *pooling* vrstva. Tento typ siete sa obvykle využíva pre *2D* obrazové údaje, uplatnenie však nachádza aj pre *1D* časové rady. Metódy neurónových sietí sú zaraďované medzi komplexné metódy predpovede a len zriedka sa používajú na predpoveď jedného časového radu. Na odhadovanie totiž vyžadujú veľké množstvo údajov. [55]

3.3.2 Klasifikácia

Ďalšou častou úlohou dolovania z časových radov je klasifikácia. Úlohou klasifikácie je zaradiť časový rad do jednej z preddefinovaných tried. Zaradenie časového radu je realizované klasifikačným modelom, ktorý je natrénovaný na množine časových radov s označenou triedou. [19]

Existuje množstvo metód vytvorenia klasifikačného modelu. Aký klasifikátor je vhodné zvoliť pre konkrétnu úlohu? To je problém, ktorý je potrebné riešiť veľmi často. Pri výbere klasifikátora časových radov je odporúčané zvážiť, akú presnosť a časovú/pamäťovú zložitosť požadujeme a akú reprezentáciu majú zdrojové údaje.

Metódy založené na vzdialenosti

Táto sekcia čerpá z [49]. Vzdialenostné klasifikátory používajú na určenie triedy metriky vzdialenosti. Ako metriku je všeobecne možné zvoliť akúkoľvek zmysluplnú vzdialenostnú funkciu. Obvykle je však zvolená Euklidova vzdialenosť alebo *DTW (Dynamic Time Warping)* vzdialenosť. Veľkou výhodou metriky *DTW* je schopnosť určovať podobnosť medzi sekvenciami s rôznym časom, frekvenciou alebo dĺžkou zberu.

Algoritmus ***KNN* (*k*-najbližších susedov)** je považovaný za jeden z najzákladnejších klasifikátorov tejto skupiny. Klasifikátor typicky slúži ako referenčný model na hodnotenie ostatných klasifikácií, nakoľko je jednoduchý, robustný a nevyžaduje zložité ladenie vstupných parametrov. Obsahuje však aj niekoľko nedostatkov. Konkrétnym je pamäťová a časová zložitosť. Počas klasifikácie totiž porovnáva každý objekt so všetkými ostatnými objektmi v trénovacej sade. Okrem toho poskytuje veľmi malé množstvo informácie o tom, prečo bol časový rad priradený do danej triede. Algoritmus rovnako môže dosahovať neuspokojivé výsledky pri zašumených časových radoch. Šum v časovom rade zapríčiňuje jemné rozdiely, ktoré môžu mať rozhodujúci vplyv pri rozlišovaní triedy.

Metódy založené na intervaloch

Klasifikátory tejto skupiny používajú na určenie členstva v triede informáciu obsiahnutú v intervaloch rozdeleného časového radu. Konkrétnym príkladom je metóda ***TSF (Time Series Forest)***, ktorá je modifikáciu algoritmu náhodného lesa pre časové rady. Princíp spočíva v rozdelení radu do náhodných intervalov (náhodná pozícia a náhodná dĺžka), pričom z každého intervalu sú extrahované súhrnné charakteristiky (stredná hodnota, štandardná odchýlka a sklon). Po extrakcii je vykonané trénovanie rozhodovacieho stromu, ktorý realizuje samotnú klasifikáciu. Proces sa opakuje, pokým nie je vytvorený požadovaný počet stromov alebo kým nevyprší čas výpočtu. Klasifikácia je realizovaná na základe väčšinového hlasovania. Vyberaná je teda trieda, ktorú predikovalo najviac stromov lesa. [14]

Výhodami algoritmu *TSF* je presnosť a rýchlosť. Algoritmus rovnako vracia presnejšie výsledky ako ostatné základné klasifikátory. Ďalším plusom algoritmu je jeho interpretovateľnosť. Z vytvoreného modelu je jednoduché vyčítať dôležitosť jednotlivých časových intervalov pre finálny výber triedy.

Metódy založené na slovníku slov

Princíp klasifikátorov založených na slovníku spočíva v transformácii časového radu na sekvenciu samostatných slov. Klasifikácia je následne založená na ich distribúcii, pričom môže byť použitý akýkoľvek klasifikátor. Použitie tejto skupiny metód je výhodné, ak je možné rozlišovať frekvenciu výskytov podsekvencií v časovom rade.

V metóde ***BOSS (Bag of SFA Symbols)*** sú „slová“ zo sekvencie extrahované pomocou transformácie *SFA (Symbolic Fourier Approximation)*. Jej princíp spočíva vo výpočte Fourierovej transformácie okna dĺžky w a následnej diskretizácii na symboly slova algoritmom *MCB (Multiple Coefficient Binning)*. Pri posúvaní kĺzavého okna je súbežne vytváraný slovník slov, ktorý zaznamenáva frekvenciu každého slova. Slovo sa započíta len raz v prí-

pade, že bolo vytvorené dvoma alebo viacerými oknami idúcimi po sebe. Po dokončení je na základe slovníka slov vytvorený histogram. [44]

Metóda *BOSS* má množstvo ďalších variantov a často sa používa skupinovo. Metóda ***cBOSS (Contractable BOSS)*** je časovo a pamäťovo efektívnejšia verzia, pričom platí, že presnosť klasifikácie nie je výrazne negatívne ovplyvnená. Metóda ***BOSSVS (BOSS in Vector Space)*** k prevodu pridáva sčítanie výsledných histogramov tried a následný prevod na *tf-idf* vektory. Pri klasifikácii sa následne *tf-idf* vektory porovnávajú s *tf-idf* vektormi vstupných vzorkov.

Metódy založené na frekvencii

Klasifikačné modely založené na frekvencii klasifikujú rad na základe frekvencie extrahovaných údajov. Použitie tejto skupiny metód je výhodné, ak je možné rozlišovať frekvenciu výskytov podsekvencií v časovom rade.

Metóda ***RISE (Random Interval Spectral Ensemble)*** je populárny variant metódy *TSF*, pričom má dve odlišnosti. Klasifikačný strom zodpovedá jednému intervalu časového radu a namiesto súhrnných charakteristík sú z intervalu extrahované spektrálne/frekvenčné charakteristiky. Konkrétne autoregresné koeficienty, autokorelačné koeficienty a koeficienty Fourierovej transformácie. Pravdepodobnosť zaradenia do triedy je vypočítaná ako podiel počtu kladných výsledkov pre triedu voči celkovému počtu rozhodovacích stromov. [18]

Metódy založené na tvaroch

Snahou klasifikátorov založených na tvaroch je nájsť v trénovacej sade tvary s najvyššou mierou odlišnosti. Tvary sú v kontexte časových radov chápané ako podsekvencie, pomocou ktorých je možné detekovať lokálne, fázovo nezávislé podobnosti. Pri klasifikácii je výber triedy časového radu realizovaný na základe prítomnosti identifikovaných tvarov. Použitie tohto typu klasifikácie je výhodné, ak najlepšou extrahovanou informáciou z časových radov je prítomnosť alebo absencia fázovo nezávislej podsekvencie.

Konkrétnym príkladom je metóda ***STC (Shapelet Transform Classifier)***. Prvým krokom metódy je identifikácia k tvarov s najvyššou mierou odlišnosti v množine trénovacích údajov. Nasleduje výpočet miery prítomnosti k tvarov v časovom rade, ktorý má byť klasifikovaný. Miera prítomnosti konkrétneho tvaru sa vypočíta ako vzdialenosť tvaru od časového radu. Časový rad, ktorý má byť klasifikovaný, je po tomto kroku reprezentovaný vektorom dĺžky k. To znamená, že na klasifikáciu môže byť použitý akýkoľvek klasifikačný model podporujúci prácu s vektormi. [33]

Metódy hlbokého učenia

Klasifikácia časových radov je často realizovaná aj s využitím techník hlbokého učenia. Modely tejto skupiny klasifikujú rad pomocou viacvrstvových neurónových sietí. Metódy ***LSTM (Long Short Term Memory)*** a ***CNN (Konvolučné neurónové siete)*** sú považované za účinné, nakoľko pri klasifikácii berú do úvahy informáciu o časovom usporiadaní. Aj napriek tomu však majú neurónové siete isté nedostatky, ktoré ich robia nevhodnými pre množstvo klasifikačných úloh. Ide napr. o výber efektívnej architektúry, ladenie vstupných parametrov, nedostatok údajov (neurónové siete potrebujú rozsiahle trénovacie sady) alebo pomalý proces trénovania. [26]

3.3.3 Zhlukovanie

Zhluková analýza je ďalšou obvyklou úlohou dolovania časových radov. Jej cieľom je nájsť skupiny časových radov, pre ktoré platí, že časové rady v rámci nich sú maximálne podobné a zároveň minimálne podobné s radmi inej skupiny (zhluku) [19].

So zhlukovaním časových radov sa spája veľa problémov. Hlavným je výber vhodnej reprezentácie časového radu a voľba vhodnej vzdialenostnej funkcie. Vzhľadom na reprezentáciu časového radu sú zhlukovacie metódy delené do niekoľkých kategórií: zhlukovanie surových údajov, zhlukovanie extrahovaných/transformovaných údajov, zhlukovanie modelov. Jednotlivé kategórie zhlukovania časových radov sú popísané v ďalších sekciách.

Ďalším riešeným problémom pri úlohe zhlukovania je spôsob vyhodnotenia výsledkov. Pri vyhodnotení je možné vyberať z rôznych metrík, ktoré v sebe typicky zapuzdrujú posúdenie kompaktnosti a oddeliteľnosti zhlukov. Príkladom takýchto metrík sú *SS (Silhouette Score)*, *CHS (Calinski-Harabasz Score)*, *DBS (Davies-Bouldin Score)*.

Metrika ***SS (Silhouette Score)***, ktorá je počítaná podľa vzťahu 3.12.

$$SS = \frac{b - a}{max(a, b)}, \tag{3.12}$$

kde a je priemerná vzdialenosť bodu s ostatnými bodmi v rovnakom zhluku a b je priemerná vzdialenosť bodu s bodmi v najbližšom zhluku. Metrika nadobúda hodnoty z intervalu $\langle -1, 1 \rangle$, pričom čím je hodnota vyššia, tým je zhlukovanie presnejšie. [42]

Metrika ***CHS (Calinski-Harabasz Score)*** je počítaná ako pomer súčtu disperzie (súčet štvorcových vzdialeností) medzi zhlukmi a disperzie v rámci zhluku pre všetky ostatné zhluky. Čím je hodnota metriky vyššia, tým je zhlukovanie presnejšie. [9]

Metrika ***DBS (Davies-Bouldin Score)*** porovnáva priemernú vzdialenosť bodov zhluku s veľkosťou zhluku. Najnižšie možná hodnota je *0*, pričom hodnoty bližšie k nej značia presnejšie zhlukovanie. [13]

Metódy surových údajov

Základným typom údajov, nad ktorými je možné realizovať zhlukovú analýzu, sú **naivné príznaky**. Ide o surové údaje časového radu, ktoré nie sú žiadnym spôsobom predspracované. Pre tento typ príznakov je možné zvoliť akúkoľvek vzdialenostnú funkciu numerického charakteru. Zhlukovanie surových údajov je výhodné realizovať, ak sú medzi časovými radmi rozdiely v úrovni/výške. [32]

Najzákladnejšou a najznámejšou funkciou je **Euklidova vzdialenosť**. Vzdialenosť časových radov X a Y je počítaná podľa vzťahu 3.13.

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.13}$$

Nevýhodou funkcie je, že vyžaduje, aby boli časové rady zarovnané. Práve preto je pri časových radoch rôznej dĺžky a neznámeho posunu používaná vzdialenostná funkcia ***DTW (Dynamic Time Warping)***, ktorá tento problém odstraňuje. Jej nevýhodou však je vysoká časová náročnosť, z čoho vyplýva, že najlepším scenárom pre realizáciu zhlukovania nad surovými údajmi je, ak sú časové rady rovnako dlhé a zarovnané.

Metódy extrahovaných/transformovaných údajov

Zhlukovacie metódy tejto skupiny prijímajú na vstupe údaje, ktoré sú istým spôsobom extrahované/transformované zo surových údajov. Typ extrakcie/transformácie sa môže odlišovať. Obvyklým typom extrahovaných príznakov sú **hodnoty autokorelácie** a **hodnoty čiastočnej autokorelácie** pre rôzne oneskorenia. Pre porovnanie časových radov X a Y je v takomto prípade možné zvoliť funkciu zo vzorca 3.14.

$$T_{n,m} = n \sum_{k=1}^{m} (r_{X,k} - r_{Y,k})^2, \tag{3.14}$$

kde $r_{X,k}$ a $r_{Y,k}$ sú odhadované autokorelácie pre oneskorenie k. Metrika je však platná, iba ak sú zhlukované rady nezávislé. [1]

Iným typom príznakov môže byť **zastúpenie frekvencií** v časovom rade. Pre zisk takýchto príznakov je potrebné časové rady previesť do frekvenčnej/spektrálnej domény. Väčšina vzdialenostných funkcií pri frekvenčných príznakoch realizuje výpočty nad periodogramami, čo sú odhady frekvenčnej hustoty časového radu. [8]

Zhlukovanie je rovnako možné realizovať na **extrahovanými symbolmi**. Algoritmov extrakcie symbolov je množstvo, ako napr. metódy *ISC (Incremental Subsequence Clustering)* a *SAX (Symbolic Aggregate Approximation)*. Ako vzdialenostnú funkciu pre symboly je možné použiť napr. Hammingovu vzdialenosť, ktorá je definovaná ako súčet zhôd symbolov na rovnakej pozícii.

Metódy založené na modeloch

Tak ako už názov napovedá, princíp zhlukovania založeného na modeloch spočíva vo využití vytvorených modelov pre úlohu zhlukovania. Pri tomto type zhlukovania je potrebné definovať vzdialenostnú funkciu v priestore parametrov modelu. Pre *ARIMA* model je napríklad možné zaviesť **Piccolovu vzdialenostnú funkciu** definovanú vzorcom 3.15.

$$D(X,Y) = \sqrt{\sum_{j=1}^{\infty} (\pi_{X,j} - \pi_{Y,j})^2}, \tag{3.15}$$

kde X, Y sú časové rady, $\pi_{X,j}, \pi_{Y,j}$ sú koeficienty *ARIMA* modelu [36]. Okrem uvedenej vzdialenostnej funkcie existuje množstvo podobných pre rôzne ďalšie modely.

Ostatné varianty metód

Väčšina doposiaľ uvedených vzdialenostných funkcií realizuje zhlukovanie iba na základe historických pozorovaných hodnôt a je tak statická. Často však nastáva situácia, kedy je potrebné pri zhlukovaní brať do úvahy aj **budúce hodnoty** a zisťovať tak, ako budú vyzerať zhluky v budúcnosti. Existuje viacero vzdialenostných funkcií, ktoré berú ohľad aj na predpovedané hodnoty v budúcnosti.

Vyššie uvedené vzdialenostné funkcie pri hľadaní zhlukov takisto využívajú iba informáciu o časovom rade ako celku a ignorujú **podobnosť podsekvencií**. Existuje množstvo vzdialenostných funkcií, ktoré pri zhlukovaní berú do úvahy aj krížovú závislosť podsekvencií.

3.3.4 Ostatné typy úloh

V tejto podkapitole sú popísané ostatné dolovacie úlohy z časových radov, ktoré je vhodné uviesť.

Regresia je dolovacia úloha, ktorá je veľmi podobná úlohe predpovedi. Na základe časového radu sú opäť predikované číselné hodnoty. Predikované číselné hodnoty však nemajú súvislosť s budúcimi hodnotami časového radu. Ide o hodnoty nezávislej premennej. Pre realizáciu úlohy regresie je rovnako ako pri predpovedi časového radu možné na základe vzťahu hodnôt premenných využiť rôzne typy regresných funkcií. [47]

Cieľom úlohy **vyhľadávania podobností/odľahlostí** je nájsť množinu časových radov, ktoré sú podľa vopred stanovenej metriky čo najviac podobné/odľahlé časovému radu na vstupe. Pri úlohe vyhľadávania je potrebné riešiť vo vysokej miere rovnaké problémy ako v úlohe zhlukovania, konkrétne problémy výberu vhodnej reprezentácie časového radu a výberu vhodnej vzdialenostnej funkcie. [36]

Anotácia časového radu je dolovacia úloha, ktorej cieľom je označovať význam bodu alebo skupiny v časovom rade. Anotácia v sebe zahŕňa množstvo podúloh ako zhlukovanie, detekcia odľahlých/anomálnych bodov, detekcia lokálnych/globálnych extrémov atď. Metódy pre realizáciu tejto úlohy sa odlišujú na základe výberu anotačných podúloh. [52]

Posledným typom dolovacej úlohy uvedenej v tejto sekcii je **sumarizácia** časového radu. Cieľom úlohy je formou rôznych analýz, výpočtu štatistických hodnôt a extrakcie rôznych iných príznakov získať aproximačné informácie o časovom rade/radoch. Výsledkom je prehľadný súhrn. Metódy pre realizáciu tejto úlohy sa odlišujú na základe výberu informácií, ktoré budú obsiahnuté v súhrne. [19]

Kapitola 4

Python a jeho podpora pre časové rady

V tejto práci je ako implementačný nástroj pre dolovanie zvolený programovací jazyk *Python*. Úlohou tejto kapitoly je predstaviť jeho základné vlastnosti a podporu pre dolovanie z časových radov. Prvá časť kapitoly je venovaná všeobecnému predstaveniu jazyka *Python* a jeho vlastnostiam. Nasleduje sekcia s popisom základnej podpory jazyka pre dolovanie z dát. Záver kapitoly je zameraný na prezentáciu podpory jazyka v oblasti dolovania z časových radov.

4.1 Základný popis jazyka

Táto sekcia čerpá z [30] a [43]. *Python*[1] bol vytvorený ako verejný a slobodný projekt v roku 1991 Guidom Van Rossumom. Ide o jeden z najpopulárnejších programovacích jazykov na svete. Pravidelne sa umiestňuje v rebríčku komunity *TIOBE*[2] medzi piatimi najobľúbenejšími programovacími jazykmi.

Python je vysokoúrovňový, interpretovaný, multiparadigmatický programovací jazyk podporujúci procedurálne a objektovo orientované programovanie. Zápis programu je najskôr preložený do bajtkódu a potom interpretovaný vo virtuálnom stroji. Je dynamicky silne typovaný, čo znamená, že v priebehu prekladu prebieha iba syntaktická analýza a typová kontrola prebieha až za behu programu. V *Pythone* je všetko reprezentované ako objekt s vlastnými atribútmi a metódami.

Veľkou výhodou jazyka je jeho jednoduchá a jasná syntax. Výraznou syntaktickou odlišnosťou je využívanie bielych znakov tabulátora k odsadzovaniu. Príkazy sú oddeľované koncom riadka alebo bodkočiarkou. *Python* podporuje automatickú správu pamäte pomocou zabudovaného nástroja s názvom *Garbage collector*. Ďalšou výhodou jazyka je prenositeľnosť medzi takmer všetkými operačnými systémami. *Python* rovnako poskytuje množstvo voľne dostupných balíkov pre rôzne účely, viď ďalšie sekcie.

Jazyk má množstvo variantov implementácií, jeho aktuálnou verziou je *Python 3*. Verzia *2* je však aj v dnešnej dobe stále využívaná. Veľkou nevýhodou je nekompatibilita medzi verziami *2* a *3*. *Python* nie je vhodné používať na veľmi náročné výpočty alebo na platformy s nízkym množstvom dostupnej pamäte, nakoľko neefektívne pracuje s pamäťovým priestorom.

[1] https://www.python.org/
[2] https://www.tiobe.com/

4.2 Základná podpora pre dolovanie z dát

V tejto sekcii sú uvedené balíky, ktoré sú využívané ako základ pre dolovanie z údajov v programovacom jazyku *Python*. Popis je zameraný na balíky pre manipuláciu/spracovanie údajov, import/export údajov, vizualizáciu a aj samotné dolovanie. Konkrétne ide o balíky *Numpy*, *Pandas*, *Datetime*, *Matplotlib*, *Plotly*, *Scipy*, *Sklearn*. Okrem týchto balíkov existuje množstvo ďalších, uvedený zoznam však poskytuje absolútny základ.

Balík *NumPy*

Cieľom balíka *Numpy*[3] je zefektívniť prácu a výpočty nad numerickými údajmi. Základným dátovým typom balíka je dátový objekt homogénneho multidimenzionálneho poľa `ndarray`, ktorý môže reprezentovať časový rad. Jeho veľkosť je na rozdiel od ostatných iných polí v Pythone stanovená už pri inicializácii. Balík nad uvedeným dátovým typom implementuje množstvo rôznych metód, ako napr. vykonanie jednoduchých operácií, vzorkovanie, prácu s polynómami, lineárnu algebru atď. Poskytuje však aj podporu pre generovanie nových údajov, vyhľadávanie, radenie či prácu so súbormi. Takisto je vhodné uviesť, že balík je využívaný ako základ väčšiny ostatných balíkov pracujúcich s maticami.

Balík *Pandas*

Pandas[4] je *open source* balík obsahujúci metódy pre rýchlu a flexibilnú prácu s viacdimenzionálnymi štrukturovanými údajmi. Hlavnými dátovými typmi sú:

- `DataFrame` – dvojrozmerný dátový typ reprezentujúci ekvivalent tabuľky.
- `Series` – jednorozmerný dátový typ reprezentujúci stĺpec `DataFrame`. Obsahuje postupnosť hodnôt, pričom má svoje meno, dátový typ a index.

Oba dátové typy je možné využiť pre reprezentáciu časových radov. Najčastejším spôsobom vytvorenia dátových typov je načítanie údajov zo súboru, napr. pomocou metódy `read_csv`. Import a export údajov je podporovaný vo viacerých formátoch: `CSV`, `TXT` atď. Nad načítanými dátovými štruktúrami je možné realizovať množstvo rôznych základných operácií, ako napr. prístup k hodnotám/atribútom, zisťovanie rozmerov atď. Balík rovnako obsahuje funkcionalitu rôznych typov agregácií a transformácií.

Balík poskytuje aj integrovanú manipuláciu s chýbajúcimi hodnotami v podobe `NaN` hodnôt. Chýbajúce hodnoty pre časový rad je možné doplniť napr. pomocou metód `ffill` a `bfill`, ktoré realizujú doplnenie na základe predošlej, resp. nadchádzajúcej hodnoty.

Balík *Pandas* nájde použitie aj pri vizualizácii. Konkrétne obsahuje modul `plotting`, využitím ktorého je možné nad časovým radom vykresľovať najrôznejšie typy grafov, ako napr. graf autokorelácie pre rôzne oneskorenia.

Balík *Datetime*

Balík *Datetime*[5] obsahuje tri moduly `datetime`, `calendar`, `zoneinfo`. Moduly poskytujú všetky potrebné operácie pre čítanie, formátovanie a manipuláciu s časom. V rámci časových radov teda balík nachádza bohaté uplatnenie.

[3] https://numpy.org/
[4] https://pandas.pydata.org/
[5] https://docs.python.org/3/library/datetime.html

Balík *Matplotlib*

Balík *Matplotlib*[6] je jedným z najznámejších balíkov podporujúcich vizualizáciu údajov. Zahŕňa takmer všetky možné *2D* a *3D* grafy. Pri časových radoch je možné využiť napr. jednostranné, dvojstranné alebo sezónne čiarové vykreslenie hodnôt. Balík rovnako umožňuje spracovávanie a transformáciu objektov v grafe. Súčasťou balíka je aj modul *mathtext*, ktorý umožňuje vytváranie matematických výrazov podobne ako v LaTeX. Vykresľovanie grafov je obvykle realizované pomocou `matplotlib.pyplot` *API*, ktoré umožňuje volaním operácií vykonávať postupné zmeny obrázka.

Balík *Plotly*

Rovnako ako balík *Matplotlib*, aj balík *Plotly*[7] je využívaný pre vizualizáciu údajov. Poskytuje podporu takmer pre všetky *2D* a *3D* grafy. Okrem základne základnej vizualizácie však umožňuje vytvárať aj interaktívne grafy.

Balík *Scipy*

SciPy[8] je balík, ktorý ponúka širokú podporu pre matematické a vedecké výpočty. Konkrétne implementuje moduly zamerané na lineárnu algebru, integráciu, interpoláciu, optimalizáciu, štatistiku, spracovanie signálov a strojové učenie. Funkcionalita balíka je postavená na dátových štruktúrach z balíka *Pandas* a *Numpy* a vizualizácii z balíka *Matplotlib*. Balík je zároveň používaný ako základ pre ďalšie pokročilejšie balíky. Pri časových radoch je balík možné využiť napr. pre doplnenie chýbajúcich hodnôt pomocou interpolácie, prácu s dekomponovanými zložkami radu, zhlukovanie, vyhľadávanie atď.

Balik *Sklearn*

Sklearn[9] je balík, ktorý je široko používaný pre účely štatistického modelovania, strojové učenia a hlboké učenia. Ide o štandard pre dolovanie v jazyku Python. Obsahuje implementáciu rôznych prispôsobiteľných modelov regresie, klasifikácie a zhlukovania. Balík má blízky vzťah s balíkom `SciPy`, nakoľko využíva jeho matematické operácie. Použitie balíka na časové rady je typicky realizované spolu s balíkom `Sktime`, ktorý je jeho špecializovanou nadstavbou.

Ostatné balíky

Okrem vyššie uvedených balíkov existuje veľké množstvo ďalších. Pre pokročilejšiu vizualizáciu je možné použiť napr. balíky *Seaborn*[10], *PyDot*[11], *Bokeh*[12]. Ak je potrebné realizovať úlohy hlbokého učenia, je vhodné využiť napr. balíky *TensorFlow*[13], *Pytorch*[14] alebo *Keras*[15].

[6] https://matplotlib.org/
[7] https://plotly.com/
[8] https://scipy.org/
[9] https://scikit-learn.org/
[10] https://seaborn.pydata.org/
[11] https://pypi.org/project/pydot/
[12] https://bokeh.org/
[13] https://www.tensorflow.org/
[14] https://pytorch.org/
[15] https://keras.io/

4.3 Podpora pre dolovanie z časových radov

V rámci tejto sekcie sú uvedené a popísané balíky, ktoré sa využívajú pre dolovanie z časových radov. Konkrétne ide o balíky *Statsmodels*, *Sktime*, *Tslearn*, *Pyts*, *Pmdarima* atď. Jednotlivé balíky môžu poskytovať rôznu úroveň podpory dolovania, niektoré implementujú funkcionalitu iba pre jednu konkrétnu dolovaciu úlohu, iné môžu zahŕňať rozsiahlu funkcionalitu nielen pre dolovanie.

Balík *Statsmodels*

Statsmodels[16] je balík, ktorý implementuje rozsiahlu funkcionalitu v oblasti štatistickej analýzy. Konkrétne umožňuje vytvárať rôzne modely, vykonávať testy a množstvo ďalších operácií štatistickej analýzy. Výhodou je, že balík dobre spolupracuje s ostatnými balíkmi.

Pre časové rady je vyhradený modul s názvom `tsa`. V module sú implementované rôzne štatistické analýzy a testy, ako napr. `acf` a `pacf` pre odhad (čiastočnej) autokorelácie alebo `adfuller` a `kpps` pre test stacionarity. Modul ďalej umožňuje dekompozíciu radu pomocou volania `seasonal_decompose`. Z časového radu je rovnako možné odstraňovať a filtrovať jednotlivé zložky rozkladu. Pomocou `bkfilter` a `hpfilter` je možné získať cyklickú alebo trendovú zložku. Zrejme najvyužívanejšou časťou modulu sú modely pre predpoveď ďalších hodnôt: `AutoReg`, `ARIMA`, `SARIMAX`, `VAR`, `VARMAX` atď.

Balík *Pmdarima*

Pmdarima[17] je alternatívou modulu `tsa` z balíka *Statsmodels*. Funkcionalita pokrýva automatické vyhľadávanie optimálneho *ARIMA* modelu pre predpoveď, testovanie stacionarity a sezónnosti, diferencovanie a inverzné diferencovanie, rôzne typy transformácií, dekompozíciu časových radov atď. Spôsob používania rozhrania *Pmdarima* je veľmi podobný používaniu rozhrania balíka *Sklearn*.

Balík *Sktime*

Balík *Sktime*[18] je balík založený na balíku *Sklearn* so špecializáciou na časové rady. Balík obsahuje implementáciu modelov pre predpoveď, regresiu, klasifikáciu a zhlukovanie časových radov.

Pre predpoveď je možné využiť napr. modely `NaiveForecaster`, `ARIMA`, `Prophet` atď. Klasifikáciu je možné realizovať napr. s využitím modelov `WEASEL`, `KNeighborsTimeSeriesClassifier`, `ShapeletTransformClassifier`, `TimeSeriesForestClassifier` atď. Pre úlohy regresie obsahuje balík modely `ComposableTimeSeriesForestRegressor`, `TimeSeriesForestRegressor`. Podpora zhlukovania je zatiaľ iba v experimentálnej fáze zastúpená modelmi `TimeSeriesKMeans`, `TimeSeriesKMedoids`. V rámci balíka je tiež obsiahnutý modul `datasets`, ktorý umožňuje načítať dátové sady z archívov *UCR* a *UEA*[19].

Hlavným cieľom vývoja balíka je schopnosť spolupracovať so všeobecnejším a rozsiahlejším balíkom *Sklearn*. Z balíka *Sklearn* sú využívané hlavne tzv. *„Pipelines“*, ktorých funkcionalita spočíva v prepájaní výstupu prvej operácie/modelu s výstupom druhej operácie/modelu.

[16] https://www.statsmodels.org/
[17] https://alkaline-ml.com/pmdarima/
[18] https://www.sktime.org/
[19] https://www.timeseriesclassification.com/

Balík *Tslearn*

Tslearn[20] je balík, ktorý poskytuje nástroje pre analýzu a strojové učenie nad časovými radmi. Tento balík závisí od balíkov *Sklearn*, *NumPy* a *SciPy*. V rámci jeho implementovanej funkcionality sú zahrnuté modely pre predpoveď, klasifikáciu, podobnostné vyhľadávanie, zhlukovanie atď. Okrem uvedenej funkcionality balík podporuje aj generovanie časových radov a ich následné predspracovanie. V rámci balíka je tiež obsiahnutý modul `datasets`, ktorý umožňuje načítať dátové sady z archívov *UCR* a *UEA*.

Predpoveď je možné realizovať s využitím modelu `TimeSeriesSVR`, `TimeSeriesMLPRegressor`, `KNeighborsTimeSeriesRegressor`. Pre úlohu klasifikácie sú dostupné modely `TimeSeriesSVC`, `TimeSeriesMLPClassifier`, `KNeighborsTimeSeriesClassifier` atď. Vyhľadávanie podobností/odľahlostí je zastúpené modelmi `KNeighborsTimeSeries`. Zhlukovanie je možné vykonať s modelmi `TimeSeriesKMeans`, `KShape`, `KernelKMeans`.

Balík *Pyts*

Pyts[21] je balík určený na klasifikáciu časových radov. Zameriava sa na to, aby bola klasifikácia časových radov ľahšie realizovateľná. Poskytuje implementáciu niekoľkých klasifikačných algoritmov z rôznych kategórií. Jednotlivé klasifikačné algoritmy okrem iného zapuzdrujú aj potrebné predspracovanie časového radu. V balíku je možné nájsť modely, ako napr. `KNeighborsClassifier`, `BOSSVS`, `LearningShapelets`, `SAXVSM`, `TimeSeriesForest` atď. Rovnako ako v predošlých balíkoch, aj v rámci tohto balíka je obsiahnutý modul `datasets`, ktorý umožňuje načítať dátové sady z archívov *UCR* a *UEA*.

Ostatné balíky

Pre časové rady existuje ešte veľké množstvo ďalších balíkov. Manipuláciu, analýzu a predpoveď časových radov je možné vykonať s balíkmi *Darts*[22], *AtsPy*[23], *Kats*[24], *Prophet*[25], *Orbit*[26], *Pastas*[27], *Pyflux*[28], *Autots*[29] alebo *Flow-forecast*[30]. V prípade potreby implementácie dolovacích úloh nad časovými radmi je možné využiť balíky *Seglearn*[31], *Tsfresh*[32] alebo *PyCaret*[33]. Generovanie nových syntetizovaných časových radov je možné realizovať s balíkom *TimeSynth*[34].

[20] https://tslearn.readthedocs.io/
[21] https://pyts.readthedocs.io/
[22] https://unit8co.github.io/darts/
[23] https://github.com/firmai/atspy/
[24] https://facebookresearch.github.io/Kats/
[25] https://facebook.github.io/prophet/
[26] https://orbit-ml.readthedocs.io/
[27] https://pastas.readthedocs.io/
[28] https://pyflux.readthedocs.io/
[29] https://winedarksea.github.io/AutoTS/
[30] https://flow-forecast.readthedocs.io/
[31] https://dmbee.github.io/seglearn/
[32] https://tsfresh.readthedocs.io/
[33] https://pycaret.org/
[34] https://github.com/TimeSynth/TimeSynth/

Kapitola 5

Aplikácia

Táto kapitola je zameraná na popis návrhu, implementácie, funkčnosti a použitia aplikácie realizujúcej experimenty dolovania z časových radov.

V prvej podkapitole 5.1 je uvedený popis požiadaviek na aplikáciu a jej počiatočný návrh. V rámci požiadaviek sú uvedené funkčné požiadavky, technické požiadavky a požiadavky na používateľské rozhranie. Samotný návrh ďalej obsahuje popis a ukážku prvotnej funkčnosti a používateľského rozhrania aplikácie. Ďalšia podkapitola 5.2 je vyčlenená pre popis spôsobu implementácie aplikácie. V podkapitole sú rovnako uvedené konkrétne technológie, nástroje, programovacie jazyky, balíky a iný softvér, ktorý je pri realizácii použitý. Posledná podkapitola 5.3 obsahuje popis a ukážku použitia vytvorenej aplikácie. Zároveň sú v podkapitole uvedené odlišnosti finálnej aplikácie od pôvodného návrhu.

V jednotlivých podkapitolách sa popis odkazuje na súbory z adresárovej štruktúry projektu. Popis a ukážka adresárovej štruktúry je dostupná v prílohe B. V prílohe C je uvedený postup inštalácie a spustenia aplikácie .

5.1 Požiadavky na aplikáciu a návrh

Cieľom a výstupom aplikácie je demonštrácia a porovnanie jednotlivých metód dolovania. Konkrétne sa aplikácia špecializuje na rôzne metódy pre úlohy predpovede/regresie, klasifikácie, zhlukovania a vyhľadávania podobností/odľahlostí. Z pohľadu funkčnosti sú na aplikáciu pokladané nasledovné požiadavky:

- import údajov,
- analýza a vizualizácia údajov,
- demonštrácia rôznych metód dolovania s rôznym nastavením,
- vizualizácia výsledkov dolovania,
- porovnanie metód dolovania (presnosť výsledkov, spotreba času/pamäte, atď.),
- export výsledkov.

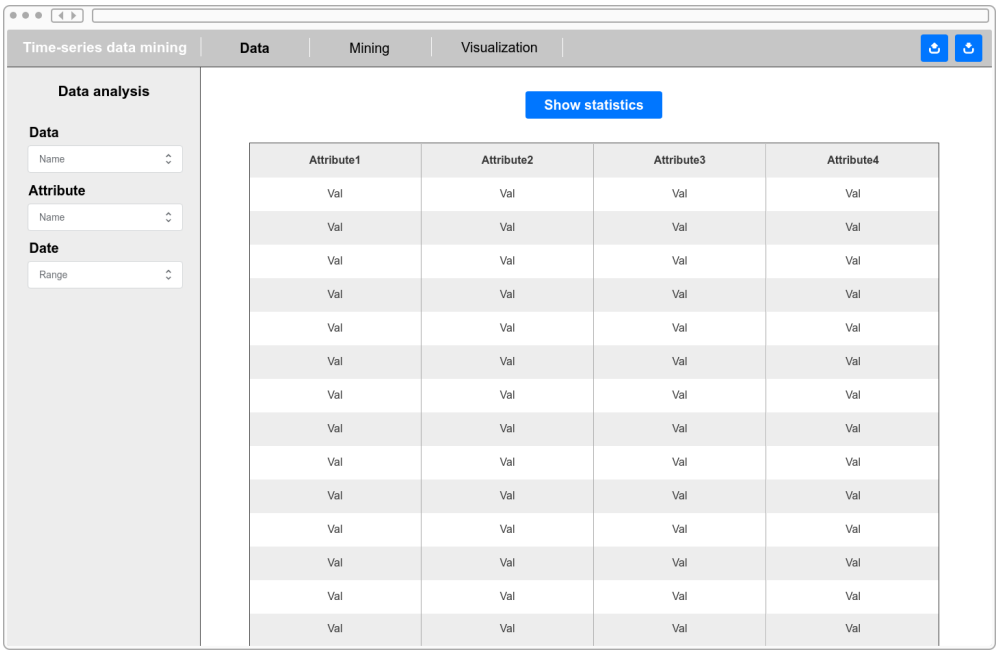
Technickou požiadavkou je, aby aplikácia fungovala, bežala a bola prístupná cez web, t. j., aby išlo o webovú aplikáciu. Všeobecné požiadavky na používateľské rozhranie sú jednoduchosť, intuitívnosť a rýchlosť.

Aplikácia umožňuje prepínanie rôznych pohľadov. Konkrétne, pohľad analýzy zdrojových údajov, pohľad dolovania z údajov a pohľad vizualizácie údajov a výsledkov dolovania.

Pohľady je možné prepínať pomocou odkazov v hornom **navigačnom paneli**, viď obrázok 5.1. Horný navigačný panel je dostupný v každom pohľade, pričom okrem prepínania pohľadov umožňuje aj import a export údajov. V rámci importu a exportu je podporovaných viacero formátov, ako napr. *CSV*, *JSON*.

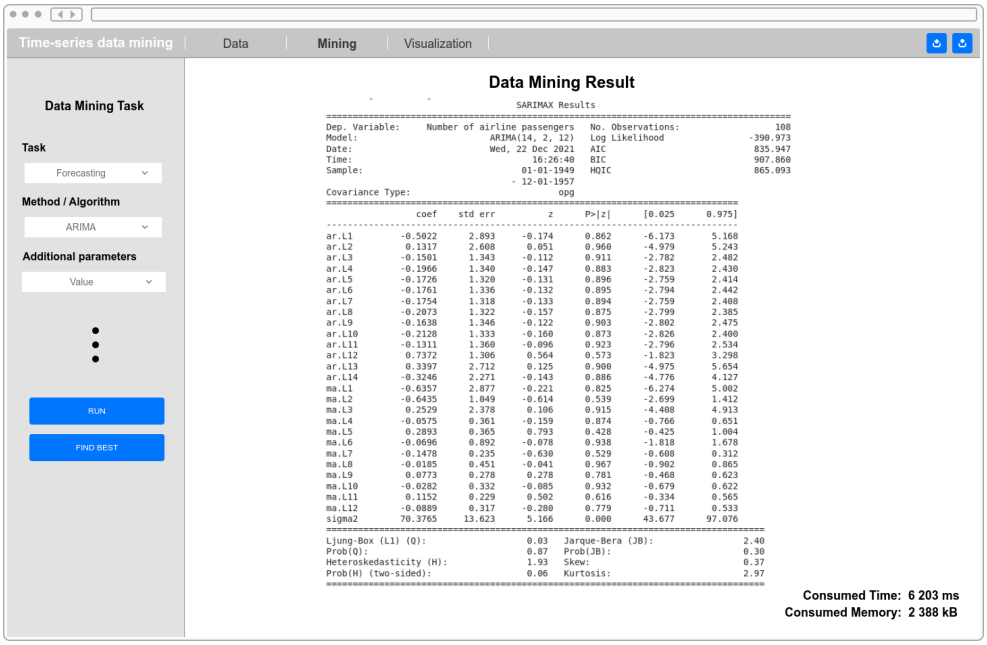
Fixným panelom každého pohľadu je aj ľavý bočný **ovládací panel**. Jeho obsah je však pri každom pohľade odlišný, a to v závislosti na tom, čo je ovládané.

Pohľad **analýzy údajov** umožňuje vizualizáciu načítaných údajov formou tabuľky a podrobnejších štatistík. Obsahom ľavého bočného ovládacieho panela je ovládanie voľby vizualizovaných údajov. Navrhnutý pohľad je vizualizovaný na obrázku 5.1.



Obr. 5.1: Pohľad analýzy zdrojových údajov

Pohľad dolovania z údajov umožňuje výber a realizáciu konkrétnej metódy pre jednotlivé úlohy dolovania. Metódy je možné spúšťať s rôznymi parametrami nastavovanými v ľavom bočnom ovládacom paneli. Po dokončení behu metódy je zobrazený súhrn výsledkov dolovania obsahujúci informácie o výsledku behu, dĺžke behu, spotrebovanej pamäte atď. Navrhnutý pohľad je vizualizovaný na obrázku 5.2.



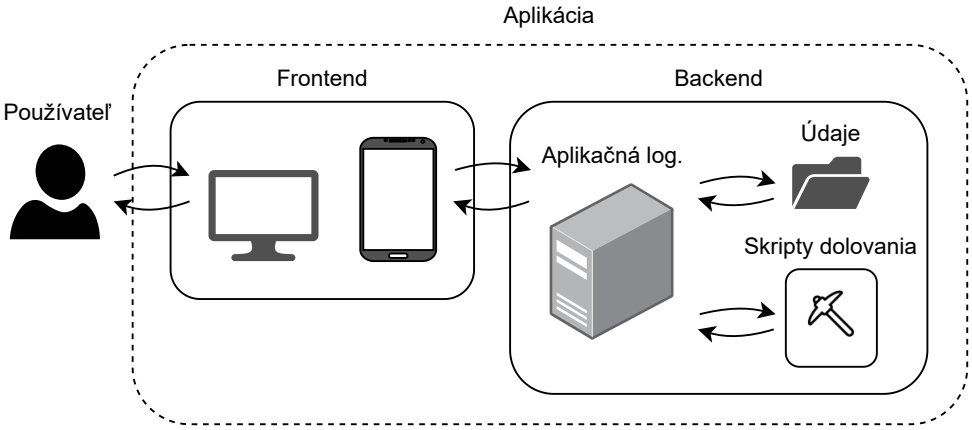
Obr. 5.2: Pohľad dolovania z údajov

Pohľad vizualizácie údajov a výsledkov dolovania zobrazuje údaje a výsledky vo forme rôznych typov grafov. Ľavým bočným ovládacím panelom je možné nastaviť typ grafu a ďalšie rôzne parametre. Navrhnutý pohľad je vizualizovaný na obrázku 5.3.

Obr. 5.3: Pohľad vizualizácie údajov a výsledkov dolovania

5.2 Implementácia aplikácie

Účelom tejto podkapitoly je popísať technológie a spôsob fungovania aplikácie. Aplikácia je implementovaná ako webová aplikácia. Pre jej realizáciu je možné vyberať z veľkého množstva rôznych technológií. Aplikácia je podľa štandardnej architektúry *Klient-server* rozdelená do dvoch oddelených častí *frontend* a *backend*. Jednotlivé časti spolu vzájomne komunikujú, *frontend* je spustený na strane klienta/používateľa, *backend*, naopak, na strane servera. *Backend* aplikácie je navyše rozdelený do ďalších dvoch menších vzájomne komunikujúcich častí. Prvá zabezpečuje obsluhu volaní, logiku a smerovanie údajov. Druhá zabezpečuje dolovanie z časových radov. Fungovanie vytvorenej aplikácie je možné vizualizovať schémou 5.4.



Obr. 5.4: Schéma architektúry implementovanej aplikácie

Podkapitola je rozdelená do niekoľkých sekcií. V prvej sekcii 5.2.1 je spolu s využitými technológiami popísaný spôsob fungovania *frontendu* aplikácie. Nasledujú sekcie 5.2.2 a 5.2.3 s popisom fungovania a technológií *backendu* aplikácie. V jednotlivých sekciách je popis funkcionality odkazovaný na súbory v projekte. Ukážka adresárovej štruktúry so súbormi projektu sa nachádza v prílohe B. Rovnako je v prílohách uvedený aj postup inštalácie a spustenia aplikácie C.

5.2.1 *Frontend*

Frontend aplikácie je implementovaný s využitím *Javascript frameworku Vue.js*[1] verzia *3* a množstvom ďalších pomocných technológií, akými sú *HTML*[2], *CSS*[3], *Javascript*[4], *Bootstrap*[5] atď. *Framework Vue.js* pri svojej činnosti využíva ďalšie balíky, viď popis nižšie. Správu použitých balíkov má na starosti nástroj *Npm*[6]. *Npm* spravuje okrem balíkov *Vue.js* aj balíky pre celý *Javascript*.

Frontend aplikácie je logicky rozdelený do niekoľkých súborov, pričom jednotlivé súbory reprezentujú znovu použiteľné komponenty. Komponent reprezentujúci základnú štruktúru

[1] https://vuejs.org/
[2] https://developer.mozilla.org/en-US/docs/Web/HTML/
[3] https://developer.mozilla.org/en-US/docs/Web/CSS/
[4] https://www.javascript.com/
[5] https://getbootstrap.com/
[6] https://www.npmjs.com/

stránky je implementovaný v súbore `Layout.vue`. Komponenty reprezentujúce stránky je možné nájsť v adresári `pages`. Implementácia komponent využívaných v jednotlivých stránkach je uvedená v adresári `components`.

Aplikácia je implementovaná ako **jednostránková aplikácia** (*Single Page Application*). Po načítaní prvotnej stránky je teda smerovanie zabezpečené v časti *frontendu*. Smerovanie je vykonané s využitím balíka *Vue-router*[7] a jeho implementáciu je možné nájsť v súbore `router.js`.

Pre správne fungovanie *frontendu* aplikácie je potrebné zabezpečiť efektívnu **komunikáciu s *backendom***. Komunikácia je realizovaná asynchrónne pomocou *Javascript* rozhrania *Fetch API*[8]. Odosielané údaje sú zabalené v objekte `Request`, prijímané údaje v objekte `Response`. Formát prenášaných údajov je vo formáte *JSON*. Výnimkou je prenos zdrojových údajov pre dolovanie, ktoré sú prenášané vo formáte *CSV*. Ich spracovanie do internej reprezentácie v podobe poľa je realizované s použitím balíka *Vue-papa-parse*[9].

Vo *frameworku Vue.js* je možné údaje komponent uchovávať iba lokálne. Zdieľanie údajov medzi komponentmi je realizované odovzdávaním cez vstupné parametre. To je však v množstve prípadoch nevhodné a nepostačujúce. Práve preto je využitý balík *Vuex*[10], ktorý zabezpečuje **globálne ukladanie údajov** komponent. Manipulácia s údajmi je vykonaná pomocou rôznych typov funkcií. Implementáciu globálneho ukladania údajov je možné nájsť v súbore `global.js`.

V aplikácii je ďalej potrebné používať **formuláre**. Pre vstupné pole dátumu a času je využitý komponent `Datepicker` z balíka *Vue-datepicker*[11]. Validáciu formulárov umožňuje balík *Vuelidate*[12]. Údaje pre vstupné polia sú načítané zo súborov v podadresári `data` adresára `resources`.

Frontend aplikácie rovnako umožňuje **exportovať výsledky** dolovacích úloh. Načítaný výsledok vo forme *Javascript* objektu je možné exportovať ako *JSON* objekt pomocou balíka *Downloadjs*[13].

V aplikácii je vyžadovaná vizualizácia údajov v podobe **grafov**. Túto funkcionalitu zabezpečuje balík *Vue-chart-3*[14]. Pre vizualizáciu časových radov je využívaný hlavne čiarový graf, ktorý je reprezentovaný komponentom `LineChart`.

Dizajn aplikácie je vylepšený pridaním rôznych **ikon**. Použité ikony sú načítané z balíka *Fontawesome*[15].

5.2.2 *Backend* – aplikačná logika

Aplikačná logika *backendu* aplikácie je vytvorená pomocou *PHP frameworku Laravel*[16] vo verzii *8*. *Framework* pri fungovaní používa množstvo rôznych balíkov, viď popis nižšie. Obsiahnuté balíky sú spravované nástrojom *Composer*[17]. Ide o správcu balíkov nielen pre *Laravel*, ale aj pre celé *PHP*.

[7] https://router.vuejs.org/
[8] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/
[9] https://www.npmjs.com/package/vue-papa-parse/
[10] https://vuex.vuejs.org/
[11] https://vue3datepicker.com/
[12] https://vuelidate.js.org/
[13] https://www.npmjs.com/package/downloadjs/
[14] https://vue-chart-3.netlify.app/
[15] https://fontawesome.com/
[16] https://laravel.com/
[17] https://getcomposer.org/

V predošlej sekcii je spomenuté, že aplikácia je jednostránková a smerovanie po načítaní prvotnej stránky zabezpečuje *frontend*. Prvotná stránka však môže byť akákoľvek. V časti *backendu* je teda potrebné zabezpečiť správne načítanie prvotnej stránky. Smerovanie pre prvotné stránky je pomocou tzv. *routes* implementované v súbore `web.php`. Smerovanie pre asynchrónne volania žiadajúce o prístup k súborom v databáze je možné nájsť v súbore `api.php`.

V rámci *backendu* aplikácie je ukladaných niekoľko súborov, s ktorými aplikácia počas činnosti pracuje. Konkrétne ide o súbory reprezentujúce:

- **zdrojové údaje pre dolovanie** – uložené vo formáte *JSON* v podadresároch `data` a `actualData` adresára `storage`,
- **algoritmy úloh** – napísané v programovacom jazyku *Python* a uložené v podadresári `scripts` adresára `storage`,
- **výsledky úloh** – uložené vo formáte *JSON* v podadresári `results` adresára `storage`.

Manipuláciu s týmito súbormi implementujú triedy `FileController` a `PythonController`, ktoré sa nachádzajú v rovnako pomenovaných súboroch v adresári `Controllers`. Na metódy týchto tried sú smerované volania z *frontendu* aplikácie. V prvom z týchto súborov sa pomocou rozhrania objektu `Storage` pristupuje k údajom zdrojových dát a výsledkom dolovacích úloh. V druhom sa pomocou rozhrania objektu `Process` z balíka *Symfony*[18] spúšťajú skripty dolovania z časových radov. Rozhranie tohoto objektu teda zabezpečuje komunikáciu s druhou časťou *backendu* pre dolovanie. Spúšťanie skriptov pre dolovanie je podmienené. Ak už bol výsledok niekedy v minulosti vypočítaný, tak sa len načíta z príslušného súboru a výpočet sa nerealizuje.

5.2.3 *Backend* – dolovanie z časových radov

Samotné dolovanie z časových radov v *backende* aplikácie zabezpečuje programovací jazyk *Python* a množstvo pridružených balíkov pre dolovanie popísaných v predošlej kapitole 4. Správa všetkých použitých balíkov je vykonaná s využitím nástroja *Pip*[19].

Jednotlivé dolovacie úlohy sú reprezentované oddelenými skriptami. Skripty sú na základe používateľom definovaných vstupov volané z aplikačnej časti *frontendu*. Skripty očakávajú na vstupe údaje vo formáte *JSON* s jasne definovanou štruktúrou. V rámci dolovania následne prebieha výpočet využívajúci balíky pre dolovanie, popísané v kapitole 4. Po procese dolovania sú vypočítané rôzne metriky, ktoré vyhodnocujú použitý model a výstup dolovania. Konkrétnymi príkladmi sú presnosť modelu alebo spotrebovaný čas a pamäť. Posledným krokom je spätný návrat údajov. Ten je zabezpečený volaním funkcie `json_dump` z balíka *Simplejson*[20], ktorá prevedie *Python* objekt na reťazec reprezentujúci *JSON* objekt. Reťazec je následne vypísaný na štandardný výstup, ktorý odchytáva časť backendu s aplikačnou logikou.

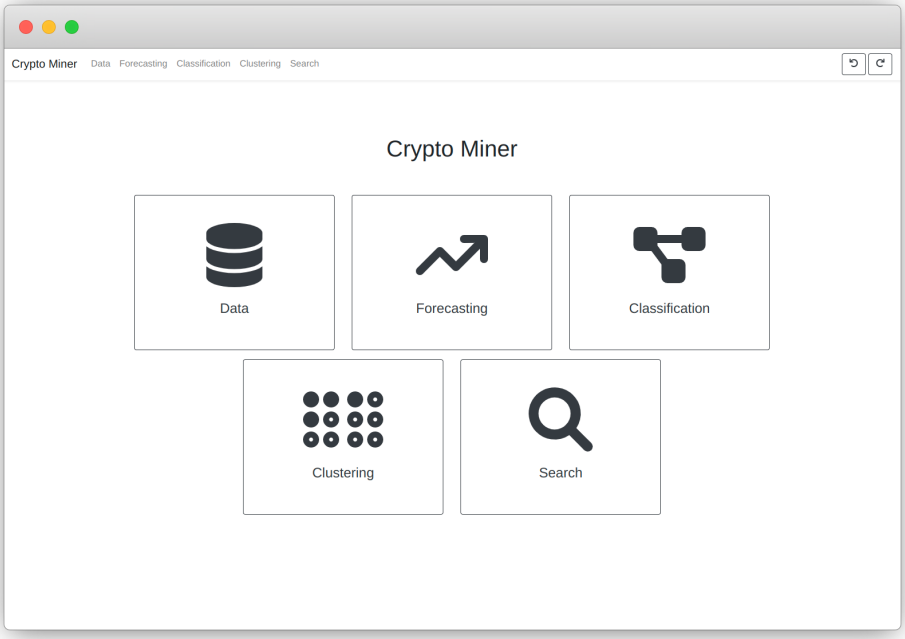
[18] https://symfony.com/
[19] https://pypi.org/project/pip/
[20] https://pypi.org/project/simplejson/

5.3 Použitie a funkčnosť aplikácie

V tejto podkapitole je uvedený popis a ukážka použitia a funkčnosti finálnej aplikácie. Vytvorená aplikácia vo vysokej miere vychádza z vytvoreného návrhu popísaného v podkapitole 5.1, ale zároveň obsahuje aj niekoľko zmien. V rámci tejto podkapitoly budú všetky zmeny popísané.

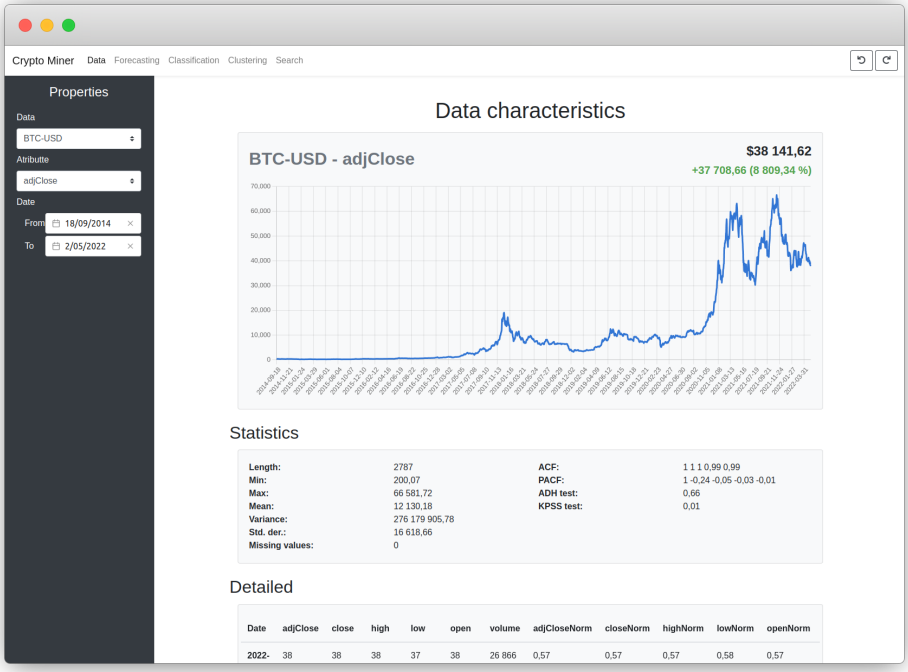
Základné rozloženie aplikácie pozostáva z horného navigačného panela, bočného ovládacieho panela a hlavného okna. Štruktúra aplikácie je teda zachovaná z prvotného návrhu. Zmenou v **hornom navigačnom paneli** oproti vytvorenému návrhu je náhrada importu údajov za automatickú aktualizáciu údajov dátovej sady pre aktuálny deň. Údaje je rovnako možné kedykoľvek vrátiť do pôvodného stavu. Zmena je vykonaná z dôvodu špecializácie aplikácie na zvolenú dátovú sadu. Podrobný popis zvolenej dátovej sady sa nachádza v podkapitole 6.1. Ďalšou zmenou v navigačnom paneli je presun tlačidla s exportom výsledkov ku konkrétnemu výsledku. Navigácia pohľadov je v paneli zachovaná. **Bočný ovládací panel** umožňuje nastavovať vstupy pre jednotlivé úlohy. Jeho obsah sa odlišuje na základe konkrétnych pohľadov, viď popis nižšie.

Funkcionalita aplikácie je rozdelená do niekoľkých pohľadov. Prvým implementovaným pohľadom je **úvodný pohľad**, ktorý sa skladá z piatich tlačidiel odkazujúcich na ostatné pohľady. Vytvorený pohľad je možné vidieť na obrázku 5.5.



Obr. 5.5: Úvodný pohľad aplikácie

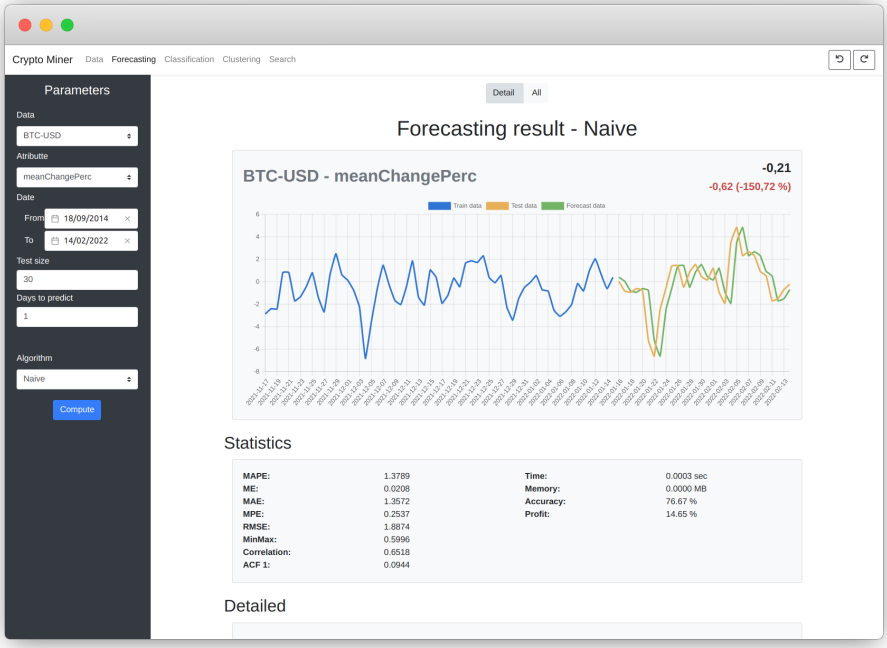
Pohľad **analýzy zdrojových údajov** je zachovaný z prvotného návrhu. Jeho ovládací panel umožňuje zvoliť zdrojové údaje, konkrétny atribút a časový rozsah analýzy. V rámci hlavného okna sa následne nachádza výsledok analýzy v podobe vizualizácie, štatistík a detailu zvolených údajov. Vizualizácia údajov je realizovaná čiarovým grafom. Detail údajov je reprezentovaný formou tabuľky. Vytvorený pohľad je možné vidieť na obrázku 5.6.



Obr. 5.6: Pohľad analýzy zdrojových údajov

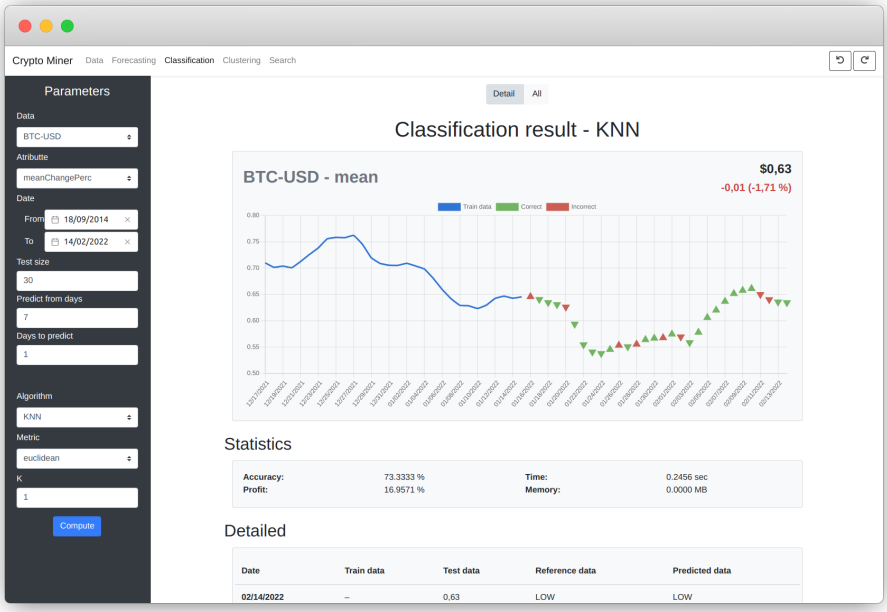
V **pohľade dolovania z údajov** v porovnaní s návrhom nastala zmena. Pohľad dolovania z údajov je rozdelený na štyri rôzne pohľady reprezentujúce konkrétne úlohy: pohľad predpovede/regresie, pohľad klasifikácie, pohľad zhlukovania a pohľad vyhľadávania. V jednotlivých pohľadoch ovládací panel umožňuje ako vstup úlohy zvoliť zdrojové dáta, atribút, časový rozsah, algoritmus a ďalšie dodatočné parametre. Algoritmy, z ktorých je možné pri jednotlivých úlohách vyberať, sú popísané a demonštrované v kapitole 6. Hlavné okno obsahuje vizualizáciu výsledkov úloh v podobe čiarového grafu, štatistiky v podobe rôznych metrík a detail výsledných údajov vo forme tabuľky.

Ukážku pohľadu **predpovede/regresie** údajov je možné vidieť na obrázku 5.7.



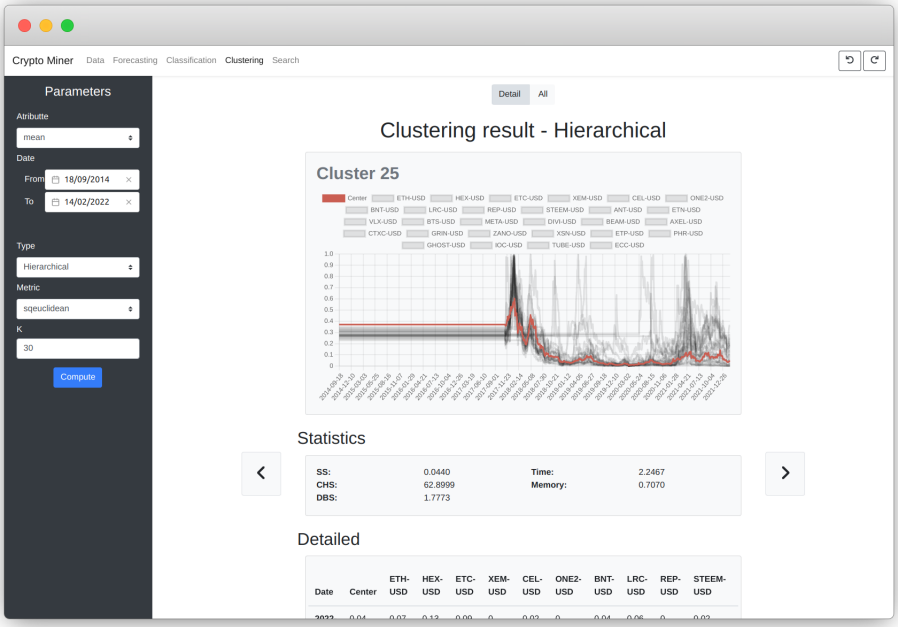
Obr. 5.7: Pohľad predpovede údajov

Pohľad **klasifikácie** údajov je zobrazený na obrázku 5.8.



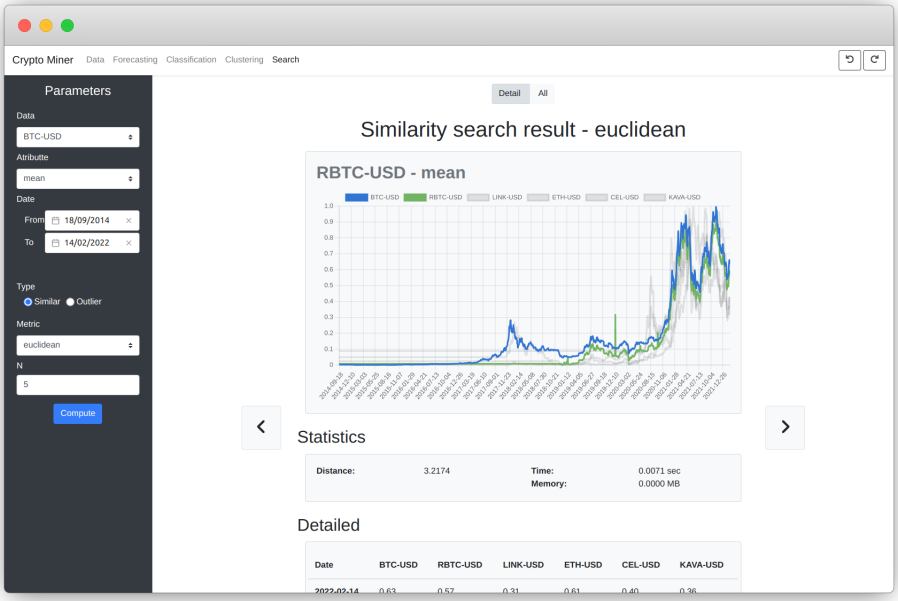
Obr. 5.8: Pohľad klasifikácie údajov

V ovládacom paneli pohľadu **zhlukovania** údajov je odstránené vstupné pole s výberom zdrojových údajov, nakoľko je úloha realizovaná nad celou dátovou sadou. Pohľad je zobrazený na obrázku 5.9.



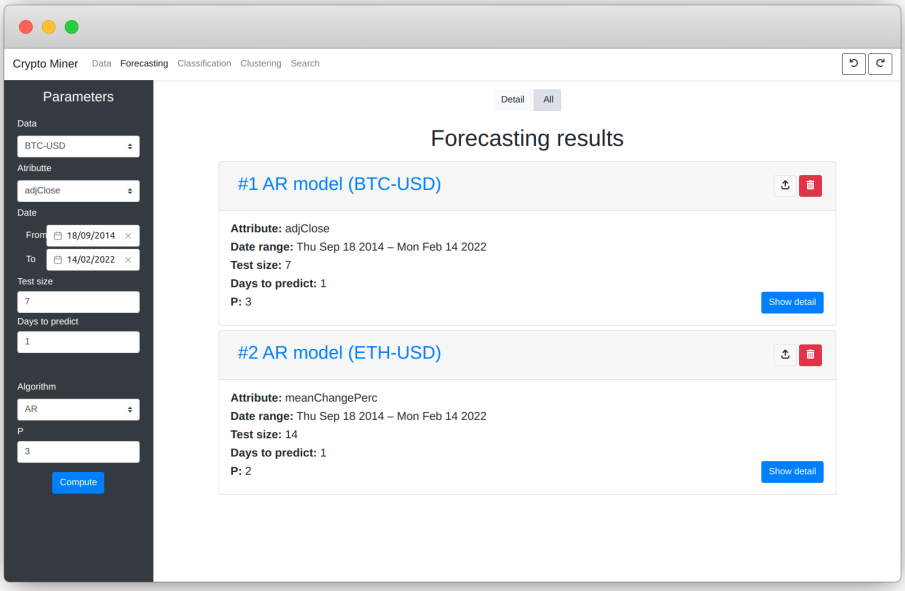
Obr. 5.9: Pohľad zhlukovanie údajov

Posledný pohľad pre úlohu **vyhľadávania** údajov je zobrazený na obrázku 5.10. Ovládací panel obsahuje možnosť výberu vyhľadávania podobných alebo odľahlých časových radov a množstvo rôznych metrík.



Obr. 5.10: Pohľad vyhľadávania údajov

V každom z pohľadov dolovania je možné zobraziť predchádzajúce **výsledky výpočtov** pre danú úlohu. Predchádzajúci výsledok je načítaný v prípade, že sú na vstupe novej operácie dolovania zvolené zhodné údaje. Načítaním historického výsledku sa zabraňuje opakovaniu výpočtov a zvyšuje sa tak rýchlosť aplikácie. Aplikácia nad výsledkami navyše umožňuje operácie exportu a zmazania. Pohľad s prehľadom predchádzajúcich výsledkov výpočtu je uvedený na obrázku 5.11.



Obr. 5.11: Pohľad prehľadu výsledkov

Pohľad **vizualizácie údajov** z prvotného návrhu je vo finálnej aplikácii odstránený. Vizualizácie sú presunuté do pohľadov popísaných vyššie.

Kapitola 6

Experimenty

Táto kapitola je zameraná na prezentáciu vytvorených experimentov. Experimenty sú implementované v programovacom jazyku *Python*. Ich cieľom je demonštrovať a porovnať jednotlivé metódy pre konkrétne úlohy dolovania z časových radov. Všeobecná teória k prezentovaným úlohám je uvedená v podkapitole 2.4. Experimenty nadväzujú a vychádzajú z kapitoly 3, kde je detailne popísaný princíp funkčnosti demonštrovaných metód dolovania z časových radov.

Kapitola je rozdelená do niekoľkých podkapitol. V prvej podkapitole 6.1 je predstavená dátová sada kryptomien, ktorá slúži ako zdroj údajov pre experimenty. Jej podrobnejšia analýza a predspracovanie je popísané v podkapitole 6.2. Jadrom kapitoly sú tri experimenty:

1. **predpoveď/regresia časových radov** – predpovedaný vývoj ceny kryptomien,
2. **klasifikácia časových radov** – klasifikovaný zisk/pokles ceny kryptomien,
3. **vyhľadávanie a zhlukovanie časových radov** – vyhľadávané a zhlukované najviac podobné/odľahlé kryptomeny.

V podkapitole 6.3 je zdokumentovaný experiment predpovede/regresie, v podkapitole 6.4 experiment klasifikácie a v podkapitole 6.5 experiment zhlukovania/vyhľadávania. Každý experiment má jasne definovanú štruktúru, ktorá sa skladá z predstavenia experimentu, uvedenia použitých zdrojových údajov a ich prípadných úprav, realizácie experimentu a zhodnotenia výsledkov experimentu.

6.1 Dátová sada kryptomien

Zdrojovými údajmi pre realizáciu experimentov je **dátová sada kryptomien**. Kryptomeny sú digitálnym aktívom, ktorý slúži ako prostriedok výmeny medzi jednotlivcami a skupinami. Ide o platobný systém, ktorý je decentralizovaný. Ukladanie a transakcie teda prebiehajú oddelene v samostatných jednotkách, pričom zabezpečenie je realizované pomocou komplikovanej kryptografie, z čoho vychádza aj samotný názov aktíva.

Kryptomeny nepatria žiadnemu národu a sú navrhnuté tak, aby neboli pod kontrolou žiadnej vlády alebo centrálnej banky. Táto vlastnosť ich odlišuje od tradičných mien ako dolár alebo euro. Spoločnou vlastnosťou kryptomien a tradičných mien je možnosť vzájomných prevodov na trhu. Trh kryptomien je veľmi nestabilný, nevyspytateľný a vývoj cien určuje celý rad faktorov. Vytvára teda veľký potenciál pre automatizovanú predpoveď a

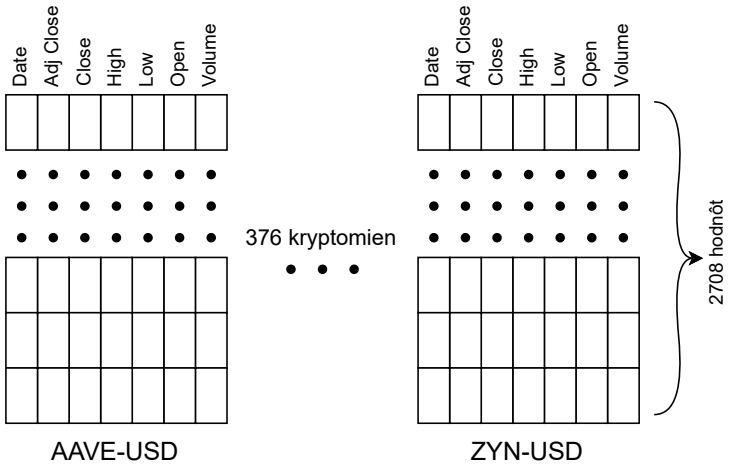
realizáciu prevodov. Predpoveď a prevody je možné realizovať podľa rôznych stratégií a s využitím rôznych modelov. V tejto práci bude časť z nich demonštrovaná a porovnaná.

Samotné údaje s cenami kryptomien sú stiahnuté z online repozitára na webovom portáli *Kaggle*[1]. Adresár okrem údajov s cenami kryptomien obsahuje aj priložené skripty s možnosťou stiahnutia aktualizovaných cien. Skripty sú napísané v programovom jazyku *Python*, pričom stiahnutie je realizované pomocou balíka *Yfinance*[2]. Ide o *open-source* balík, sťahujúci údaje z verejne dostupného rozhrania *Yahoo Finance*[3].

V dátovej sade je celkovo *376* odlišných kryptomien s rôznou frekvenciou zberu: denná, hodinová, minútová. V rámci tejto práce je pozornosť zameraná na dennú frekvenciu. Každá kryptomena je reprezentovaná siedmimi atribútmi:

- `date` – dátum/deň zaznamenania ceny,
- `adj close` – upravená uzatváracia cena dňa (zahrnuté všetky ovplyvňujúce faktory),
- `close` – uzatváracia cena dňa,
- `open` – otváracia cena dňa,
- `high` – maximálna cena dňa,
- `low` – minimálna cena dňa,
- `volume` – množstvo kryptomeny v obehu v daný deň.

Údaje sú zozbierané v časovom rozsahu od *17.9.2014* do *14.2.2022*, čo predstavuje dĺžku *2708* hodnôt. V experimentoch je dátová sada uložená ako slovník dátových rámcov z balíka *Pandas*. K údajom jednotlivých kryptomien je teda možné pristupovať pomocou ich názvu. Štruktúra dátovej sady je vizualizovaná na obrázku 6.1.



Obr. 6.1: Štruktúra dátovej sady kryptomien

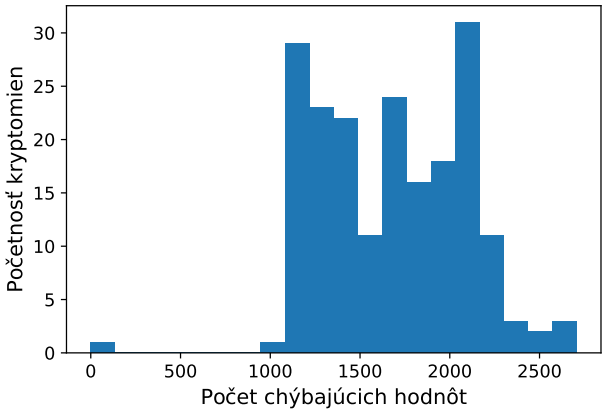
[1] https://www.kaggle.com/datasets/benjaminpo/crypto-historical-price/
[2] https://pypi.org/project/yfinance/
[3] https://finance.yahoo.com/

6.2 Predspracovanie a analýza údajov

Dátová sada nie je dokonalá, obsahuje hneď niekoľko nedostatkov. Odstránenie týchto nedostatkov a ďalšie predspracovanie pre experimenty je popísané v tejto podkapitole. Konkrétne sú analyzované a spracované chýbajúce hodnoty, odľahlé hodnoty. Následne je realizovaná normalizácia, extrakcia nových atribútov a ich korelačná analýza. V závere je vykonaná analýza vhodnosti jednotlivých atribútov pre experimenty a ich následný výber.

6.2.1 Analýza chýbajúcich hodnôt

Takmer každá kryptomena v dátovej sade obsahuje **chýbajúce hodnoty**. Konkrétny počet chýbajúcich hodnôt pre jednotlivé kryptomeny je odlišný. Histogram počtu chýbajúcich hodnôt je uvedený na obrázku 6.2.

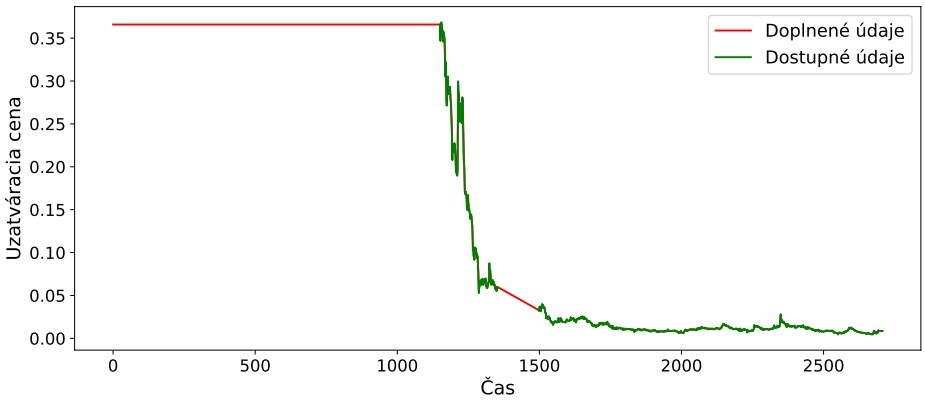


Obr. 6.2: Počet chýbajúcich hodnôt

Chýbajúce hodnoty sa vo väčšine prípadov nachádzajú na začiatku časových radov. Vyplýva to z faktu, že hodnoty všetkých časových radov sú zbierané od rovnakého dátumu, ale ich vznik je v každom prípade odlišný. Chýbajúce hodnoty sa v niektorých prípadoch nachádzajú aj medzi prítomnými hodnotami a na konci časových radov.

Chýbajúce hodnoty na začiatku časových radov sú spracované metódou spätného doplnenia, t. j. chýbajúce hodnoty sú doplnené prvou skutočnou prítomnou hodnotou. Doplnenie je realizované metódou `bfill` triedy `DataFrame` z balíka `pandas`. Metóda je zvolená z dôvodu, že najlepším možným spôsobom popisuje skutočnosť doposiaľ neexistujúcej kryptomeny.

Chýbajúce hodnoty uprostred časových radov sú doplnené metódou lineárnej interpolácie, ktorá je v jazyku *Python* implementovaná metódou `interpolate` triedy `DataFrame`. Metóda je zvolená z dôvodu návratu najpresnejších výsledkov. Výsledok metód spätného doplnenia a interpolácie pre kryptomenu *Lykke* je vizualizovaný na obrázku 6.3.



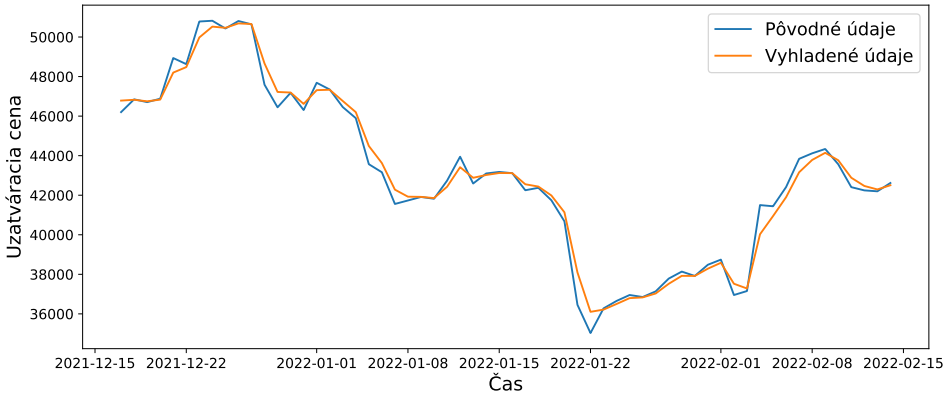
Obr. 6.3: Doplnený časový rad (kryptomena *Lykke*)

Kryptomeny, ktoré obsahujú chýbajúce hodnoty na konci časového radu, je možné považovať za neaktuálne/zaniknuté. Sú teda z dátovej sady odstránené. Počet odstránených kryptomien je *18* a veľkosť dátovej sady je tak znížená na *358* kryptomien.

6.2.2 Analýza odľahlých hodnôt

Kryptomeny obsahujú v niekoľkých prípadoch v časovom rade **odľahlé hodnoty**. Už v sekcii 3.2.7 bolo uvedené, že odľahlé hodnoty môžu spôsobiť zníženie presnosti modelov. Je teda žiadúce ich odstrániť.

Pre odstránenie chýbajúcich hodnôt je zvolená metóda exponenciálneho vyhladenia s parametrom miery vyhladenia $\alpha = 0{,}65$. Funkcionalitu implementuje metóda `ewm` triedy `DataFrame`. Vizualizácia vyhladenia atribútu `Close` kryptomeny *Bitcoin* je zobrazená na obrázku 6.4.



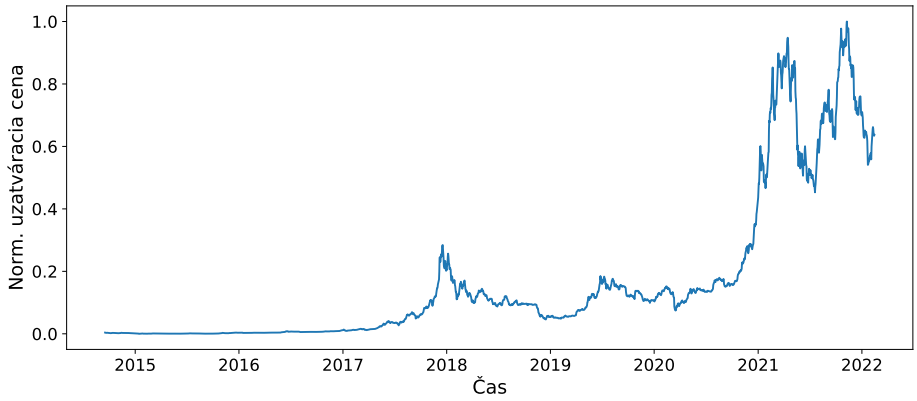
Obr. 6.4: Vyhladený časový rad (kryptomena *Bitcoin*)

6.2.3 Normalizácia hodnôt

Ďalšou úpravou, ktorá je vykonaná nad dátovou sadou, je **normalizácia hodnôt**. Normalizácia hodnôt je vykonaná z dôvodu, že ceny kryptomien v dátovej sade sa výrazne

líšia. Výška ceny by však pri učení modelov nemala mať žiaden vplyv na ich výsledky. Je teda žiadúce, aby ceny kryptomien boli v rovnakom rozsahu.

Hodnoty sú normalizované funkcionalitou triedy `MinMaxScaler` z balíka *Sklearn*, ktorá pôvodným hodnotám priraďuje hodnoty z intervalu *0* až *1*. Najmenšej hodnote v časovom rade je priradená hodnota *0*, najväčšej hodnote je priradená hodnota *1*. Výsledok normalizácie pre kryptomenu *Bitcoin* je vizualizovaný na obrázku 6.5.



Obr. 6.5: Normalizovaný časový rad (kryptomena *Bitcoin*)

6.2.4 Extrakcia nových atribútov

V dátovej sade sú po predošlých úpravách okrem pôvodných doplnených atribútov aj vyhladené a normalizované varianty. Pre dolovanie z kryptomien však ide do vysokej miery o nepostačujúce príznaky, s ktorými by vytvorené modely dosahovali iba slabé výsledky. Z tohto dôvodu je vytvorených niekoľko ďalších **nových atribútov** v podobe nasledovných indikátorov:

- `avg` – denná priemerná cena,
- `avgc` – denný zisk/pokles priemernej ceny,
- `avgcp` – denný percentuálny zisk/pokles priemernej ceny,
- `ema5/14/21/50` – exponenciálny kĺzavý priemer dĺžky *7/14/21/50*,
- `rsi` – relatívna sila,
- `macd` – kĺzavý priemer konvergencie/divergencie,
- `stoch` – stochastický oscilátor,
- `adl` – akumulačno-distribučné vedenie,
- `atr` – priemerný skutočný rozsah,
- `mom` – trhová dynamika,
- `mfi` – tok peňazí,
- `roc` – miera zmeny,

- `obv` – objem rovnováhy,
- `cci` – komoditný kanál,
- `emv` – jednoduchosť pohybu,
- `vortex` – vírenie.

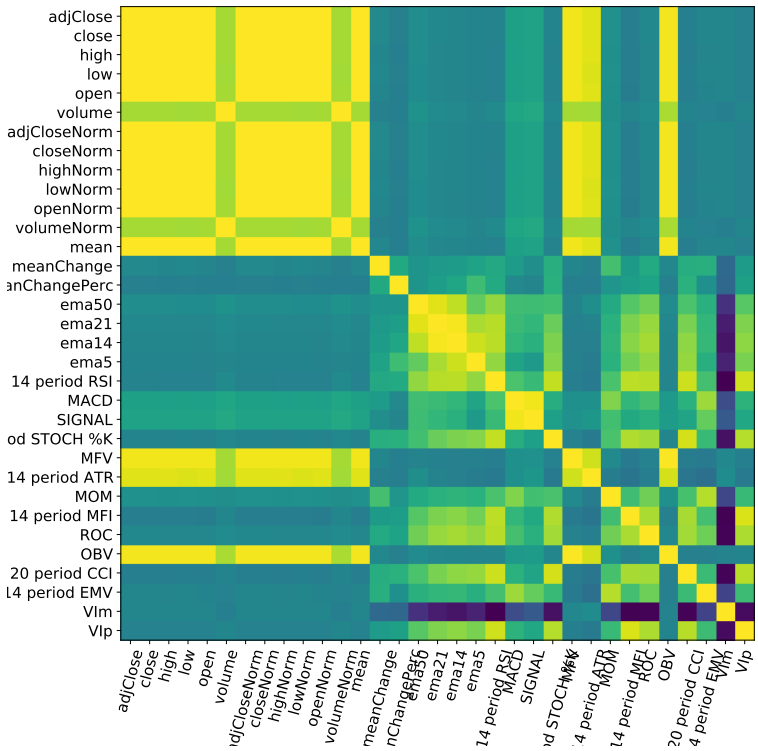
Väčšina uvedených indikátorov je vypočítaná pomocou balíka *FinTA*[4]. V dokumentácii balíka je možné nájsť podrobnejší význam uvedených indikátorov.

V dátovej sade je po pridaní celkovo *33* atribútov, čo je relatívne vysoký počet. Pre vstup jednotlivých experimentov je vhodné vybrať iba ich podmnožinu. Ďalšie sekcie sa budú zaoberať práve analýzou vhodnosti a následným výberom podmnožiny zdrojových údajov pre experimenty dolovania.

6.2.5 Korelačná analýza

Tak ako je uvedené v sekcii 3.2.5, korelačnou analýzou atribútov sú zisťované vzťahy medzi atribútmi. Vo všeobecnosti nie je vhodné vytvárať modely nad korelovanými atribútmi, nakoľko môžu spôsobiť zahltenie rozhodovania a znížiť tak presnosť modelu. Cieľom tejto časti je vyhľadať atribúty, ktoré sú korelované a znížiť ich počet.

Koreláciu medzi atribútmi je možné vypočítať metódou `corr` triedy `DataFrame` z knižnice *Pandas*. Výsledok výpočtu korelácie pre kryptomenu *Bitcoin* je vizualizovaný na obrázku 6.6.



Obr. 6.6: Korelačná analýza atribútov (kryptomena *Bitcoin*)

[4] https://github.com/peerchemist/finta/

Výsledok informuje o tom, že pôvodné atribúty a ich normalizované varianty sú do vysokej miery korelované. Rovnako je viditeľná korelácia aj medzi variantmi príznakov `EMA`. Z dátovej sady sú teda odstránené všetky pôvodné atribúty spolu s ich normalizovanými variantmi.

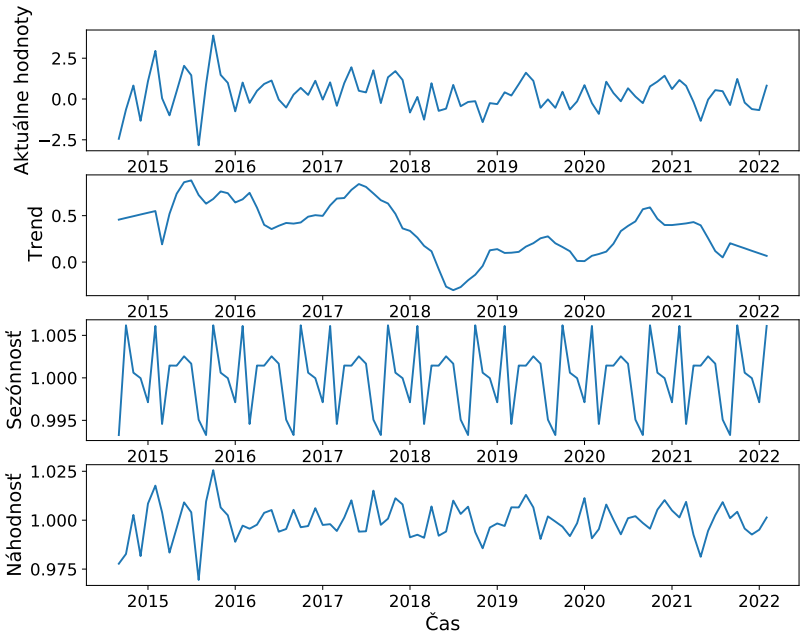
6.2.6 Analýza stacionarity

Nad atribútmi dátovej sady sú vykonané dva testy stacionarity: *ADH (Augmented Dickey Fullet)* test a *KPSS (Kwiatkowski Phillips Schmidt Shin)* test. Pre výpočet *ADH* a *KPSS* testu sú zvolené funkcie `adfuller` a `kpss` z balíka *Statsmodels*.

Výsledok *ADH* testu je možné interpretovať nasledovne: Ak je *P*-hodnota nižšia ako hladina významnosti *0,05*, tak je časový rad stacionárny. Najnižšiu *P*-hodnotu *2,716e-16* dosiahol atribút `avgcp` (nižšia hodnota ako hladina významnosti). Výsledok *KPSS* testu je možné interpretovať nasledovne: Ak je *P*-hodnota nižšia ako hladina významnosti *0,05*, tak je časový rad nestacionárny. Najvyššiu *P*-hodnotu v podobe *0,01* dosiahol atribút `avgcp` (vyššia hodnota ako hladina významnosti). Pre oba testy vyšlo, že najviac stacionárny je atribút `avgcp` (denná percentuálna zmena ceny). Modely v ďalších experimentoch budú trénované práve na tomto atribúte, nakoľko sa predpokladá, že dosiahne najlepšie výsledky.

6.2.7 Dekompozícia

Pre lepšie porozumenie vybraného atribútu `avgcp` je vhodné vykonať jeho **dekompozíciu** pomocou funkcie `seasonal_decompose` z balíka *Statsmodels*. Dekompozícia je realizovaná na multiplikatívne zložky trendu, sezónnosti a náhodnosti pre kryptomenu *Bitcoin*. K zvýšeniu prehľadnosti dekompozície sú hodnoty atribútu zredukované z dennej frekvencie na mesačnú. Vizualizáciu zložiek spolu so samotným atribútom je možné vidieť na obrázku 6.7.



Obr. 6.7: Dekompozícia percentuálnej zmeny dennej ceny (kryptomena *Bitcoin*)

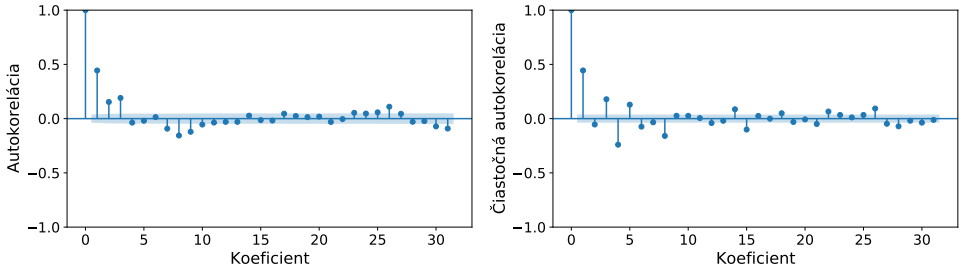
Z vizualizácie je možné vidieť, že atribút percentuálnej zmeny dennej ceny pre kryptomenu *Bitcoin* osciluje okolo nulovej hodnoty. Trendová zložka mierne klesá, jej vplyv je však

minimálny. Zo sezónnosti je viditeľné, že nárast ceny kryptomeny je najvyšší v priebehu začiatka roka.

6.2.8 Autokorelačná analýza

Posledná analýza v tejto podkapitole je zameraná na autokoreláciu a čiastostočnú autokoreláciu. Autokorelačnou analýzou je možné zistiť vzájomnú závislosť hodnôt v časovom rade. Informáciu o závislosti je následne možné využiť pre vhodný výber počtu hodnôt braných do úvahy pri vytváraní modelov predpovede a klasifikácie.

Autokorelácia a čiastočná autokorelácia je vypočítaná pomocou funkcií `acf` a `pacf` z balíka *Statsmodels*. Balík rovnako obsahuje funkcie `plot_acf` a `plot_pacf`, ktorými je možné vypočítané výsledky vykresliť. Výsledok autokorelačnej analýzy a čiastočnej autokorelačnej analýzy pre kryptomenu *Bitcoin* je vizualizovaný na obrázku 6.8.



Obr. 6.8: Autokorelačná a čiastočná autokorelačná analýza (kryptomena *Bitcoin*)

Z výsledku je možné vidieť, že hodnota autokorelácie je vyššia pre oneskorenia v intervaloch *1* až *3* a *7* až *9*. Pri trénovaní modelov predpovede je teda vhodné brať do úvahy tieto hodnoty. Hodnota čiastočnej autokorelácie sa výrazne priblíži k nule v hodnote *10*. Z pohľadu náhodnej zložky je teda vhodné brať do úvahy maximálne *9* predchádzajúcich hodnôt.

6.3 Predpoveď časových radov

Úvodný experiment je zameraný na demonštráciu a porovnanie metód predpovede časových radov. Cieľom experimentu je predpovedať vývoj budúcich cien kryptomeny *Bitcoin*. Ide o jednu z najznámejších a najziskovejších kryptomien v dátovej sade. V predošlej podkapitole z analýz vyplynulo, že najoptimálnejším atribútom pre predpoveď je `avgcp`, t. j. atribút dennej percentuálnej zmeny ceny. Vývoj je predpovedaný na najbližší deň počas obdobia posledných *30* dní, pričom pre dosiahnutie maximálnej presnosti je každý deň vykonané pretrénovanie modelu. Cena je z pohľadu denného nárastu/poklesu počas tohto obdobia zastúpená v pomere *15/15* a predstavuje celkovú zmenu *-0,95 %*.

Implementovaných je niekoľko metód z niekoľkých skupín: naivné metódy, metódy spriemerovania, metódy exponenciálneho vyhladzovania, metódy rodiny *SARIMAX*, viacsezónne metódy a metódy neurónových sietí. Metódy sú vyhodnotené v podobe metrík: stredná absolútna percentuálna chyba (`mape`), odmocnina strednej kvadratickej chyby (`rmse`), presnosť predpovede zisku/poklesu ceny (`acc`), získané zhodnotenie z predpovede zisku/poklesu ceny (`profit`) a spotrebovaný čas/pamäť.

6.3.1 Naivné metódy

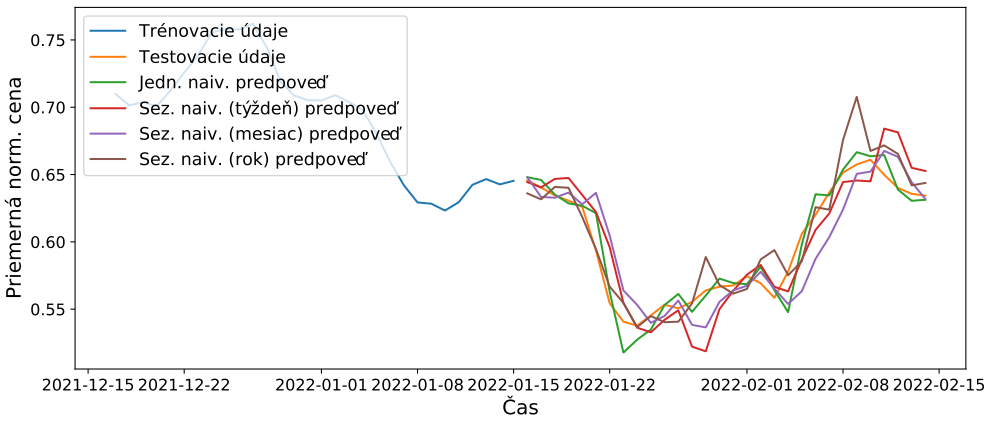
Prvou skupinou metód, ktoré sú v experimente demonštrované, sú naivné metódy. Tak ako bolo uvedené v podkapitole 3.3.1, ide o najjednoduchšie metódy, ktoré je vhodné použiť ako referenčné voči ostatným. **Základná naivná metóda** predpovedá budúce hodnoty na základe poslednej pozorovanej hodnoty. Metódu je možné implementovať jednoduchým prístupom k poslednému prvku časového radu:

```
result = BTC['avgpc'][-1]
```

Sezónna naivná metóda predpovedá budúce hodnoty na základe poslednej pozorovanej sezónnej hodnoty. Dĺžka periódy môže byť rôzna: týždeň, mesiac, rok atď. Metódu je opäť možné implementovať prístupom, tentokrát však s posunutím o dĺžku sezónnej periódy:

```
result = BTC['avgpc'][-SEASON_PERIOD_LENGTH]
```

Výsledok predpovedí naivných metód je vizualizovaný na obrázku 6.9.



Obr. 6.9: Vizualizácia predpovede naivných metód

Vyhodnotenie predpovedí naivných metód je uvedené v tabuľke 6.1.

	mape	rmse	corr	acc	profit	čas (s)	pamäť (MB)
Jedn. naiv.	*0,014*	*0,011*	*0,97*	*76,67 %*	*14,64 %*	*0,001*	*0,00*
Sez. naiv. (týž.)	*0,027*	*0,020*	*0,91*	*36,67 %*	*0,55 %*	*0,001*	*0,00*
Sez. naiv. (mes.)	*0,027*	*0,021*	*0,88*	*23,34 %*	*-19,19 %*	*0,001*	*0,00*
Sez. naiv. (rok)	*0,021*	*0,017*	*0,94*	*56,67 %*	*10,53 %*	*0,001*	*0,00*

Tabuľka 6.1: Presnosť predpovede naivných metód

Z vizualizácie a tabuľky je možné vidieť, že najpresnejšou z metód je jednoduchá naivná metóda. Príčinou je fakt, že denný percentuálny nárast/pokles je vysoko závislý od predchádzajúceho dňa. Zo sezónnych naivných metód vrátila najpresnejšie výsledky ročná dĺžka periódy, čo značí, že daná sezónna perióda má najvyšší vplyv na predpovedané hodnoty. Všetky metódy sú rovnako časovo a pamäťovo vysoko efektívne, nakoľko ide iba o jednoduchý prístup do dátového rámca.

6.3.2 Metódy spriemerovania

Metódy spriemerovania modelujú predpovedanú hodnotu ako priemer minulých hodnôt. **Jednoduchá metóda spriemerovania** uvažuje pri spriemerovaní všetky hodnoty časového radu. Implementáciu je možné realizovať metódou `mean` dátového rámca z balíka *Pandas*:

```
result = BTC['avgpc'].mean()
```

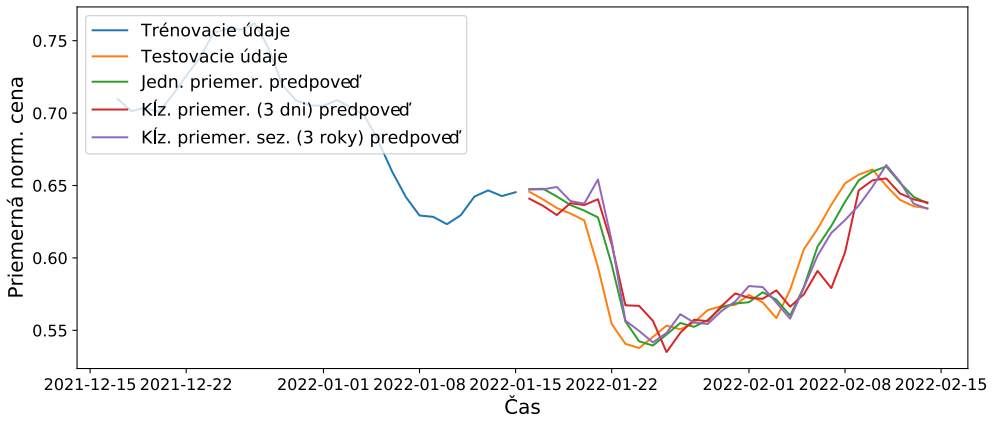
Metóda kĺzavého spriemerovania uvažuje v spriemerovaní iba posledných n hodnôt. Implementácia metódy je nasledovná:

```
result = BTC['avgpc'][-N:].mean()
```

Metóda kĺzavého sezónneho spriemerovania kombinuje kĺzavé spriemerovanie dĺžky n so sezónnosťou dĺžky m. Metódu je možné realizovať nasledovne:

```
for in in range(M):
    season_date = i*SEASON_PERIOD_LENTH
    result[i] = BTC['avgpc'][-season_date-N:-season_date]
result.mean()
```

Vizualizácia predpovedí je ukázaná na obrázku 6.10.



Obr. 6.10: Vizualizácia predpovede metód spriemerovania

Vyhodnotenie predpovedí s optimálnymi parametrami n a m sa nachádza v tabuľke 6.2.

	`mape`	`rmse`	`corr`	`acc`	`profit`	čas (s)	pamäť (MB)
Jednoduché	*0,017*	*0,014*	*0,95*	*50,00 %*	*-0,95 %*	*0,035*	*0,00*
Kĺz. ***(3)***	*0,027*	*0,023*	*0,83*	*43,33 %*	*-9,76 %*	*0,025*	*0,00*
Kĺz. sez. ***(3,3)***	*0,024*	*0,020*	*0,89*	*33,34 %*	*-12,45 %*	*0,081*	*0,00*

Tabuľka 6.2: Presnosť predpovede metód spriemerovania

Vizualizácia a tabuľka informuje o tom, že najpresnejší výsledok vracia metóda jednoduchého spriemerovania. Dôvodom je, že spriemerovaním všetkých hodnôt časového radu je hodnota predpovedanej percentuálnej zmeny blízka nule, čo znamená, že predpovedaná cena je takmer zhodná s predošlým dňom. Najvyššiu časovú náročnosť dosahuje metóda kĺzavého sezónneho spriemerovania, nakoľko je potrebné lokalizovať historické hodnoty.

6.3.3 Metódy exponenciálneho vyhladzovania

Skupina týchto metód modeluje predpoveď ako exponenciálne váženú lineárnu funkciu minulých hodnôt. **Metóda jednod. exp. vyhladzovania (*SES*)** je základným variantom týchto metód. Reprezentantom je trieda `SimpleExpSmoothing` z balíka *Statsmodels*. Vstupný parameter `smoothing_level` určuje mieru vyhladenia. Metóda dosahuje najlepší výsledok pre hodnotu *0.65*. Vytvorenie, natrénovanie a použitie modelu je vykonané nasledovne:

```
model = SimpleExpSmoothing(BTC).fit(smoothing_level=0.65)
result = model_fit.forecast(1)
```

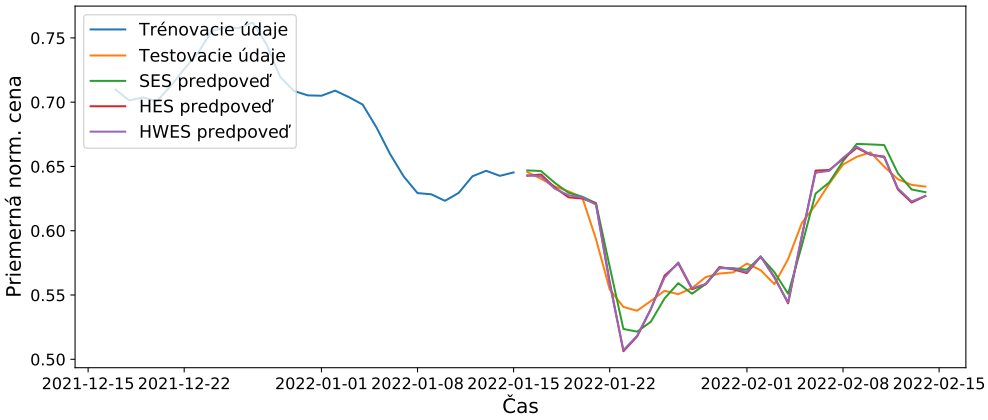
Metóda Holt exp. vyhladzovania (*HES*) oproti predošlej metóde berie do úvahy trendovú zložku časového radu. Jej predstaviteľom je trieda `Holt` z balíka *Statsmodels*. Metóda má na vstupe navyše parameter `smoothing_trend` definujúci mieru vyhladenia trendu. Najlepší výsledok je vrátený pre hodnotu *0.7*. Vytvorenie, natrénovanie a použitie modelu je nasledovné:

```
model = Holt(BTC).fit(smoothing_level=0.65, smoothing_trend=0.7)
result = model_fit.forecast(1)
```

Metóda Holt-Winters exp. vyhladzovania (*HWES*) oproti základnému variantu zohľadňuje trendovú a sezónnu zložku. Implementáciu zapuzdruje trieda `ExponentialSmoothing` z balíka *Statsmodels*. Vstupom naviac sú parametre popisujúce typ trendovej zložky, typ sezónnej zložky a dĺžku sezónnej periódy. Typ zložiek je stanovený na aditívny, dĺžka periódy na štvrťrok. Vytvorenie, natrénovanie a použitie modelu je nasledovné:

```
model = ExponentialSmoothing(
  BTC, seasonal_periods=4, trend="add", seasonal="add"
).fit(smoothing_level=0.65, smoothing_trend=0.7)
result = model_fit.forecast(1)
```

Výsledok predpovedí jednotlivých metód je vizualizovaný na obrázku 6.11.



Obr. 6.11: Vizualizácia predpovede metód exponenciálneho vyhladzovania

Vyhodnotenie predpovedí metód exp. vyhladzovania je uvedené v tabuľke 6.3.

	mape	rmse	corr	acc	profit	čas (s)	pamäť (MB)
SES	*0,015*	*0,011*	*0,97*	*80,00 %*	*15,51 %*	*0,88*	*3,46*
HES	*0,018*	*0,014*	*0,95*	*83,33 %*	*18,16 %*	*2,47*	*3,59*
HWES	*0,017*	*0,014*	*0,95*	*86,67 %*	*19,37 %*	*3,42*	*3,69*

Tabuľka 6.3: Presnosť predpovede metód exponenciálneho vyhladzovania

Z vizualizácie a tabuľky je možné vidieť, že najlepšiu presnosť dosiahol model *HWES*, čo je zapríčinené faktom zohľadnenia sezónnosti. Model *HES* vylepšuje výsledky modelu *SES*, nakoľko zohľadňuje trend. Z tabuľky je rovnako viditeľné, že zohľadnenie jednotlivých zložiek zvyšuje dobu výpočtu a spotreb. pamäť.

6.3.4 Metódy rodiny *SARIMAX*

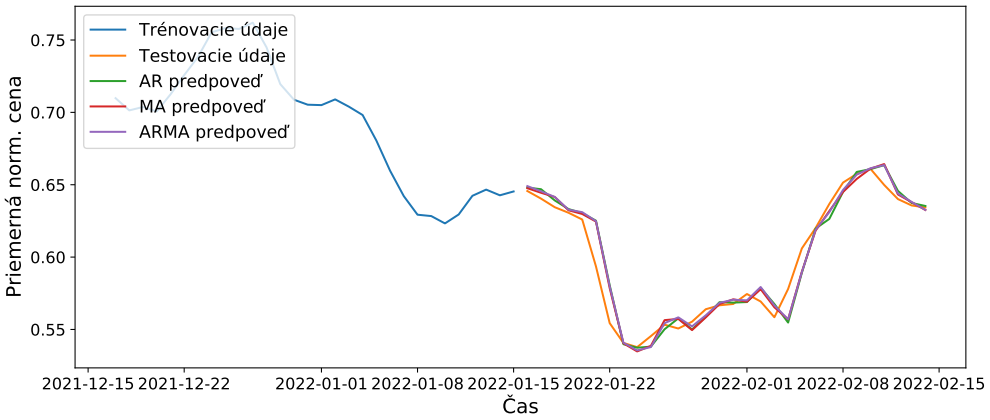
Princíp fungovania metód *SARIMAX* spočíva v predpovedi budúcich hodnôt pomocou váženej lineárnej funkcie predošlých hodnôt. Implementáciu všetkých metód rodiny *SARIMAX* zastupuje trieda `ARIMA` z balíka *Pmdarima*.

Základným variantom je metóda ***AR***. Vstupným parametrom metódy je p, ktorý definuje počet predchádzajúcich hodnôt braných do úvahy pri odhade lineárnej funkcie. Vstupným parametrov metódy ***MA*** je q, ktorý udáva počet hodnôt braných do úvahy pri odhade chybovej lineárnej funkcie. Metóda ***ARMA*** pri predpovedi berie do úvahy lineárnu funkciu a aj lineárnu chybovú funkciu. Vstupom sú teda oba parametre p a q. Metóda ***ARIMA*** je v podobe viacnásobného rozdielu susedných hodnôt rozšírená o prevod časového radu na stacionárny. Počet rozdielov udáva parameter d. Metóda ***SARIMA*** oproti predošlej metóde berie do úvahy aj sezónnu zložku časového radu. Vstupom metódy je sedem parametrov, prvé tri sa zhodujú s predošlou metódu a zvyšné štyri definujú *ARIMA* parametre sezónnej zložky a dĺžku jej periódy. Metóda ***SARIMAX*** k predchádzajúcemu modelu navyše pridáva exogénnu (dodatočnú) premennú. Vytvorenie, natrénovanie a použitie modelov pre jednotlivé metódy je vykonané nasledovne:

```
model = ARIMA(
   order=(p,d,q),
   seasonal_order=(P,D,Q,SEASONAL_PERIOD)
).fit(BTC)
result = model_fit.forecast(1)
```

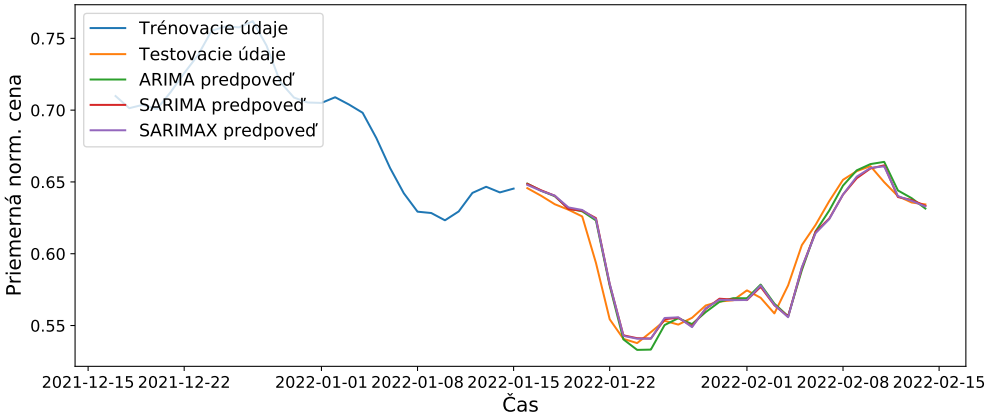
Z autokorelačnej analýzy v predošlej podkapitole vyplynulo, že pri predpovedi je vhodné uvažovať maximálne *9* minulých hodnôt pre lineárnu a lineárnu chybovú funkciu. Pre metódy sú hľadané optimálne hodnoty práve z tohto intervalu. Hodnoty parametrov sa pre jednotlivé metódy odlišujú: *AR(2)*, *MA(5)*, *ARMA(1,4)*, *ARIMA(3,1,2)*, *SARIMA(1,1,2,2,0,0)* – štvrťročná sezónna perióda, *SARIMAX(1,1,2,2,0,0)* – štvrťročná sezónna perióda a ročná sezónna perióda ako exogénna premenná.

Výsledok predpovedí metód *AR*, *MA*, *ARMA* je vizualizovaný na obrázku 6.12.



Obr. 6.12: Vizualizácia predpovede metód *AR*, *MA*, *ARMA*

Výsledok predpovedí zvyšných troch metód je vizualizovaný na obrázku 6.13.



Obr. 6.13: Vizualizácia predpovede metód *ARIMA*, *SARIMA*, *SARIMAX*

Vyhodnotenie predpovedí z rodiny *SARIMAX* je uvedené v tabuľke 6.4.

	`mape`	`rmse`	`corr`	`acc`	`profit`	čas (s)	pamäť (MB)
AR	*0,012*	*0,010*	*0,97*	*80,00 %*	*16,06 %*	*7,49*	*14,08*
MA	*0,012*	*0,010*	*0,97*	*80,00 %*	*15,96 %*	*24,94*	*46,11*
ARMA	*0,012*	*0,010*	*0,97*	*80,00 %*	*16,07 %*	*24,50*	*34,97*
ARIMA	*0,012*	*0,010*	*0,97*	*83,33 %*	*16,32 %*	*27,35*	*27,02*
SARIMA	*0,012*	*0,010*	*0,97*	*83,33 %*	*16,34 %*	*108,45*	*90,34*
SARIMAX	*0,012*	*0,010*	*0,97*	*83,33 %*	*17,10 %*	*288,15*	*111,39*

Tabuľka 6.4: Presnosť predpovede metód rodiny *SARIMAX*

Z vizualizácie a tabuľky s presnosťou je možné pozorovať, že s pribúdajúcimi zložkami rastie presnosť predpovede, čas výpočtu a spotreb. pamäť. Zmeny však nie sú výrazné, čo je spôsobené faktom, že zvolený atribút je už vopred stacionárny s malou mierou sezónnosti.

6.3.5 Viacsezónne metódy

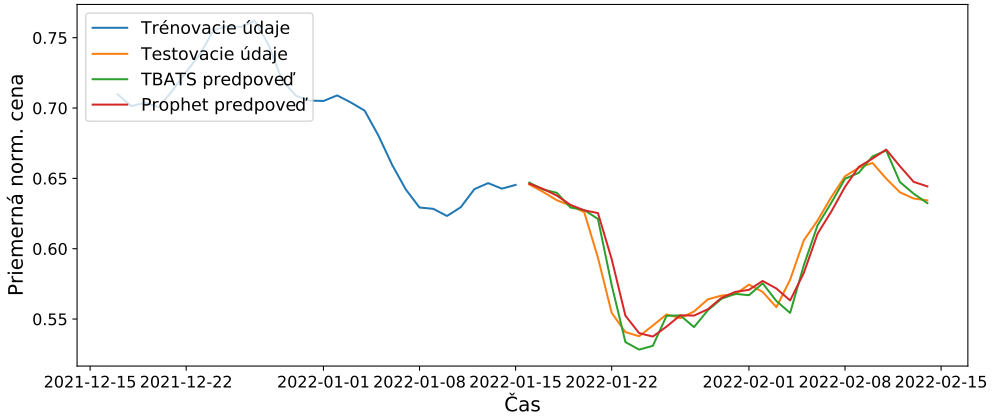
V tejto podkapitole sú demonštrované a porovnané viacsezónne metódy. Ide o metódy, ktoré rozširujú doposiaľ uvedené metódy o viacnásobnú sezónnosť. Metóda ***TBATS (Trigonometric, Box-Cox transform, ARMA errors, Trend Seasonal components)*** je metóda exponenciálneho vyhladzovania, ktorá je rozšírená o viaceré sezónnosti. Funkcionalita metódy je implementovaná v triede `TBATS` z balíka *Sktime*. Vstupom metódy je pole období sezónnosti (počet dní). Vytvorenie, natrénovanie a použitie modelu je nasledovné:

```
model = TBATS(sp=[7,30.5,365.25]).fit(BTC)
result = model.forecast(fh=1)
```

Ďalšou viacsezónnou metódou je aditívny lineárny regresor ***Prophet***. Metódu implementuje trieda `Prophet` z balíka *prophet*. Vstupnými parametrami triedy sú nastavenie rôznych typov sezónnosti a definícia dňov prázdnin, ktoré v rámci kryptomeny *Bitcoin* reprezentujú dni so znížením ceny na polovicu. Vytvorenie, natrénovanie a použitie modelu je nasledovné:

```
model = Prophet(
    yearly_seasonality=True, weekly_seasonality=True,
    daily_seasonality=True, holidays=BTC_halving
).fit(BTC)
result = model.predict(1)
```

Vizualizácia predpovedí viacsezónnych metód je vizualizovaná na obrázku 6.14.



Obr. 6.14: Vizualizácia predpovede viacsezónnych metód

Vyhodnotenie viacsezónnych metód sa nachádza v tabuľke 6.5.

	`mape`	`rmse`	`corr`	`acc`	`profit`	čas (s)	pamäť (MB)
TBATS	*0,013*	*0,010*	*0,97*	*80,00 %*	*14,73 %*	*224,53*	*182,31*
Prophet	*0,015*	*0,013*	*0,96*	*73,33 %*	*14,54 %*	*128,71*	*133,14*

Tabuľka 6.5: Presnosť predpovede metód neurónových sietí

Z tabuľky je možné pozorovať, že model *TBATS* dosiahol mierne presnejšie výsledky, avšak za cenu vyššej časovej a pamäťovej náročnosti.

6.3.6 Metódy neurónových sietí

Poslednou prezentovanou skupinou sú metódy, ktoré k predpovedi využívajú model neurónovej siete. Jednou z najzákladnejších neurónových sietí je sieť ***MLP (Multi Layer Perceptron)***. Tento typ neurónovej siete je zastúpený triedou `TimeSeriesMLPRegressor` z balíka *Tslearn*. Vstupnými parametrami siete sú rozmery skrytých vrstiev, typ optimalizačnej trénovacej funkcie a typ aktivačnej funkcie. Pri trénovaní je ako optimalizačná metóda zvolená stochastický gradientný zostup. Ako aktivačná funkcia je vybratá *ReLU (Rectified Linear Unit)*. Vytvorenie, natrénovanie a použitie sa uskutočňuje nasledovne:

```
model = TimeSeriesMLPRegressor(
  hidden_layer_sizes=(64,64), solver='adam', activation='relu'
).fit(BTC_train_x, BTC_train_y)
result = model.predict(BTC_test_x)
```

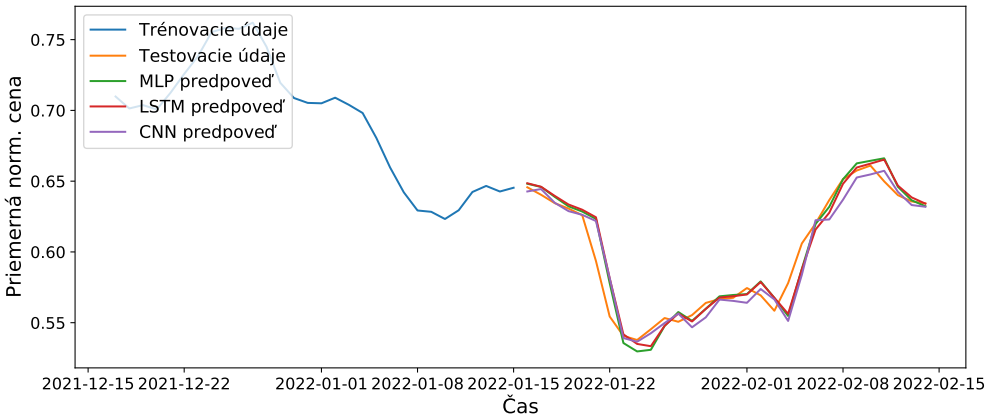
Iným typom je neurónová sieť ***LSTM (Long Short Term Memory)***. Sieť je možné implementovať pomocou balíka *Keras*. Balík poskytuje triedu *Sequential*, ktorá predstavuje neurónovú sieť s jednovstupnými a jednovýstupnými vrstvami. Do siete sú pridané nasledovné vrstvy: `LSTM` vrstva, plne prepojená skrytá vrstva `Dense`, `Dropout` vrstva reprezentujúca mieru straty hodnoty, lineárna aktivačná vrstva `Activation`. Pri trénovaní je ako optimalizačná metóda zvolená stochastický gradientný zostup so stratovou funkciou priemernej štvorcovej chyby. Vytvorenie, natrénovanie a použitie je možné vykonať nasledovne:

```
model = Sequential()
model.add(LSTM(128, input_shape=(1, N)))
model.add(Dropout(0.1))
model.add(Dense(1))
model.add(Activation('linear'))
model.compile(optimizer='adam', loss='mse')
model.fit(BTC_train_x, BTC_train_y)
result = model.predict(BTC_test_x)
```

Pre predpoveď je rovnako možné použiť neurónovú sieť ***CNN (Convolutional neural network)***. Sieť je implementovaná rovnako ako v predošlom prípade. Zmenou sú použité vrstvy: konvolučná vrstva `Conv1D`, redukčná vrstva `MaxPooling1D`, splošťujúca vrstva `Flatten`. Vytvorenie, natrénovanie a použitie siete spolu s aplikovanými parametrami je nasledovné:

```
model = Sequential()
model.add(Conv1D(
  filters=64, kernel_size=2, activation='relu', input_shape=(N,1)
))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.fit(BTC_train_x, BTC_train_y)
result = model.predict(BTC_test_x)
```

Výsledok predpovedí jednotlivých metód je vizualizovaný na obrázku 6.15.



Obr. 6.15: Vizualizácia predpovede metód neurónových sietí

Vyhodnotenie predpovedí neurónových sietí je uvedené v tabuľke 6.6.

	`mape`	`rmse`	`corr`	`acc`	`profit`	čas (s)	pamäť (MB)
MLP	*0,013*	*0,011*	*0,97*	*80,00 %*	*15,51 %*	*189,56*	*56,98*
LSTM	*0,012*	*0,011*	*0,97*	*83,33 %*	*16,34 %*	*434,93*	*260,72*
CNN	*0,013*	*0,012*	*0,96*	*73,33 %*	*15,97 %*	*245,12*	*109,96*

Tabuľka 6.6: Presnosť predpovede metód neurónových sietí

Tabuľka a vizualizácia informuje, že najpresnejší výsledok dosiahla sieť *LSTM*. Druhý najlepší výsledok z pohľadu presnosti dosiahla sieť *MLP*. Sieť *CNN* však dosiahla vyššie zhodnotenie. Zdrojovo najnáročnejšou je sieť *LSTM*, naopak, najmenej náročnou sieť *MLP*.

6.3.7 Zhodnotenie predpovede

V experimente bolo otestovaných celkovo *21* modelov z rôznych skupín. Jednotlivé skupiny sa vzhľadom na vlastnosti vzájomne odlišujú a to preukázali aj dosiahnuté výsledky.

Celkovo najpresnejšiu predpoveď dosiahla metóda *SARIMAX* z hodnotou *MAPE 0,012*. Z pohľadu úspešnosti predpovede poklesu/nárastu ceny a odpovedajúceho zhodnotenia dosiahla v podobe hodnôt *86,67 %* a *19,37 %* najlepší výsledok metóda *HWES*. Ako časovo a pamäťovo najefektívnejšie sa ukázali naivné metódy a metódy spriemerovania. Efektívnosť využitia zdrojov však v tom prípade spôsobuje aj nízku presnosť. Výnimkou je jednoduchá naivná metóda, ktorá dosiahla vyššiu presnosť z dôvodu vysokej závislosti hodnôt atribútu na predošlom dni. Najmenej časovo a pamäťovo efektívnou vyšla skupina viacsezónnych metód a metód neurónových sietí. Viacsezónne metódy nedosiahli výrazne lepšie výsledky, nakoľko v zvolenom atribúte je prítomná len nízka miera sezónností. Metódy neurónových sietí dosiahli slabšie výsledky z dôvodu úplného ignorovania sezónnosti a dlhodobého trendu.

Z výsledkov predpovede percentuálnej zmeny ceny kryptomeny *Bitcoin* je možné vyvodiť nasledovné závery: Ak je potreba maximalizovať rýchlosť predpovede pre veľké množstvo kryptomien, je vhodné zvoliť metódy jednod. exp. vyhladzovania alebo jednod. naivnej predpovede. V prípade, že je snaha o maximálne presnú predpoveď, je vhodné využiť konzervatívnu metódu *SARIMAX* alebo rizikovejšiu metódu *HWES*.

6.4 Klasifikácia časových radov

Cieľom tohto experimentu je pomocou rôznych modelov klasifikovať vývoj ceny na ziskový alebo stratový. Ide teda o binárnu klasifikáciu do dvoch tried.

Rovnako ako v predošlom experimente je klasifikácia vykoná pre atribút `avgcp` (denná percentuálna zmena ceny) kryptomeny *Bitcoin*. Oproti predošlému experimentu je potrebné časový rad rozdeliť na n častí, kde $n = l$ - m, l je dĺžka časového radu, m je dĺžka časti. Dĺžka častí odpovedá počtu dní, z ktorých je žiadúce klasifikáciu vykonať. Každej časti je priradená trieda `H` (nárast) alebo `L` (pokles) podľa hodnoty nasledovného dňa. Každý klasifikačný model je natrénovaný na dĺžke častí *1* až *31*. Model s najlepším výsledkom je použitý pre finálne testovanie, ktoré spočíva v klasifikácii posledných *30* dní časového radu. Model je každý deň pretrénovaný, tak aby vracal max. presné výsledky. Cena je z pohľadu denného nárastu/poklesu počas tohto obdobia zastúpená v pomere *15/15*, čo predstavuje celkovú zmenu *-0,95 %*.

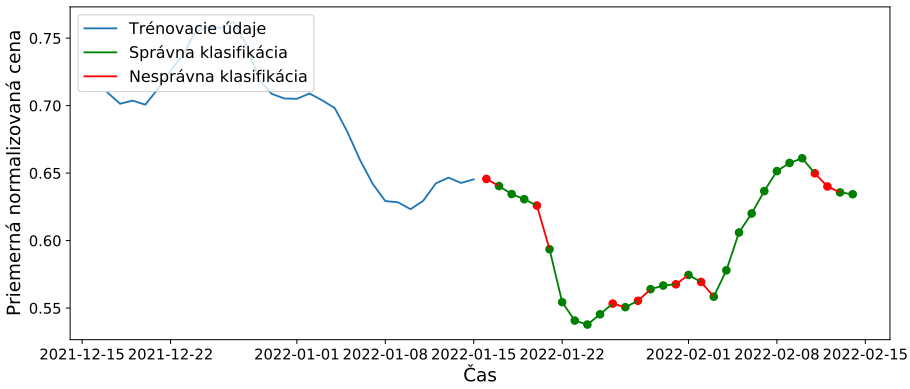
Implementovaných je niekoľko metód klasifikácie z rôznych kategórií: vzdialenostné, intervalové, frekvenčné, slovníkové, tvarové metódy a metódy neurónových sietí. Vyhodnotenie metód je realizované vizualizáciou a uvedením presnosti predpovede zisku/poklesu (`acc`), dosiahnutého zhodnotenia (`profit`) a časovej/pamäťovej náročnosti.

6.4.1 Metódy založené na vzdialenosti

Metódy založené na vzdialenosti podľa vopred definovanej metriky klasifikujú surové časové rady na základe ich vzájomných vzdialeností. Princíp činnosti metód je vhodné demonštrovať na klasifikátore k-najbližších susedov. Klasifikátor ***KNN* (*k*-najbližších susedov)** je reprezentovaný triedou `KNeighborsTimeSeriesClassifier` z balíka *Tslearn*. Vstupnými parametrami modelu je počet susedov braných do úvahy pri klasifikácii a vzdial. metrika. Vytvorenie, natrénovanie a použitie modelu spolu so zvolenými parametrami je nasledovné:

```
model = tslearn_KNeighborsTimeSeriesClassifier(
    n_neighbors=1, metric='euclidean'
).fit(BTC_train_x, BTC_train_y)
result = model.predict(BTC_test_x)
```

Model dosiahol najlepšiu presnosť pre dĺžku vzorky *7* dní. Výsledok klasifikácie pre metriku Euklidovej vzdialenosti je uvedený na obrázku 6.16.



Obr. 6.16: Výsledok klasifikácie *KNN* klasifikátora (metrika `euclidean`)

Okrem vzdialenostnej funkcie `euclidean` je model vytvorený pre viacero ďalších metrík: `sqeuclidean`, `dtw`, `softdtw`, `ctw`, `cityblock`. Ich vyhodnotenie je uvedené v tabuľke 6.7.

	dĺžka	`acc`	`profit`	čas (s)	pamäť (MB)
`euclidean`	*7*	*73,33 %*	*16,96 %*	*7,33*	*0,00*
`sqeuclidean`	*7*	*73,33 %*	*16,96 %*	*6,81*	*0,00*
`cityblock`	*7*	*76,67 %*	*13,66 %*	*8,08*	*0,00*
`dtw`	*2*	*73.33 %*	*11,43 %*	*17,16*	*0,00*
`softdtw`	*9*	*76,67 %*	*14,64 %*	*25,01*	*0,00*
`ctw`	*9*	*70,00 %*	*11,45 %*	*95,52*	*0,00*

Tabuľka 6.7: Presnosť *KNN* klasifikátora

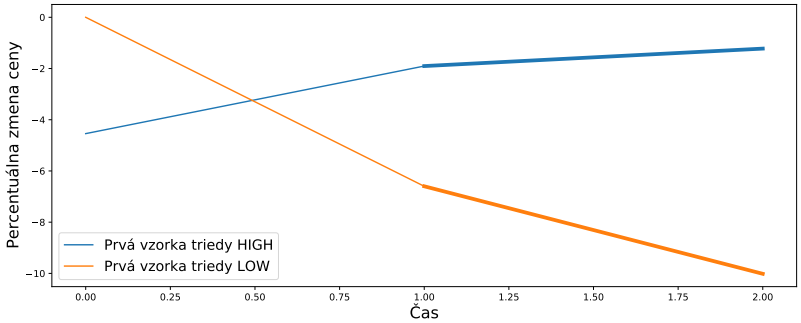
Z vizualizácie a tabuľky s presnosťou je možné vidieť, že najlepší výsledok z pohľadu presnosti dosiahli metriky `cityblock` a `softdtw`. Najziskovejšie a najrýchlejšie však boli metriky `euclidean` a `sqeuclidean`. Najmenej presný výsledok vyšiel pre metriku `ctw`, ktorej výpočet trval aj najdlhšie.

6.4.2 Metódy založené na intervaloch

Metódy tohto typu využívajú pre klasifikáciu informáciu obsiahnutú v intervaloch rozdeleného časového radu. Konkrétnym príkladom je klasifikátor ***Time Series Forest (TSF)***, ktorý je zastúpený triedou `TimeSeriesForest` z balíka *Pyts*. Klasifikátor na vstupe vyžaduje počet stromov lesa, počet okien a minimálnu dĺžku okna. Vytvorenie, natrénovanie a použitie modelu spolu s aplikovanými parametrami je nasledovné:

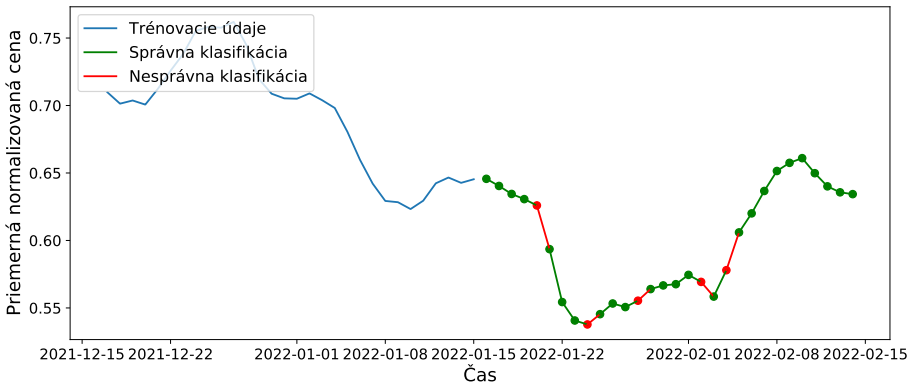
```
model = TimeSeriesForest(
    n_estimators=100, n_windows=1.0, min_window_size=1
).fit(BTC_train_x, BTC_train_y)
model.predict(BTC_test_x)
```

Najlepší výsledok dosiahol klasifikátor pre dĺžku vzorky *3* dní. Vplyv hodnôt rozdelených častí na klasifikáciu podľa skóre dôležitosti je vizualizovaný na obrázku 6.17. Zvýraznená časť informuje, že posledné 2 hodnoty rozdelených častí majú najväčší vplyv na výsledok klasifikácie.



Obr. 6.17: Vplyv hodnôt na klasifikáciu (prvá vzorka tried)

Výsledok *TSF* klasifikácie je vizualizovaný na obrázku 6.18.



Obr. 6.18: Výsledok klasifikátora *TSF*

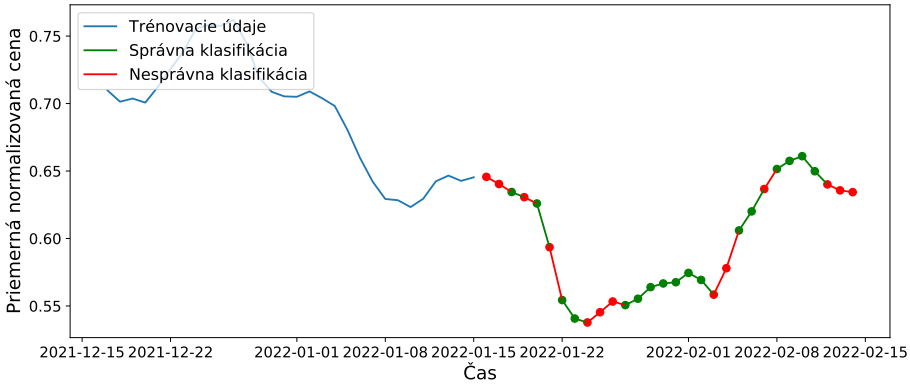
TSF klasifikátor dosiahol presnosť *83,34 %* a zhodnotenie *17.76 %*. Celkový čas výpočtu trval *9,62* sekundy, pričom bolo spotrebovaných *31,93 MB* pamäte.

6.4.3 Metódy založené na frekvencii

Metódy z tejto skupiny pred samotným procesom klasifikácie vykonávajú prevod časového radu do frekvenčnej domény. Príkladom je klasifikátor ***RISE (Random Interval Spectral Ensemble)***, ktorý je zastúpený triedou `RandomIntervalSpectralEnsemble` z balíka *Sktime*. Vstupom je počet stromov lesa a min. dĺžka okna. Vytvorenie, natrénovanie a použitie modelu je nasledovné:

```
model = RandomIntervalSpectralEnsemble(
  n_estimators=100, min_interval=4
).fit(BTC_train_x, BTC_train_y)
model.predict(BTC_test_x)
```

Klasifikácia dosiahla najlepší výsledok pre vzorky dĺžky *5* dní. Výsledok *RISE* klasifikácie je vizualizovaný na obrázku 6.19.



Obr. 6.19: Výsledok klasifikátora *RISE*

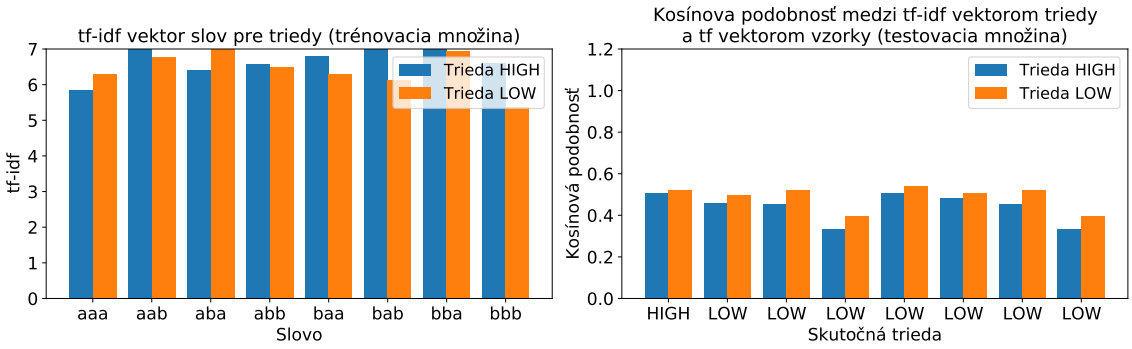
Presnosť *RISE* klasifikácie dosiahla hodnotu *56,67 %*, čo predstavuje zhodnotenie *3,77 %*. Výpočet spotreboval *34,62 MB* pamäte, počas *76,8* sekundovej doby výpočtu.

6.4.4 Metódy založené na slovníku slov

Princíp slovníkových metód spočíva v prevode časového radu na sekvenciu slov, pričom samotná klasifikácia sa realizuje na základe ich distribúcie. Konkrétnym príkladom je klasifikátor ***BOSSVS (Bag-of-SFA Symbols in Vector Space)***, ktorý je implementovaný triedou `BOSSVS` z balíka *Pyts*. Na vstupe prijíma parametre veľkosť slova, počet použitých znakov a veľkosť okna. Vytvorenie, natrénovanie a použitie modelu je nasledovné:

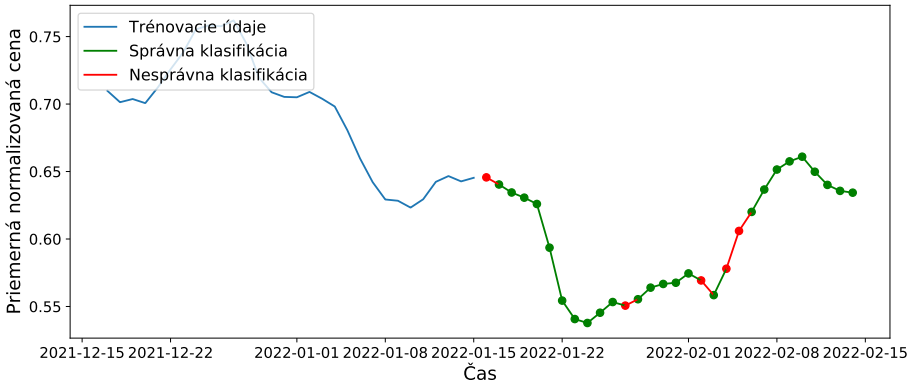
```
model = BOSSVS(
    word_size=3, n_bins=2, window_size=3
).fit(BTC_train_x, BTC_train_y)
model.predict(BTC_test_x)
```

Model dosiahol najlepší výsledok pre vzorky s dĺžkou *4*. *Tf-idf* vektor slov pre triedy a následná kosínová podobnosť s prvými ôsmimi vzorkami je vizualizovaná na obrázku 6.20.



Obr. 6.20: Analýza klasifikátora *BOSSVS*

Výsledok *BOSSVS* klasifikácie je vizualizovaný na obrázku 6.21.



Obr. 6.21: Výsledok klasifikátora *BOSSVS*

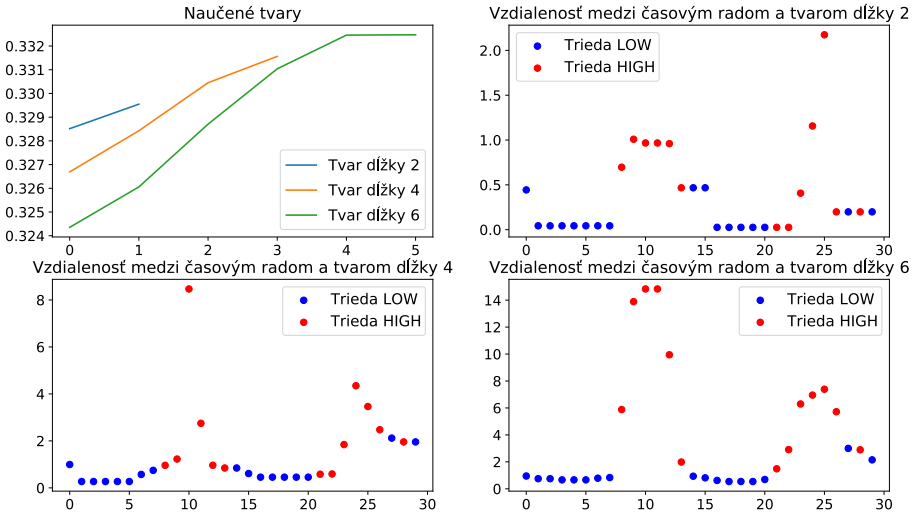
Klasifikácia dosiahla presnosť *83,34 %*, čo v prípade využitia modelu k strategickému rozhodovaniu znamená zhodnotenie *18,15 %*. Výpočet trval *15,38* sekundy a spotreboval *18,46 MB* pamäte.

6.4.5 Metódy založené na tvaroch

Metódy založené na tvaroch realizujú klasifikáciu na základe podsekvencií časového radu s najväčšou mierou odlišnosti. Príkladom algoritmu je ***ST (Shapelet Transform)*** klasifikátor, ktorý je reprezentovaný triedou `LearningShapelets` z balíka *Pyts*. Algoritmus na vstupe požaduje počet tvarov pre dĺžku, minimálnu dĺžka tvaru a škálu dĺžok tvarov. Vytvorenie, natrénovanie a použitie modelu je nasledovné:

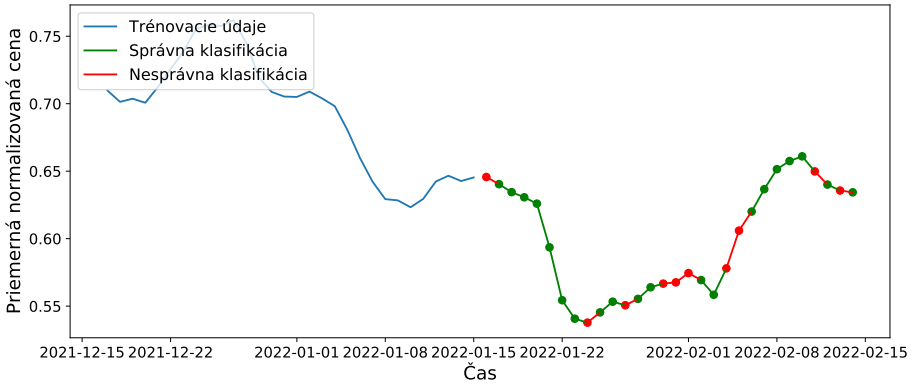
```
model = LearningShapelets(
  n_shapelets_per_size=1, min_shapelet_length=2, shapelet_scale=3
).fit(BTC_train_x, BTC_train_y)
model.predict(BTC_test_x)
```

Najlepšie výsledky dosiahli vzorky dĺžky *8* dní. Analýza modelu je zobrazená na obr. 6.22. V ľavom hornom obrázku sú vizualizované naučené tvary. V zvyšných troch ich vzdialenosť voči testovacím vzorkám, kde je možné pozorovať, že najlepšie rozlišuje tvar dĺžky *6*.



Obr. 6.22: Analýza klasifikátora *ST*

Výsledok *ST* klasifikácie je uvedený na obrázku 6.23.

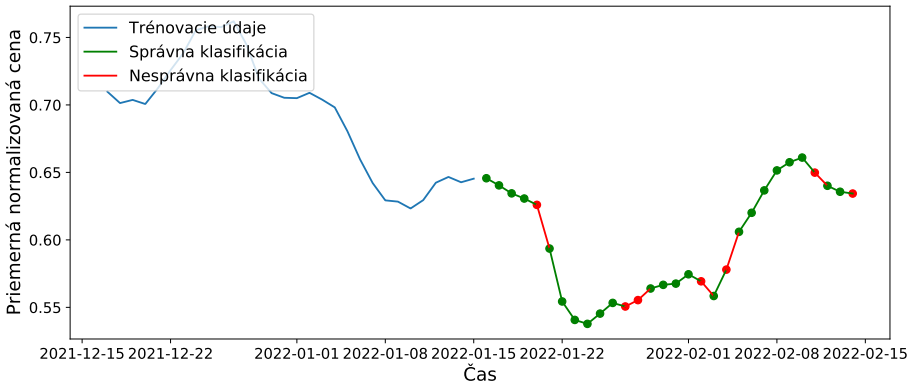


Obr. 6.23: Výsledok klasifikátora *ST*

Presnosť klasifikácie dosiahla hodnotu *66,67 %*, čo predstavuje zhodnotenie *10.66 %*. Výsledok sa počítal *202,11* sekúnd a spotreboval *46,75 MB* pamäte.

6.4.6 Metódy neurónových sietí

Posledným typom klasifikačných metód, ktoré sú demonštrované v tomto experimente, sú metódy neurónových sietí. Rovnako ako v predošlom experimente predpovede sú použité siete *MLP*, *LSTM* a *CNN*. Vstupné parametre neurónových sietí sú takisto zachované. Jedinou zmenou z predošlého experimentu je, že siete majú pri trénovaní na výstupe namiesto percentuálnej zmeny dve hodnoty: `0.0` reprezentujúca pokles ceny, `1.0` reprezentujúca nárast ceny. Výsledok klasifikácie neurónovej siete *MLP* je vizualizovaný na obrázku 6.24.



Obr. 6.24: Výsledok klasifikácie neurónovej siete *MLP*

Vyhodnotenie klasifikácie metód neurónových sietí pre použitú dĺžku vzorky je uvedené v tabuľke 6.8.

	dĺžka	`acc`	`profit`	čas (s)	pamäť (MB)
MLP	*3*	*80,00 %*	*16,14 %*	*182,35*	*51,49*
LSTM	*3*	*76,67 %*	*14,65 %*	*393,18*	*259,46*
CNN	*3*	*76,67 %*	*15,75 %*	*202,21*	*103,72*

Tabuľka 6.8: Presnosť klasifikácie neurónových sietí

Najlepšiu presnosť, rýchlosť a pamäťovú náročnosť dosiahla sieť *MLP*. Naopak, najmenej presný výsledok vyšiel pre sieť *CNN*. Najmenej časovo a pamäťovo efektívna je sieť *LSTM*.

6.4.7 Zhodnotenie klasifikácie

V rámci experimentu bolo implementovaných *13* modelov klasifikácie využívajúcich rôzne údaje a postupy. Odlišnosť vlastností klasifikátorov potvrdili aj dosiahnuté výsledky.

Najvyššiu presnosť predpovede *83,34 %* a zhodnotenie *18,15 %* dosiahla metóda *BOSSVS*. Zredukovanie hodnôt atribútu na symboly je teda možné brať ako výhodnú reprezentáciu pre riešenú úlohu. Najhoršiu presnosť klasifikácie dosiahol frekvenčný klasifikátor *RISE*. Dôvodom je nevhodnosť frekvenčnej charakteristiky ako zdrojových údajov.

Časovo a pamäťovo najefektívnejší výpočet dosahujú vzdialenostné klasifikátory *KNN* s jednoducho spočítateľnými metrikami variantov Euklidových vzdialeností. Naopak, časovo a pamäťovo najnáročnejšie vychádzajú metódy neurónových sietí. Vysokú zdrojovú

náročnosť z dôvodu hľadania odlišných podsekvencií dosahuje aj tvarová metóda *ST*. Nízku presnosť tejto metódy pre zvolený problém spôsobila nevhodnosť použitia odlišných podsekvencií k úlohe klasifikácie.

Z vlastností a výsledkov klasifikačných metód sú vyvodené nasledovné závery: Ak je pri klasifikácií snaha o maximalizáciu úspešnosti, je vhodné využiť klasifikátor *BOSSVS*. Ak je potreba dosiahnuť rýchlejšiu klasifikáciu bez výraznej straty úspešnosti, je vhodné použiť klasifikátor *TSF*. Klasifikátor *KNN* s metrikami Euklidových variantov je vybratý v prípade, že je cieľom maximálna rýchlosť klasifikácie.

6.5 Zhlukovanie/Vyhľadávanie časových radov

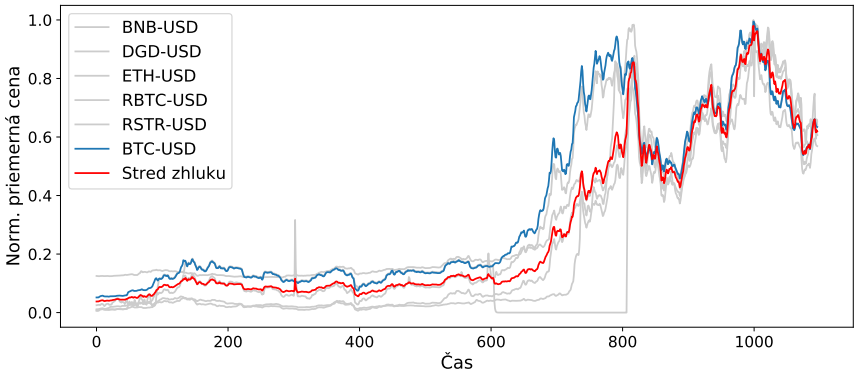
Cieľom posledného experimentu je pomocou úloh zhlukovania a vyhľadávania analyzovať podobnosti a odlišnosti v rámci kryptomien. V experimente je upriamená pozornosť na typ zdrojových údajov a použité metriky. Pre úlohy sú konkrétne demonštrované a porovnané metódy surových údajov, extrahovaných údajov a modelov. Výsledkom tohto experimentu je okrem nájdených podobností/odlišností aj porovnanie jednotlivých transformácií a metrík.

Ako zdroj údajov je zvolený atribút `avgNorm` (priemer. norm. cena). Veľká časť kryptomien obsahuje v období staršom ako tri roky neplatné hodnoty. Pre zvýšenie presnosti sú teda staršie hodnoty odstránené. Zhlukovanie je vo všetkých prípadoch realizované hierarchickým spôsobom. Vzdialenosti medzi kryptomenami sú počítané na základe konkrétnej metriky/typu údajov, viď ďalšie sekcie. Výsledný počet zhlukov v rámci metrík je z dôvodu ich porovnania jednotne stanovený na hodnotu *30*. Matica prepojení je vypočítaná na základe kritéria maximálnej vzdialenosti funkciou `complete` z balíka *Scipy*. Rozdelenie do zhlukov je realizované funkciou `fcluster` z rovnakého balíka. Pre kryptomeny nie sú vopred známe triedy, a tak je výsledok vyhodnotený pomocou metrík *SS (Silhouette score)*, *CHS (Calinski-Harabasz score)* a *DBS (Davies-Bouldin score)*. Ich vysvetlenie je uvedené v podkapitole 3.3.3. Vyhodnotenie navyše obsahuje informáciu o spotreb. čase/pamäti a veľkosti zhluku (N) kryptomeny *Bitcoin*. Vyhľadávanie je vykonané pre *5* najbližších a *5* najvzdialenejším kryptomien. Výsledky sú prezentované z pohľadu kryptomeny *Bitcoin*.

6.5.1 Metódy surových údajov

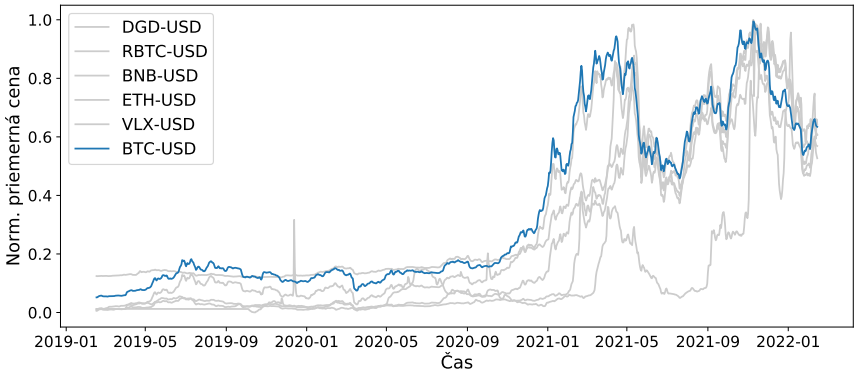
Už názov predznamenáva, že vyhľadávanie a zhlukovanie surových údajov je realizované nad údajmi, ktoré nie sú nijako predspracované. Odlišnosť medzi týmto typom údajov posudzuje vzdialenostná funkcia, ktorá môže byť rôzna.

Pre demonštráciu je možné zvoliť vzdialenostnú funkciu ***DTW (Dynamic Time Wraping)***. Implementácia výpočtu *DTW* vzdialenosti je zapuzdrená vo funkcií `dtw` z balíka *Dtaidistance*. Výsledok zhlukovania pre túto funkciu dosiahol presnosť *SS=0,065*, *CHS=39,110* a *DBS=1,840*. Výpočet trval *16104,12* sekundy a spotreboval *0,94* pamäte. Vizualizácia výsledku v podobe zhluku kryptomeny *Bitcoin* je uvedená na obrázku 6.25.



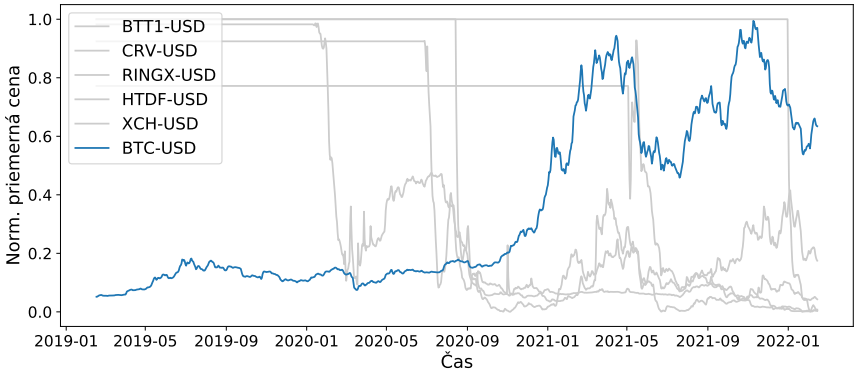
Obr. 6.25: Zhluk kryptomeny *Bitcoin* (*DTW* vzdialenosť)

Z výsledku vyhľadávania vyšlo, že *5* najpodobnejších kryptomien podľa *DTW* vzdialenosti pre *Bitcoin* sú `DGD`, `RBTC`, `BNB`, `ETH` a `VLX`. Vypočítaný zhluk tak obsahuje všetky, s výnimkou poslednej, čo značí pomerne vysokú presnosť hier. zhlukovania. Najpodobnejšie kryptomeny sú vizualizované na obrázku 6.26.



Obr. 6.26: Vyhľadávanie podobností kryptomeny *Bitcoin* (*DTW* vzdialenosť)

Naopak, najodľahlejšie vyšli kryptomeny `BTT1`, `CRV`, `RINGX`, `HTDF` a `XCH`, viď obrázok 6.27.



Obr. 6.27: Vyhľadávanie odľahlostí kryptomeny *Bitcoin* (*DTW* vzdialenosť)

Vyhľadávanie a zhlukovanie surových údajov je možné realizovať s množstvom iných vzdialenostných funkcií. Výsledky zhlukovania pre všetky implementované vzdialenosti sú uvedené v tabuľke 6.9. Tabuľka informuje, že z pohľadu metrík vyšlo ako najpresnejšie zhlukovanie so štvorcovou Euklidovou vzdialenosťou. Najhorší výsledok dosiahlo zhlukovanie s Jensen-Shannonovou vzdialenosťou. Časovo najefektívnejším je zhlukovanie s korelačnou vzdialenosťou. Naopak, najmenej časovo efektívne je zhlukovanie s vzdialenosťou *DTW*. Z pohľadu pamäťovej náročnosti dosahuje každé zhlukovanie rovnaké výsledky. Význam všetkých použitých vzdialenostných funkcií je možné nájsť v dokumentácii balíka *Scipy*[5].

	N	*SS*	*CHS*	*DBS*	čas (*s*)	pamäť (*MB*)
corr	*85*	*-0,208*	*11,732*	*3,097*	*0,51*	*0,38*
euclid	*7*	*0,152*	*48,518*	*1,801*	*3,84*	*0,83*
sqeuclid	*3*	*0,276*	*48,826*	*1,434*	*2,72*	*0,70*
cityblock	*3*	*0,192*	*41,607*	*1,809*	*3,79*	*0,82*
dtw	*6*	*0,065*	*39,110*	*1,840*	*16104,13*	*0,94*
softdtw	*5*	*0,032*	*31,754*	*1,967*	*8302,35*	*0,86*
ctw	*6*	*0,061*	*37,814*	*1,878*	*8187,81*	*0,96*
braycurtis	*8*	*-0,009*	*23,690*	*1,829*	*3,98*	*0,69*
canberra	*12*	*-0,116*	*23,539*	*2,995*	*7,62*	*0,76*
chebyshev	*25*	*0,055*	*21,676*	*2,660*	*16,77*	*0,70*
cosine	*42*	*-0,269*	*8,432*	*3,610*	*8,75*	*0,76*
jensen	*45*	*-0,300*	*7,680*	*3,615*	*15,66*	*0,75*
minkowski	*7*	*0,152*	*48,518*	*1,801*	*3,58*	*0,77*

Tabuľka 6.9: Výsledky zhlukovania nad surovými údajmi (kryptomena *Bitcoin*)

Výsledky vyhľadávania nad surovými údajmi sú uvedené v tabuľke 6.9. Tabuľka informuje o tom, že medzi nájdenými podobnosťami sa najčastejšie vyskytujú kryptomeny `RBTC`, `DGD`, `TOMO`, `KAVA` a `LINK` s počtom *13*, *11*, *10*, *7* a *4*. Medzi najodľahlejšími sa najčastejšie vyskytujú `XCH`, `HTDF`, `CRV`, `BTT1` s počtom *7* a `RINGX` s počtom *6*.

	najpodobnejšie	najodľahlejšie
euclid	`RBTC DGD TOMO LINK KAVA`	`BTT1 CRV RINGX HTDF XCH`
dtw	`DGD RBTC BNB ETH VLX`	`BTT1 CRV RINX HTDF XCH`
corr	`RBTC CVC DFI GLM SBD`	`HNT1 MASS MHC MIR IOC`
sqeuclid	`RBTC DGD TOMO LINK KAVA`	`BTT1 CRV RINGX HTDF XCH`
cityblock	`RBTC DGD TOMO RSTR KAVA`	`CRV BTT1 RINGX HTDF XCH`
softdtw	`RBTC DGD TOMO BNB ETH KAVA`	`BTT1 CRV RINGX HTDF XCH`
ctw	`DGD RBTC BNB ETH VLX`	`BTT1 CRV RINGX HTDF XCH`
braycurtis	`RBTC DGD TOMO RSTR KAVA`	`HNT1 BCA FRST LCC ATB`
canberra	`DGD LTC RBTC SUSHI TOMO`	`HNT1 BCA FRST BNFR CSC`
chebyshev	`RBTC TOMO NYE KAVA CELO`	`BNFR HNT1 BCA FRST BPS`
cosine	`RBTC DMD LINK DGD TOMO`	`BNFR PZM BPS DDK DNA1`
jensen	`RBTC DGD XMR TOMO TRX`	`BNFR PZM BPS DNA1 DDK`
minkowski	`RBTC DGD TOMO LINK KAVA`	`BTT1 CRV RINGX HTDF XCH`

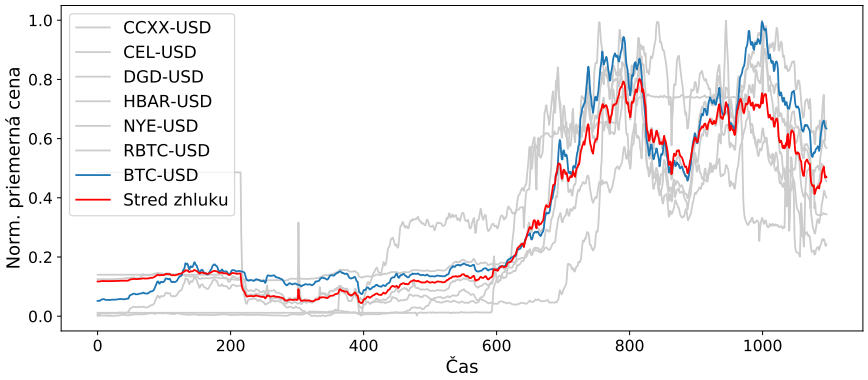
Tabuľka 6.10: Výsledky vyhľadávania nad surovými údajmi (kryptomena *Bitcoin*)

[5] https://docs.scipy.org/doc/scipy/reference/spatial.distance.html

6.5.2 Metódy extrahovaných/transformovaných údajov

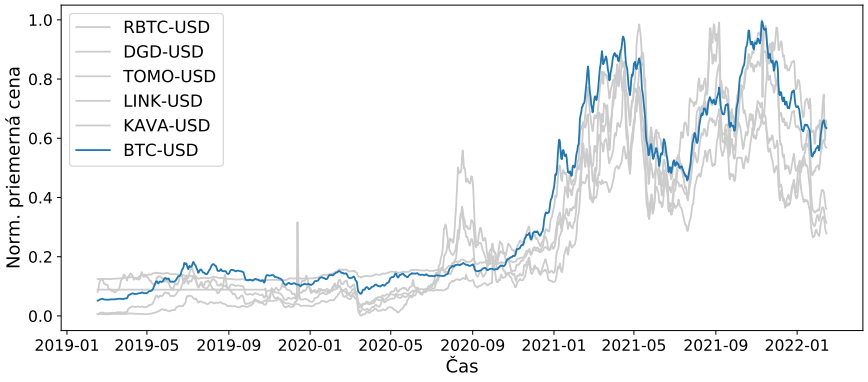
Skupina týchto metód realizuje zhlukovanie nad extrahovanými/transformovanými údajmi. Typ extrahovaných/transformovaných údajov môže byť rôzny.

Pre demonštráciu je možné zvoliť **diskrétnu vlnkovú transformáciu (*DWT*)**. Transformácia je vypočítaná funkciou `dwt` z balíka *Pywt*. Ako metriku pre odhalenie podobností je možné zvoliť Euklidovu vzdialenosť. Zhlukovanie so zvolenou transformáciou a metrikou dosahuje presnosť *SS=0,152*, *CHS=48,518* a *DBS=1,801*. Výpočet trvá *3,45* sekundy, pričom spotrebovaných je *1,05 MB* pamäte. Výsledok zhlukovania v podobe zhluku kryptomeny *Bitcoin* pre uvedený typ údajov a metriku je uvedený na obrázku 6.28.



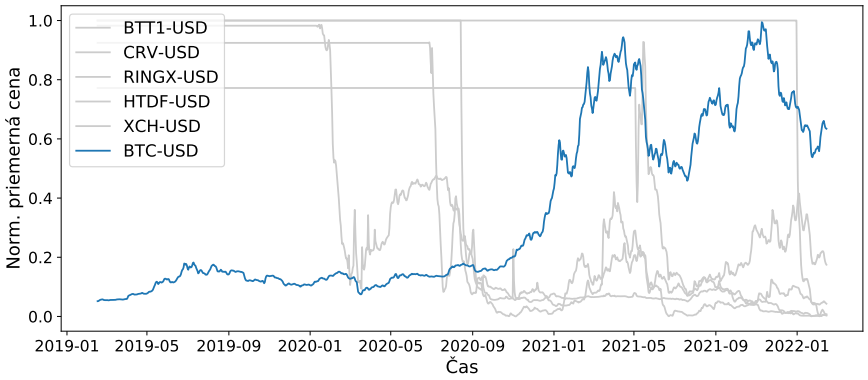
Obr. 6.28: Zhluk kryptomeny *Bitcoin* (Euklid. vzdial. transf. *DWT*)

Podľa Euklidovej vzdialenosti transformácie *DWT* vyšlo pre *Bitcoin* ako *5* najbližších kryptomien `RTBC`, `DGD`, `TOMO`, `LINK` a `KAVA`. To znamená, že v rámci zhlukovej analýzy sa do zhluku nepodarilo zaradiť *3* z *5* najbližších kryptomien, čo značí podpriemernú presnosť hier. zhlukovania. Najbližšie kryptomeny sú vizualizované na obrázku 6.29.



Obr. 6.29: Vyhľadávanie podob. kryptomeny *Bitcoin* (Euklid. vzdial. transf. *DWT*)

Naopak, ako najvzdialenejšie vyšli kryptomeny `BTT1`, `CRV`, `RINGX`, `HTDF` a `XCH`, zobrazené na obrázku 6.30.



Obr. 6.30: Vyhľadávanie odľah. kryptomeny *Bitcoin* (Euklid. vzdial. transf. *DWT*)

Nad časovými radmi je možné extrahovať/transformovať množstvo ďalších typov údajov. Výsledky zhlukovania všetkých implementovaných extrakcií/transformácií sú uvedené v tabuľke 6.11. Pre všetky extrakcie/transformácie s výnimkou symbolických je použitá Euklidova vzdialenosť. Vzdialenostnou funkciou symbolických reprezentácií je Hammingova vzdialenosť. Najpresnejší výsledok dosiahlo zhlukovanie redukčnej transformácie *PAA (Piecewise Aggregate Approximation)*, frekvenčnej transformácie *DWT (Discrete Wavelet Transform)* a symbolickej transformácie *SAX (Symbolic Aggregate Approximation)*. Naopak, najhorší výsledok vyšiel pre zhlukovanie čiastočnej autokorelačnej transformácie. Z pohľadu náročnosti na zdroje dosahuje najlepší výsledok zhlukovanie redukčnej transformácie *PAA*. Časovo a pamäťovo najmenej efektívne je zhlukovanie symbolickej transformácie *BOW (Bag Of Words)*. Význam všetkých použitých transformácií je možné nájsť v popise aplikačného rozhrania balíkov *Pyts*[6] a *Tslearn*[7].

	N	*SS*	*CHS*	*DBS*	čas (*s*)	pamäť (*MB*)
AC	*13*	*-0,198*	*5,243*	*2,326*	*2,95*	*1,37*
PAC	*268*	*-0,557*	*0,955*	*5,176*	*15,15*	*1,30*
PAA	*4*	*0,193*	*41,122*	*2,094*	*2,75*	*0,84*
DFT	*3*	*0,179*	*41,491*	*2,069*	*4,44*	*0,76*
DWT	*7*	*0,152*	*48,518*	*1,801*	*3,45*	*1,05*
MCB	*8*	*-0,098*	*14,574*	*3,587*	*63,63*	*5,79*
SFA	*4*	*-0,286*	*7,680*	*4,719*	*65,75*	*15,24*
SAX	*4*	*0,185*	*41,411*	*1,785*	*3,43*	*0,75*
BOW	*6*	*-0,348*	*5,117*	*5,979*	*358,2*	*127,96*
BOP	*5*	*-0,352*	*5,489*	*5,485*	*5,66*	*93,91*
BOSS	*9*	*-0,250*	*7,332*	*4,102*	*5,13*	*15,09*
ROCKET	*5*	*-0,095*	*16,325*	*3,721*	*14,18*	*20,09*

Tabuľka 6.11: Výsledky metód pre extrah./transf. údaje (kryptomena *Bitcoin*)

Výsledky vyhľadávania pre všetky typy implementovaných extrakcií/transformácií a metrík sú uvedené v tabuľke 6.12. Z tabuľky je možné pozorovať, že medzi nájdenými podobnosťami sa najčastejšie vyskytujú kryptomeny `RBTC`, `DGD` s počtom *9* a *6*, `ETH`, `DFI`

[6] https://pyts.readthedocs.io/en/stable/api.html
[7] https://tslearn.readthedocs.io/en/stable/reference.html

s počtom *4* TOMO, LINK s počtom *3*. Medzi nájdenými odľahlosťami sa najčastejšie vyskytujú BTT1, DGD s počtom *7* a *6*, HTDF, CRV s počtom *5* a RINGX s počtom *4*.

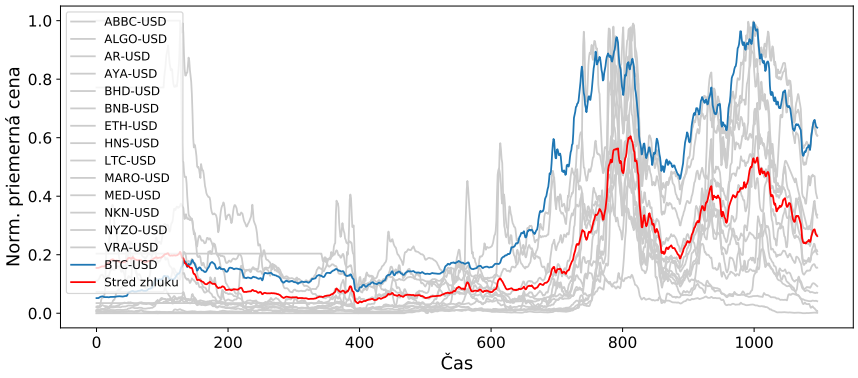
	najpodobnejšie	najodľahlejšie
AC	RBTC DFI LINK GLM ETH	SOLVE CRV BLOCK NIX DACC
PAC	ETH RSTR HIVE XTZ BCH	DNT XDC ZYN GHOST1 AAVE
PAA	RBTC DGD TOMO CEL KAVA	BTT1 CRV RINGX HTDF XCH
DFT	RBTC DGD TOMO NYE KAVA	BTT1 CRV RINGX HTDF XCH
DWT	RBTC DGD TOMO LINK KAVA	BTT1 CRV RINGX HTDF XCH
MCB	RSTR DGD SUSHI FO BAL	NXS PPT WINGS DYN WAXP
SFA	ETH BNB AVAX CRO DFI	VBK BHD BTT1 DDK HTDF
SAX	RBTC CCX CEL DFI AAVE	BTT1 RINGX CRV HTDF XCH
BOW	GLM KIN HYC RBTC DCR	HNT1 BNFR BTT1 STX1 SOL1
BOP	RBTC LINK DCR NYE XEM	HNT1 BNFR BTT1 STX1 SOL1
BOSS	RBTC LTC XLM RSTR DGD	SUB ATB FRST CSC DCN
ROCKET	RBTC ETH DFI DGD ENJ	HNT1 BCA FRST CRV SUB

Tabuľka 6.12: Výsledky metód pre extrah./transf. údaje (kryptomena *Bitcoin*)

6.5.3 Metódy založené na modeloch

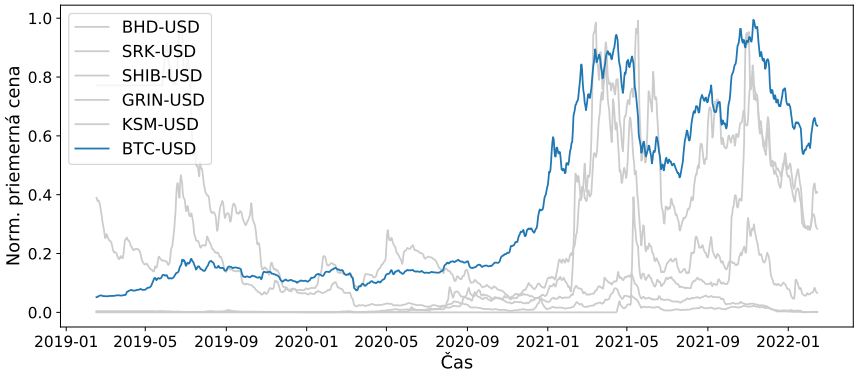
Tento typ vyhľadávania a zhlukovania využíva ako zdrojové údaje parametre modelov vytvorených nad časovými radmi.

Konkrétnym príkladom je model predpovede ***AR (Autoregressive)***. Pre každú kryptomenu je natrénovaný vlastný *AR* model, pričom zhlukovanie/vyhľadávanie je následne realizované nad extrahovanými koeficientami lineárnej funkcie dĺžky *3*. Výsledok zhlukovania nad koeficientami modelu *AR* dosiahol presnosť *SS=-0,467*, *CHS=2,447* a *DBS=6,176* s dĺžkou výpočtu *153,83* sekúnd a spotrebovaným množstvom pamäte *22,91 MB* . Vizualizácia je v podobe zhluku obsahujúceho kryptomenu *Bitcoin* uvedená na obrázku 6.31.



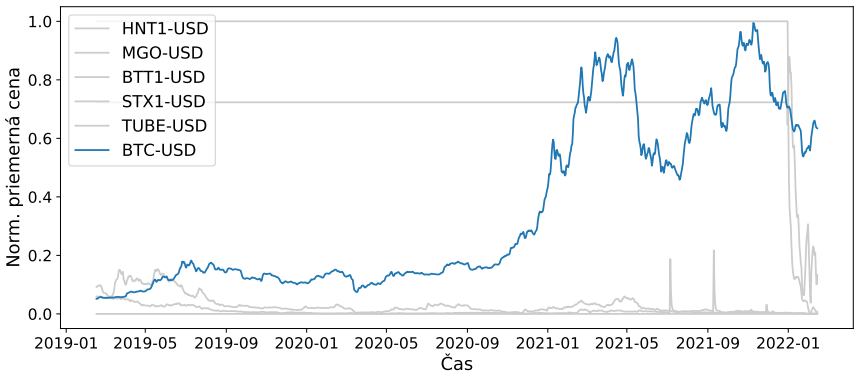
Obr. 6.31: Zhluk kryptomeny *Bitcoin* (Euklidova vzdialenosť koeficientov modelu *AR*)

Z vyhľadávania podobností vyplynulo, že najviac podobnými kryptomenami pre koeficienty modelu *AR* sú BHD, SRK, SHIB, GRIN a KSM. V zhluku je zahrnutá iba *1* z *5* najbližších kryptomien, čo značí nízku presnosť zhlukovania. Najbližšie kryptomeny sú vizualizované na obrázku 6.32.



Obr. 6.32: Vyhľadávanie podob. kryptomeny *Bitcoin* (Euklidova vzdial. koef. modelu *AR*)

Naopak, najvzdialenejšími kryptomenami sú `HNT1`, `MGO`, `BTT1`, `STX1` a `TUBE`, viď obrázok 6.33.



Obr. 6.33: Vyhľadávanie odľah. kryptomeny *Bitcoin* (Euklidova vzdial. koef. modelu *AR*)

Okrem *AR* modelu bolo zhlukovanie otestované aj pre modely *MA* a *ARMA*. Výsledky pre všetky otestované modely sú uvedené v tabuľke 6.13. Najpresnejšie výsledky dosiahlo zhlukovanie modelu *MA*, najhoršie zhlukovanie modelu *ARMA*. Časovo a pamäťovo najefektívnejšie je zhlukovanie *AR*, naopak najnáročnejšie je zhlukovanie *ARMA*. Vyplýva to zo skutočnosti, že v modeli *ARMA* sú hľadané koeficienty pre lineárnu a aj lineárnu chybovú funkciu.

	N	*SS*	*CHS*	*DBS*	čas (s)	pamäť (MB)
AR	*15*	*-0,467*	*2,447*	*6,176*	*153,83*	*22,91*
MA	*4*	*-0,358*	*7,394*	*5,480*	*404,53*	*31,68*
ARMA	*7*	*-0,492*	*1,576*	*7,063*	*540,34*	*47,27*

Tabuľka 6.13: Výsledky zhlukovania/vyhľadávania pre modely (*Bitcoin*)

Výsledky vyhľadávania sú uvedené v tabuľke 6.14. Tabuľka informuje, že vyhľadávanie je takmer úplne odlišné pre všetky otestované modely. Túto skutočnosť je možné pripísať odlišnostiam extrahovaných koeficientov jednotlivých modelov.

	najpodobnejšie	najodľahlejšie
AR	BHD SRK SHIB GRIN KSM	HNT1 MGO BTT1 STX1 TUBE
MA	DDK LRG DNA1 RINGX BNB	HNT1 BCA MGO FAIR USDT
ARMA	LTC MLN VSYS NKN SLS	AAVE XCH BHD XDC ALGO

Tabuľka 6.14: Výsledky zhlukovania/vyhľadávania pre modely (*Bitcoin*)

6.5.4 Zhodnotenie zhlukovania/vyhľadávania

V experimente bolo otestovaných celkovo *28* modelov zhlukovania z rôznych skupín. Odlišnosti medzi jednotlivými modelmi potvrdili aj dosiahnuté výsledky.

Najvyššiu presnosť zo všetkých modelov dosiahlo zhlukovanie využívajúce štvorcovú Euklidovu vzdialenosť nad surovými údajmi s metrikami presnosti dosahujúcimi hodnoty *SS=0,276 CHS=48,826* a *DBS=1,434*. Pre zarovnané časové rady kryptomien sa preto oplatí využiť práve tento typ vzdialenostnej funkcie. Metódy surových údajov takisto všeobecne vykazujú najefektívnejšiu prácu s pamäťou, nakoľko nie je potrebná žiadna dodatočná transformácia/extrakcia údajov. Ako časovo najefektívnejšie sa ukázalo korelačné zhlukovanie surových údajov a zhlukovanie využívajúce redukčnú transformáciu *PAA (Piecewise Aggregate Approximation)*. Metódy zhlukovania založených na modeloch dosiahli všeobecne najnižšiu presnosť a výsledky nájdených podobností/odľahlostí sa pre jednotlivé modely vo vysokej miere odlišujú. Ich výsledky je teda vhodné použiť iba pre ďalšiu analýzu a prácu s modelom, nad ktorým sú realizované.

Okrem porovnania metód je výstupom experimentu aj zisk najviac podobných/odlišných kryptomien. Ako najviac podobné kryptomeny pre kryptomenu *Bitcoin* vyšli *RSK Smart Bitcoin DigixDAO*, *DeFiChain*, *TomoChain*, *Chainlink*. Najviac odlišnými naopak sú *BitTorrent*, *Orient Walt*, *Curve DAO Token*, *Chia* a *RINGX*. Najviac podobné kryptomeny je možné využiť napríklad pre dodatočné trénovanie modelov s cieľom zvýšenia presnosti. Najviac odlišné kryptomeny je naopak možné z dátovej sady odstrániť a zvýšiť tak presnosť ďalších analýz a modelov.

Kapitola 7

Záver

Práca sa zaoberala problematikou získavania znalostí z časových radov. Pozornosť bola venovaná hlavne úlohám predpovede/regresie, klasifikácie, zhlukovaniu a vyhľadávaniu podobností/odlišností. Okrem samotných úloh boli objasnené základné myšlienky, princípy, postupy a metódy, ktoré so ziskom znalostí z časových radov súvisia. Práca v podobe popisu dostupných balíkov predstavila podporu programovacieho jazyka *Python* pre časové rady. Programovací jazyk *Python* bol použitý pre implementáciu troch experimentov dolovania.

Ako zdrojové údaje pre experimenty bola zvolená dátová sada *377* kryptomien. Dátovú sadu bolo potrebné pre jednotlivé experimenty vhodne predspracovať a zanalyzovať.Prvý experiment bol zameraný na predpoveď budúcich cien kryptomien. Konkrétne bolo vytvorených *21* rôznych modelov predpovede dennej percentuálnej zmeny ceny na najbližších *30* dní. Najlepší výsledok z pohľadu presnosti predpovede dosiahla metóda *SARIMAX* so strednou absolútnou percentuálnou chybou *0.012*. Vzhľadom na úspešnosť predpovede zisku/poklesu ceny dosiahol najlepší výsledok model *HWES* s úspešnosťou *86,67 %*. Cieľom druhého experimentu bolo vykonať klasifikáciu nárastu/poklesu cien kryptomien. Z celkovo *13* vytvorených klasifikačných modelov dosiahol najvyššiu úspešnosť *83,34 %* model *BOSSVS*. V treťom experimente bolo implementovaných *28* modelov zhlukovania/vyhľadávania podobností/odľahlostí atribútu normalizovanej priemernej dennej ceny všetkých kryptomien. Ako najpresnejšie sa ukázalo podľa zvolených metrík zhlukovanie surových údajov so štvorcovou Euklidovou vzdialenostnou funkciou. Najviac podobné kryptomeny pre *Bitcoin* vyšli *RSK Smart Bitcoin DigixDAO*, *DeFiChain*, *TomoChain*, *Chainlink* a najviac odlišné *BitTorrent*, *Orient Walt*, *Curve DAO Token*, *Chia* a *RINGX*.

V priebehu práce som rozšíril svoje poznatky v oblasti získavania znalostí z časových radov. Rovnako som si osvojil prácu s využitými nástrojmi, programovacími jazykmi, balíkmi a iným softvérom. Výstupom práce je porovnanie metód pre jednotlivé úlohy a aplikácia implementujúca zvolené metódy. Aplikácia okrem demonštrovania a porovnania metód dolovania umožnila analýzu zdrojových údajov. Prácu by v budúcnosti bolo možné rozšíriť o pridanie ďalších kryptomien, modelov a rôznych analýz. Presnosť vytvorených modelov by bolo možné zvýšiť rozšírením na vektorové varianty alebo pridaním najpodobnejších kryptomien do procesu trénovania. Iným, pokročilejším rozšírením by zase mohlo byť využitie vytvorených modelov pre implementovanie automatizovaného strategického rozhodovania.

Literatúra

[1] Aghabozorgi, S., Shirkhorshidi, A. S. a Wah, T. Y. Time-series clustering–a decade review. *Information Systems*. Elsevier. 2015, zv. 53, č. 1, s. 16–38. ISSN 1365-2575.

[2] Azzouzi, M. a Nabney, I. T. Analysing time series structure with Hidden Markov Models. In: *Neural Networks for Signal Processing VIII*. Piscataway, NJ: IEEE, 1998, s. 402–408. ISBN 0-7803-5060-X.

[3] Bagnall, A., Ratanamahatana, C., Keogh, E., Lonardi, S., Janacek, G. et al. A bit level representation for time series data mining with shape based similarity. *Data mining and knowledge discovery*. Springer. 2006, zv. 13, č. 1, s. 11–40. ISSN 1384-5810.

[4] Bauwens, L., Laurent, S. a Rombouts, J. V. Multivariate GARCH models: a survey. *Journal of applied econometrics*. Wiley Online Library. 2006, zv. 21, č. 1, s. 79–109. ISSN 0883-7252.

[5] Bergmeir, C. a Benítez, J. M. On the use of cross-validation for time series predictor evaluation. *Information Sciences*. Elsevier. 2012, zv. 191, č. 1, s. 192–213. ISSN 0020-0255.

[6] Bharati, M. a Ramageri, M. Data mining techniques and applications. *Indian Journal of Computer Science and Engineering*. Citeseer. 2010, zv. 1, č. 4, s. 301–305. ISSN 0976-5166.

[7] Brockwell, P. J., Brockwell, P. J., Davis, R. A. a Davis, R. A. *Introduction to time series and forecasting*. 3. vyd. Switzerland: Springer, 2016. ISBN 978-3-319-29852-8.

[8] Caiado, J., Crato, N. a Peña, D. A periodogram-based metric for time series classification. *Computational Statistics & Data Analysis*. Elsevier. 2006, zv. 50, č. 10, s. 2668–2684. ISSN 167-9473.

[9] Caliński, T. a Harabasz, J. A dendrite method for cluster analysis. *Communications in Statistics*. Taylor Francis. 1974, zv. 3, č. 1, s. 1–27. ISSN 0090-3272.

[10] Chakrabarti, K., Keogh, E., Mehrotra, S. a Pazzani, M. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems (TODS)*. ACM New York, NY, USA. 2002, zv. 27, č. 2, s. 188–228. ISSN 0362-5915.

[11] Chatfield, C. *Time-series forecasting*. 1. vyd. Boca Raton, Florida, USA: CRC press, 2000. ISBN 978-1584880639.

[12] Cleveland, R. B., Cleveland, W. S., McRae, J. E. a Terpenning, I. STL: A seasonal-trend decomposition. *Journal of Official Statistics*. 1990, zv. 6, č. 1, s. 3–73. ISSN 0282-423X.

[13] Davies, D. L. a Bouldin, D. W. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE. 1979, PAMI-1, č. 2, s. 224–227. ISSN 0162-8828.

[14] Deng, H., Runger, G., Tuv, E. a Vladimir, M. A time series forest for classification and feature extraction. *Information Sciences*. Elsevier. 2013, zv. 239, č. 1, s. 142–153. ISSN 0020-0255.

[15] Du Preez, J. a Witt, S. F. Univariate versus multivariate time series forecasting: an application to international tourism demand. *International Journal of Forecasting*. Elsevier. 2003, zv. 19, č. 3, s. 435–451. ISSN 0169-2070.

[16] Dunham, M. H. *Data mining: Introductory and advanced topics*. 1. vyd. New Jersey: Pearson Education PTR, 2006. ISBN 0-13-088892-3.

[17] Fink, E. a Gandhi, H. S. Compression of time series by extracting major extrema. *Journal of Experimental & Theoretical Artificial Intelligence*. Taylor & Francis. 2011, zv. 23, č. 2, s. 255–270. ISSN 1362-3079.

[18] Flynn, M., Large, J. a Bagnall, T. The contract random interval spectral ensemble (c-RISE): the effect of contracting a classifier on accuracy. In: *International Conference on Hybrid Artificial Intelligence Systems*. Switzerland: Springer, Cham, 2019, s. 381–392. ISBN 978-3-030-29858-6.

[19] Fu, T. chung. A review on time series data mining. *Engineering Applications of Artificial Intelligence*. International Federation of Automatic Control. 2011, zv. 24, č. 1, s. 164–181. ISSN 0952-1976.

[20] Granger, C. W. J. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica*. [Wiley, Econometric Society]. 1969, zv. 37, č. 3, s. 424–438. ISSN 0012-9682.

[21] Grieser, J., Trömel, S. a Schönwiese, C.-D. Statistical time series decomposition into significant components and application to European temperature. *Theoretical and applied climatology*. Springer. 2002, zv. 71, č. 3, s. 171–183. ISSN 0177-798X.

[22] Gullo, F. From patterns in data to knowledge discovery: what data mining can do. *Physics Procedia*. Elsevier. 2015, zv. 62, č. 62, s. 18–22. ISSN 1875-3892.

[23] Han, J., Pei, J. a Kamber, M. *Data mining: concepts and techniques*. 3. vyd. Waltham: Morgan Kaufmann Publishers, 2012. ISBN 978-0-12-381479-1.

[24] Hanna, M. Data mining in the e-learning domain. *Campus-wide information systems*. Emerald Group Publishing Limited. 2004, zv. 21, č. 1, s. 29–34. ISSN 1065-0741.

[25] Kantardzic, M. *Data mining: concepts, models, methods, and algorithms*. 2. vyd. New Jersey: John Wiley & Sons, 2011. ISBN 978-0-470-89045-5.

[26] Karim, F., Majumdar, S., Darabi, H. a Chen, S. LSTM fully convolutional networks for time series classification. *IEEE access*. IEEE. 2017, zv. 6, č. 1, s. 1662–1669. ISSN 2169-3536.

[27] Keogh, E., Chu, S., Hart, D. a Pazzani, M. Segmenting time series: A survey and novel approach. In: *Data mining in time series databases*. 1. vyd. 27 Warren Street, Suite 401-402, Hackensack, NJ 07601: World Scientific, 2004, s. 1–21. ISBN 981-238-290-9.

[28] Keogh, E., Lin, J., Lee, S.-H. a Van Herle, H. Finding the most unusual time series subsequence: algorithms and applications. *Knowledge and Information Systems*. Springer. 2007, zv. 11, č. 1, s. 1–27. ISSN 0219-1377.

[29] Keogh, E. J. a Pazzani, M. J. Scaling up dynamic time warping for datamining applications. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: Association for Computing Machinery, 2000, s. 285–289. ISBN 978-1-58113-233-5.

[30] Kuhlman, D. *A python book: Beginning python, advanced python, and python exercises*. 1. vyd. 725 8th St. SE, Washington, D. C.: Platypus Global Media, 2011. ISBN 978-0984221233.

[31] Kwiatkowski, D., Phillips, P. C., Schmidt, P. a Shin, Y. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*. Elsevier. 1992, zv. 54, 1-3, s. 159–178. ISSN 0304-4076.

[32] Liao, T. W. Clustering of time series data—a survey. *Pattern recognition*. Elsevier. 2005, zv. 38, č. 11, s. 1857–1874. ISSN 0031-3203.

[33] Lines, J., Davis, L. M., Hills, J. a Bagnall, A. A shapelet transform for time series classification. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: Association for Computing Machinery, 2012, s. 289–297. ISBN 978-1-4503-1462-6.

[34] Nason, G. P. Stationary and non-stationary time series. *Statistics in volcanology*. Geological Society of London. 2006, zv. 60, č. 6, s. 129–142. ISSN 1750-8207.

[35] Percival, D. B. a Walden, A. T. *Wavelet methods for time series analysis*. 4. vyd. New York: Cambridge university press, 2000. ISBN 978-0-521-64068-8.

[36] Piccolo, D. A distance measure for classifying ARIMA models. *Journal of time series analysis*. Wiley Online Library. 1990, zv. 11, č. 2, s. 153–164. ISSN 0143-9782.

[37] Pratama, I., Permanasari, A. E., Ardiyanto, I. a Indrayani, R. A review of missing values handling methods on time-series data. In: *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*. Bandung, Indonesia: IEEE, 2016, s. 1–6. ISBN 978-1-5090-2450-6.

[38] Rai, P. a Singh, S. A survey of clustering techniques. *International Journal of Computer Applications*. Foundation of Computer Science. 2010, zv. 7, č. 12, s. 1–5. ISSN 0975-8887.

[39] Ramakrishnan, R. a Gehrke, J. *Database management systems*. 3. vyd. New York: McGraw Hill, 2003. ISBN 0-07-246563-8.

[40] Rehfeld, K., Marwan, N., Heitzig, J. a Kurths, J. Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics*. Copernicus GmbH. 2011, zv. 18, č. 3, s. 389–404. ISSN 1023-5809.

[41] Richman, J. S. a Moorman, J. R. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*. American Physiological Society Bethesda, MD. 2000, zv. 278, č. 6, s. 2039–2049. ISSN 0363-6135.

[42] Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*. Elsevier. 1987, zv. 20, č. 1, s. 53–65. ISSN 0377-0427.

[43] Sanner, M. F. et al. Python: a programming language for software integration and development. *Journal of molecular graphics & modelling*. Elsevier. 1999, zv. 17, č. 1, s. 57–61. ISSN 1093-3263.

[44] Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*. Springer. 2015, zv. 29, č. 6, s. 1505–1530. ISSN 1384-5810.

[45] Ševcech, J. Towards Symbolic Representation of Potentially Infinite Time Series. *Information Sciences and Technologies Bulletin of the ACM Slovakia*. STU Press. 2017, zv. 9, č. 1, s. 1–5. ISSN 1338-1237.

[46] Shatkay, H. a Zdonik, S. B. Approximate Queries and Representations for Large Data Sequences. In: *Proceedings of the Twelfth International Conference on Data Engineering*. Los Alamitos, CA, USA: IEEE Computer Society, 1996, s. 536. ISSN 1063-6382.

[47] Shumway, R. H., Stoffer, D. S. a Stoffer, D. S. *Time series analysis and its applications*. 4. vyd. Cham: Springer, 2017. ISBN 978-3-319-52451-1.

[48] Sundararajan, D. *The discrete Fourier transform: theory, algorithms and applications*. 1. vyd. London: World Scientific, 2001. ISBN 981-02-4521-1.

[49] Susto, G. A., Cenedese, A. a Terzi, M. Chapter 9 - Time-Series Classification Methods: Review and Applications to Power Systems Data. In: *Big Data Application in Power Systems*. Elsevier, 2018, s. 179–220. ISBN 978-0-12-811968-6.

[50] Teng, M. Anomaly detection on time series. In: *2010 IEEE International Conference on Progress in Informatics and Computing*. Shanghai, China: IEEE, 2010, sv. 1, s. 603–608. ISBN 978-1-4244-6788-4.

[51] Vishwas, B. a Patel, A. *Hands-on Time Series Analysis with Python*. 1. vyd. Berkeley, CA: Apress, 2020. ISBN 978-1-4842-5991-7.

[52] Wu, D., Wang, X., Su, J., Tang, B. a Wu, S. A labeling method for financial time series prediction based on trends. *Entropy*. Multidisciplinary Digital Publishing Institute. 2020, zv. 22, č. 10, s. 1162. ISSN 1099-4300.

[53] Xie, T. a Ding, J. Forecasting with Multiple Seasonality. In: *2020 IEEE International Conference on Big Data (Big Data)*. Atlanta, GA, USA: IEEE, 2020, s. 240–245. ISBN 978-1-7281-6252-2.

[54] Zaki, M. J. a Wong, L. Data mining techniques. In: *Selected Topics in Post-Genome Knowledge Discovery*. 3. vyd. London: World Scientific, 2004, s. 125–163. ISBN 981-238-780-3.

[55] Zheng, J., Xu, C., Zhang, Z. a Li, X. Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network. In: *2017 51st Annual Conference on Information Sciences and Systems (CISS)*. Baltimore, MD, USA: IEEE, 2017, s. 1–6. ISBN 978-1-5090-2697-5.

[56] Åström, K. On the choice of sampling rates in parametric identification of time series. *Information Sciences*. Elsevier. 1969, zv. 1, č. 3, s. 273–278. ISSN 0020-0255.

Príloha A

Obsah priloženého pamäťového média

Súčasťou práce je priložený dátový nosič obsahujúci:

- `DP_xkruty00` – koreňový adresár
  - `experimenty` – adresár obsahujúci experimenty
    - `predspracovanie_analyza.ipynb` – predspracovanie a analýza dátovej sady
    - `predpoved.ipynb` – experiment predpovede
    - `klasifikacia.ipynb` – experiment klasifikácie
    - `zhlukovanie.ipynb` – experiment zhlukovania
    - `vyhladavanie.ipynb` – experiment vyhľadávania
  - `aplikacia` – adresár obsahujúci implementáciu aplikácie, podrobnejší popis v prílohe B
  - `technicka_sprava` – adresár obsahujúci súbory potrebné na vytvorenie technickej správy (preklad otestovaný na školskom serveri *Merlin* a v prostredí *Overleaf*)
  - `README.md` – dokumentácia projektu
  - `dp.pdf` – technická správa vo formáte `pdf` (preložené v prostredí *Overleaf*)
  - `dp_print.pdf` – technická správa vo formáte `pdf` určená na tlač (preložené v prostredí *Overleaf*)

Príloha B

Adresárová štruktúra aplikácie

Implementácia aplikácie je zapuzdrená v adresári `aplikacia` s nasledovnou štruktúrou (uvedené iba relevantné adresáre/súbory):

- `app`
  - `Http`
    - `Cotrollers`
      - `FileController.php` – Implementácia manipulácie s uloženými súbormi
      - `PythonController.php` – Implementácia spúšťania skriptov dolovania
- `resources`
  - `js`
    - `components` – Adresár obsahujúci implementáciu komponent pohľadov
    - `data` – Adresár obsahujúci statické údaje pohľadov
    - `helpers` – Adresár obsahujúci pomocné skripty
    - `layouts` – Adresár obsahujúci implementáciu základnej štruktúry aplikácie
    - `pages` – Adresár obsahujúci implementáciu pohľadov aplikácie
    - `store` – Adresár obsahujúci globálne zdieľané údaje pohľadov
    - `app.js` – Hlavný súbor aplikácie
    - `router.js` – Implementácia smerovania pohľadov v časti *frontendu*
- `routes`
  - `api.php` – Implementácia smerovania asynchrónnych volaní v časti *backendu*
  - `web.php` – Implementácia smerovania pohľadov v časti *backendu*
- `storage`
  - `app`
    - `public`
      - `actualData` – Uloženie údajov kryptomien k aktualizovanému dňu
      - `data` – Uloženie údajov kryptomien ku dňu *14.2.2022*
      - `results` – Uloženie výsledkov úloh dolovania
      - `scripts` – Uloženie skriptov dolovania

Príloha C

Inštalácia a spustenie aplikácie

Postup inštalácie a spustenia vytvorenej aplikácie je nasledovný:

1. Rozbalenie archívu s obsahom práce.
2. Stiahnutie údajov dennej ceny kryptomien z repozitára `crypto-historical-price` na portáli *Kaggle*[1] a presun do adresárov `data` a `actualData`, viď príloha B.
3. Stiahnutie a inštalácia softvéru (napr. *XAMPP*[2]) pre vytvorenie lokálneho webového a databázového serveru.
4. Spustenie databázového a aplikačného webového serveru.
5. Inštalácia *Python* balíkov *backendu* aplikácie príkazom `pip install -r req.txt`.
6. Inštalácia *PHP* balíkov *backendu* aplikácie príkazom `npm install`.
7. Inštalácia *Javascript* balíkov *frontendu* aplikácie príkazom `composer install`.
8. Preklad *frontendu* aplikácie príkazom `npm run dev`.
9. Spustenie behu aplikácie príkazom `php artisan serve`.
10. Prístup na stránku aplikácie vo webovom prehliadači cez odkaz `http://localhost/`.

[1] https://www.kaggle.com/datasets/benjaminpo/crypto-historical-price/
[2] https://www.apachefriends.org/